

Linux Printing Systems

Kwan L. Lowe

November 14, 2002

Contents

I	Introduction	2
II	The Printing System	2
1	Overview	2
1.1	Abstract the print device	2
1.2	Allow multiple clients to access the print device	3
1.3	Provide job control and queue management	3
1.4	Provide and authentication/authorization layer	3
2	What Happens In The Printing Process?	3
III	Linux Print Systems	4
3	LPR/LP	4
3.1	LPR/LP Administration	4
3.1.1	Create the print queue	5
3.1.2	Create the print filter	5
3.2	Print a file	5
4	LPRng	6
4.1	LPRng Administration	7
5	CUPS	7
5.1	CUPS Administration	7
5.1.1	Via the Web Interface	7
5.1.2	From the Command Line	10
5.2	CUPS Print Classes	11
6	Other Front End Programs	11
IV	Printing from Windows to Linux	11
7	Configuring Samba	12
8	Creating a PDF Printer	12
V	References	14
VI	Miscellaneous Notes	15

Part I

Introduction

Printing, unfortunately can sometimes seem a lot more difficult than it really is. In this talk I'll try to cover what happens when you send a file off to be printed, whether through an application print menu or by calling the print spooler directly. In addition to the traditional unix `lpr` printing system, I'll also cover CUPS (the Common Unix Printing System) and LPRng (LPR, the Next Generation) and printing through non-Unix print resources such as the Windows printing facility.

The current version of this document is available at the author's website at:

<http://www.digitalhermit.com/linux/printing/index.html>

Part II

The Printing System

Anyone remember dot matrix printers? At that time, printing was as easy as copying a file to the print device. Though computers have gotten a lot more sophisticated since then, sometimes it appears that the printing process has become much more convoluted and difficult. Why add the relative complexity of a printing system when the old way seemed so much easier?

1 Overview

The printing system does several things:

- Abstracts the printer specific control commands from applications
- Allows multiple users and process access to the print device
- Provides a system of job control and queue management
- Provides an authentication/authorization layer

1.1 Abstract the print device

In the simplest case with a directly connected ASCII capable line printer, you can merely `cat` text files to the appropriate device and watch it print. ¹

For example:

```
[klowe@omega] cat outfile.txt > /dev/lp0
```

¹In fact, even for straight text it's almost always best to pre-process the input so that page breaks, margins, and other options are configured so that the best looking output is achieved.

This works fine if you're printing pure text to a locally attached device that supports it. Modern printers do a lot more, however, and to exploit their features such as graphics, Postscript, HP PCL (Printer Control Language) we need to filter the input to the printer. Of course, not all printers are Postscript or PCL compatible so the print system serves the purpose of translating the input into a format that is understood by the printer. This allows applications to be relatively ignorant of the capabilities and driver requirements of the printer, but still take full advantage of them. On another level, the printing system allows a client to submit jobs to printers across a network or even across the Internet.

1.2 Allow multiple clients to access the print device

The actual print device in the `dev` filesystem, like other devices, can only be used by a single process at a time. Linux gets around this by having one process take ownership of the device and then acting as an arbitrator to the resource. In the case of printing, the `lpd` process takes control of the print device and listens for connections from clients, either local or remote, and processes their submitted jobs.

1.3 Provide job control and queue management

The next big reason for using the print system is related to the first. Printing is a slow process and takes eons in comparison to some of the other things going on in the system. The print server does an important function in *queueing* items for printing. In most cases the server operates on a first-come, first served basis, spooling the jobs to the print device in the order in which they appear but it can also re-order the jobs to be printed, remove jobs from the print queue, reprint jobs, or schedule the job for later printing.

1.4 Provide and authentication/authorization layer

Finally, the printing system limits the print resources to only authorized clients. It can limit when and how often users can submit jobs to the queue. It can limit the size or usage of printers.

2 What Happens In The Printing Process?

We've already seen the trivial example of sending a file directly to a print device. What happens when a print system is used instead?

The first step in the chain is the submitting of the file to the print queue via the `lpr` utility (or whatever the print front-end is for the printing system). Using a utility known as a *magic filter*², `lpr` determines the input file type and prepares

²It's nothing magical, really. The filter determines the file type by inspecting the file and looking for specific file signatures. Sometimes the filter can get confused, leading to interesting printouts

the document for sending to the print device. This requires some knowledge of both the input format and the printer capabilities. For example, if a user sends a pure ASCII text document for printing to a remote Postscript printer, the print system will first convert the text to Postscript using Ghostscript³ then send it across the network to the remote printer. Once the file is in a format compatible with the print device, it is placed in a queue to await its turn.

Part III

Linux Print Systems

In this section we'll take a look at some of the various print systems available for Linux. Though they differ in their functionality, scope, and administration procedures, they do maintain at least some compatibility with traditional commands.

3 LPR/LP

`lpr` and `lp` refer to the traditional Unix print systems. After the Great Schism hinted at by gray bearded Unix old-timers, there were two main print systems, the BSD style and the SysV style. They were, from an end-user standpoint, rather comparable in their abilities but differed in implementation details.

BSD is considered by some to be more network friendly and better able to scale to large workgroups. BSD systems use `lpr` to submit jobs to the queue, `lpq` to show the contents of the queue, and `lprm` to remove jobs from the queue. SysV UNIX uses `lp` to submit, `lpstat` to show queue contents, and `cancel` to remove jobs.

In general, Linux has tended towards the BSD print system in earlier distributions. In more recent offerings from RedHat, SuSe, and Mandrake the print systems install compatibility layers that mimic both BSD and SysV commands. The example in the previous section used commands from the BSD system though the print system is actually LPRng, which we'll discuss next.

3.1 LPR/LP Administration

For an example of LPR/LP administration we'll send a file to be printed to a dummy printer and show what happens to the file as it passes through the system. The first thing we'll do is to create a dummy printer using the old-fashioned tool of a text editor. This dummy printer will do nothing to the input file except echo it verbatim to a temporary file.

³Ghostscript is the name of a set of software that provides an interpreter for the PostScript language and the Adobe Portable Document Format, input modules for reading a variety of raster formats, and output modules for a wide variety of window systems .

3.1.1 Create the print queue

First, edit the `/etc/printcap` file to add the dummy device. `printcap` is the “print capabilities” database. ⁴

```
1 dummy:\
2 :sd=/var/spool/lpd/dummy:\
3 :sh:\
4 :lp=|/usr/share/printconf/util/nofilter:
```

We’ll use this opportunity to explain some basic options of the `printcap` file. `dummy` of course refers to the queue name. `sd` specifies the spool directory where jobs are held as they await printing. `sh` instructs the print system to suppress headers and banner pages. `lp` specifies the device name or file to which the output is sent. For a full explanation of all the `printcap` options, please refer to the `printcap` manual pages.

3.1.2 Create the print filter

Next, we create a null print filter that just echoes whatever it receives on `stdin` straight back to a temporary file. In this case, we edit the `./nofilter` file.

```
#!/bin/sh
exec 1>"/tmp/outfile"
/bin/cat
```

Once the dummy filter is created, we restart the `lpd` server. For example:

```
[root@omega]# /sbin/service lpd restart
Stopping lpd:          [ OK ]
Starting lpd:         [ OK ]
```

3.2 Print a file

The new print queue has been added and is now ready for testing. We’ll pipe some text directly to `lpr` by typing:

```
[root@omega]# fortune | lpr -P dummy
```

If everything worked correctly we should be able to look in `/tmp/outfile` for our text. Sure enough, a peek inside of `/tmp/outfile` shows the output from our `fortune` program:

⁴The location of the `printcap` file varies somewhat by Linux distribution. RedHat 8.0 happens to use `/etc/printcap.local` for user modifications. Other print systems may use something else entirely.

```
[root@omega] cat /tmp/outfile
```

```
The camel died quite suddenly on the second day, and Selena
fretted sullenly and, buffing her already impeccable nails --
not for the first time since the journey began -- pondered
snidely if this would dissolve into a vignette of minor
inconveniences like all the other holidays spent with Basil.
-- Winning sentence, 1983 Bulwer-Lytton bad fiction contest.
```

One is not limited to pure text printing, of course. Part of the print system is a series of filters that not only do the work of converting from one format to another, but also change the appearance of documents. There are a host of other input filters available. Some do nothing more than strip carriage returns and line feeds from the input text. Others will detect C source code and apply line-numbering and syntax highlighting. There are even filters to automatically arrange pages for making folded pamphlets and to print multiple pages per sheet.

Though this was a trivial example, it does serve to show how the print system filters input before sending it to the print device. In a normal print queue, the print system will use a *magic filter* to first determine the input type. Without the magic filter, sending a Postscript file to a Postscript printer may result in the raw source code being printed. Unless one happens to be troubleshooting Postscript itself, this result is unlikely to be desirable.

4 LPRng

LPRng (LPR Next Generation) provides huge improvements over the traditional print systems. It integrates well into heterogeneous networks and is being actively developed. As mentioned, LPRng provides emulation of traditional printing commands so many scripts and utilities will work as expected. The homepage at <http://www.lprng.com/> describes some of its advantages and features:

The LPRng print spooler is an enhanced, extended, and portable implementation of the Berkeley lpr print spooler functionality. While providing the same interface and meeting RFC1179 requirements, the implementation is completely independent and provides support for the following features: lightweight (no databases needed) lpr, lpc, and lprm programs; dynamic redirection of print queues; printer pooling and load balancing; automatic job holding; highly verbose diagnostics; client programs do not need to run SETUID root; greatly enhanced security checks; load balancing across multiple printers; and a greatly improved permission and authorization mechanism.

4.1 LPRng Administration

TOBEWRITTEN

5 CUPS

CUPS, the Common UNIX Printing System, is an ambitious project to provide a common printing environment across different Unix and Unix-like implementations, including Linux. Some of its main features include the ability discover print resources, provide a web-based administration front-end, and support a wide variety of existing and new print systems. From the overview page at <http://www.cups.org> :

CUPS uses IPP/1.1⁵ to provide a complete, modern printing system for UNIX that can be extended to support new printers, devices, and protocols while providing compatibility with existing UNIX applications.

5.1 CUPS Administration

5.1.1 Via the Web Interface

Figure ?? shows the main CUPS administration screen. One useful feature of the CUPS administration page is that it is accessible via any standard HTML browser. Though its configuration files can be edited directly, the browser front end can do all the main administration tasks. To access the administration screen, point your browser to `x.x.x.x:631` , where `x.x.x.x` is the IP address of the machine on which the CUPS server is running.⁶

Clicking on the **Do Administration Tasks** link (Figure ??) will load the module that will allow the addition and deletion of printers, definition of classes, and queue/job management.

From the **Admin** page, click on the **Add Printer** link to define a printer on the system. A screen similar to Figure ?? will appear. Enter the name, location, and description of the printer.

Next, select the type of printing device from the drop-down menu. As of this writing, CUPS supports the following printing protocols:

- AppSocket/Hp JetDirect
- Internet Printing Protocol (http)

⁵The Internet Printing Protocol defines extensions to the HyperText Transport Protocol 1.1 [RFC2616] to provide support for remote printing services. IPP/1.0 was accepted by the IETF as an experimental Request For Comments [RFC] document in October of 1999. Since then the Printer Working Group has developed an updated set of specifications for IPP/1.1 which have been accepted by the IETF and are awaiting publication as proposed standards.

⁶By default, administration requests are limited to the `localhost` interface. Edit the `/etc/cups/cupsd.conf` file to allow other machines to have administration access.



Figure 1: Main CUPS Screen

- Internet Printing Protocol (ipp)
- LPD/LPR Host or Printer
- Windows Printer via Samba

The **AppSocket/HPJetDirect** option is used for many types of network-aware printers. The JetDirect devices, of which compatible devices are sold by different companies, is essentially a remote parallel port. JetDirect devices listen on TCP port 9100, and printing to them is a matter of sending the raw print commands directly to this port. The JetDirects can often support multiple parallel ports and can help eliminate some of the interrupt conflicts when dealing with legacy x86 equipment.

The **Internet Printing Protocol**, as mentioned previously, is a proposed standard for printing systems that will allow, among other things, the ability to print across the Internet. Besides making facsimile devices somewhat obsolete, it could allow easier and better printing networks. The downside is that it is a relatively new standard⁷ and not widely supported.

LPD/LPR were discussed previously where we mentioned that the LPR system is a client/server model. As in most Linux servers, the clients need not be on the same physical machine and can be configured to listen for remote

⁷Remember the phrase, “You’ve got to love standards, there are so many to choose from.”



Figure 2: CUPS Administration Screen

TCP connections. One main downside is the relatively poor authentication mechanisms available for this system.

Finally, shared **Windows printers** are also supported via the Samba ⁸ suite. Though Windows has gained a reputation for its inability to work with non-Windows applications, instability, bloat, anti-trust violations and general evil, it is somewhat ubiquitous in many corporate offices. In fact, there are some non-PC network printer devices (similar to the JetDirects) and even some printers, that advertise themselves on the network as a Windows printer.

For our example we'll choose an HP JetDirect printer. Like CUPS, the JetDirects can be administered via a browser. One can also `telnet` directly into the JetDirect to change passwords, IP information, and other features. Refer to the user manuals for more information.

Next, we are prompted for the Device URI. Because we are printing to a socket on a remote device, we specify the URI as `socket://192.168.1.100:9100`. Of course, substitute the correct IP information for your device. If the JetDirect supports multiple printers they are often accessed by incrementing the port number. Thus, for the second printer on a single JetDirect device we'd use `socket://192.168.1.100:9101`.

Then, select the Model/Driver for the new device on next two screens. The list of supported printers is quite extensive. If however, you do not find your

⁸Samba is a powerful set of utilities to aid Unix and Windows interoperability. It consists of applications for file and print sharing, authentication, name resolution, and service browsing.

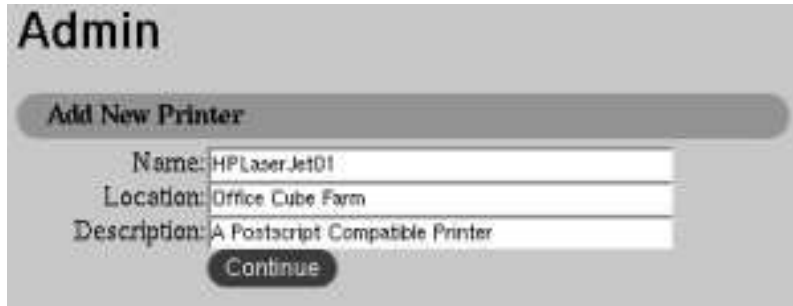


Figure 3: CUPS Add Printer



Figure 4: CAPTION

identical model, try one that seems close. On a related note, use the correct model if you do find it on the list. Though Postscript itself should create pages identically on different printers, there may be slight differences in margins and color/grayscale manipulation that can affect print quality. ⁹

5.1.2 From the Command Line

Like LPRng, CUPS emulates the traditional print administration commands. Either `cancel` or `lprm` can be used to dequeue, or remove, jobs from the spooler. `lpstat` shows the printer status:

```
[klowe@omicon klowe]$ lpstat -p
printer print@omicon unknown state. enabled since Nov 13 21:33 2002. available
Printer: print@omicon
Queue: no printable jobs in queue
Server: no server active
Status: job 'klowe@omicon+62' removed at 16:00:11.980
Rank  Owner/ID                Class Job Files          Size Time
```

⁹On yet another note, you may notice that there are several options for Gimp-Print drivers. The Gimp-Print project's goal is to provide high quality output on inexpensive inkjet printers. They've succeeded. Remarkably. Often, output from Gimp-Print rivals or puts to shame the manufacturer's Windows-only drivers.

5.2 CUPS Print Classes

Another useful feature of CUPS is its ability to create *print pools* which allow jobs to be submitted to a collection of printers and print on the first available one. To define this collection, called a *class* in the CUPS system, choose the **Add Class** option from the Administration menu. Define a name, location, and description for the print class. Next, select the list of printers that belong to the class. When this is completed, the class can be accessed as just another print device and will automatically print to the first available member.

6 Other Front End Programs

We concentrated so far on submitting items to the queue via the `lpr` utility. `lpr`, especially in versions shipped with LPRng and CUPS, allows great control over the output, enabling a user to specify printer type, paper type, multi-page output, banners, etc.. As Linux systems became more and more GUI oriented, several other front-end programs have arisen.

One notable example is the front-end shipped with the KDE desktop, called *kprinter* (Figure ??). For the most part, *kprinter* can be substituted in any instance where one would normally use `lpr`. Instead of having to remember the sometimes cryptic switches, one can use menus and shiny buttons to choose options such as multi-page output or pamphlet printing.



Figure 5: The kprinter main screen

Part IV

Printing from Windows to Linux

Up to this point we've shown ways of printing from Linux clients to other devices. In this part we'll show how to configure a Linux print server to allow Windows clients to have access.

7 Configuring Samba

8 Creating a PDF Printer

This section provides an example of using Linux to create a PDF printer for Windows clients. Though Linux users can easily convert documents into PDF format, Windows users may sometimes need to purchase a commercial package for similar functionality. The following section is based on the informative "PDF Service with Samba" document by John Bright.

Our first goal is to define a print queue within Samba. The main configuration file is located in `/etc/samba/smb.conf` (this may vary according to your distribution). We add the following entries to create a `pdfprinter` resource:

```
[pdfprinter]
  path = /tmp
  printable = yes
  writeable = no
  create mask = 0700
  guest ok = yes
  printer name = PDFPrinter
  print command = /usr/share/pdfconvert
```

As in our LPR/LP example, we see that Samba also allows customization of the printing script and filters. In this case, we define a `pdfconvert` script that will convert the input, in the form of Postscript, into the PDF format using the `ps2pdf` utility. It will then drop the converted document into an accessible part of the filesystem.

```
#!/bin/sh

# Simple script to convert a specified postscript file into a PDF document
# and place it in a location that is shared by the Samba server.
#
# Arguments:
# 1st - The name of the spool file
#
# John Bright, 2001, jbright@winfordeng.com

# We will create the pdf into a temporary file based upon the current
# date and time. After we are finished, we'll rename it to a file with
```

```
# the same date, but ending in .pdf. We do this because if a user tries
# to open a PDF that is still being written, they will get a message
# that it is corrupt, when it is actually just not done yet.
```

```
DATE='date +%b%d-%H%M%S'
```

```
# Directory in which to place the output
# Be sure this directory exists and is writable by the user that Samba
# is running as (for example, the nobody user)
OUTDIR=/shr/pdfdropbox
```

```
ps2pdf $1 $OUTDIR/$DATE.temp
mv $OUTDIR/$DATE.temp $OUTDIR/$DATE.pdf
rm $1
```

Part V

References

References

- [Bright] John Bright, <http://www.linuxgazette.com/issue72/bright.html>, Issue 72 of Linux Gazette, November 2001
- [CUPS] <http://www.cups.org>
- [Ghostscript] *Background information for new users of Ghostscript*, <http://www.cs.wisc.edu/ghost/doc/AFPL/7.04/New-user.htm>
- [LPRng] Patrick A. Powell, *LPRng HOWTO*, LPRng 3.8.15 August 11, 2002
- [IPP] Carl-Uno Manros, *IPP FAQ*, April 2000
- [Samba] The Samba Team, *Samba Project Documentation* , April 1, 2002

Part VI

Miscellaneous Notes

RedHat 8.0 Print problems During the writing of this document I noticed that, no matter what I tried, I could not get xpdf or OpenOffice to print any page that included any sort of graphic. The test pages would print fine, but everything else gave a brief warning message on the printer, then nothing. After a couple hours of troubleshooting, I determined that the xpdf RPM was buggy. I rebuilt the package and enabled the `gzip` library and everything worked as expected. To do this, download the xpdf source RPM then install it. Change to the `/usr/src/rpm/SPECS` directory then edit the `xpdf.spec` file. Look for the following line:

```
%configure --disable-shared --enable-static
```

Add `--gzip` to the end of the line so it reads:

```
%configure --disable-shared --enable-static --with-gzip
```

Rebuild the package using: `rpmbuild --ba xpdf.spec`

Finally, reinstall the new package.