LPRng-HOWTO

Patrick Powell *papowell@astart.com* 21 Dec 1998 (For LPRng-3.5.4)

The LPRng software 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 independant and provides support for the following features: lightweight (no databases needed) lpr, lpc, and lprm programs; dynamic redirection of print queues; automatic job holding; highly verbose diagnostics; multiple printers serving a single queue; 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. The source software compiles and runs on a wide variety of UNIX systems, and is compatible with other print spoolers and network printers that use the LPR interface and meet RFC1179 requirements. The package comes with filters for PostScript and HP printers, as well as the usual 'dumb' printers, and the PostScript and HP filters support page counting and produce accounting information accounting. In addition there are a wide variety of filters for page formatting and to produce banner pages. For users that require compatibility with the SVR4 lp and lpstat programs, the LPRng package provides emulation of these programs. In addition, a publically available PCNFSD server is distributed with LPRng and interfaces with the PC/DOS/Windows based NFS style print spoolers. In addition, LPRng can be easily integrated with the Samba SMB support package. For users that require secure and/or authenticated printing support, LPRng supports Kerberos 5, MIT Kerberos 4 extensions to LPR, and PGP authentication methods. Additional authentication support is extremely simple to add.

Contents

1 Introduction **8**

1.1 What is LPRng? 9

1.2 More Information 9

1.3 Frequently Asked Questions 10

1.4 Copyright 10

1.5 Disclaimer 10

1.6 Commercial Support 10

1.7 Web Site 10

1.8 FTP Sites 11

1.9 Mailing List 11

1.10 PGP Public Key 11

2 Print Spooling Overview **12**

2.1 Overview 12

2.2 Sample Printcap Entry 12

2.3 Print Server Configuration 13

3 Attaching Your Printer **13**

3.1 Network Printers 13

3.1.1 Socket API 14
3.1.2 LPD (RFC 1179) Protcol 14
3.1.3 APPsocket 15
3.2 Network Print Server Boxes 15
3.3 Parallel Printers 16
3.4 Serial Printers 16

4 Installing the Software **17**
4.1 Files and Setup 17
4.2 Source Code and Support Programs 18
4.3 Configuration and Installation 18
4.4 Configuration Options and Problems with Compilation 18
4.5 Advanced Configuration Options 19
4.6 Printcap and lpd.conf files 19
4.7 Security, Permissions, and CHECKPC 20
4.7.1 Using CHECKPC 20
4.7.2 Advanced Security Concerns 22
4.8 Stopping Existing Spooler Software 22
4.8.1 SunOS and BSD Derived 22
4.8.2 Solaris, HP, AIX, and SysVR4 Derived Systems 23
4.9 Initial System Testing 25
4.10 Startup Scripts 26
4.10.1 SunOS and BSD Derived 26
4.10.2 Solaris, Linux, and SysVR4 27
4.11 Replacing UNIX SystemV lp, lpstat Printing Services 27

5 System Specific Notes **28**
5.1 Solaris 2.4, 2.5, 2.6, 2.7 28
5.2 Solaris, Newsprint and FrameMaker 29
5.3 Linux 30
5.4 AIX 31
5.5 Appletalk Support 32
5.6 Apple to LPRng Printing 32
5.7 LPRng to Appletalk Printing 33
5.8 SAMBA Support 34
5.8.1 What is SMB 34
5.8.2 Samba and LPRng 35
5.9 Printer Specific notes 36

5.10 HP Deskjet . . . 36
5.11 HP LaserJet IIISiMX . . . 37
5.12 HP JetDirect Interface . . . 37
5.12.1 Setting Up IP Networking and Address . . . 38
5.12.2 BOOTP Information . . . 38
5.12.3 Paper Tray Selection . . . 42
5.13 Lexmark Printers . . . 42
5.14 Network Base Tektronix Phaser Printers . . . 42
5.15 Duplex Printing . . . 43
5.16 Test Version and Portability Testing . . . 43
5.16.1 Compiling the Test Version . . . 43
5.16.2 Setting Up The Test Version Spool Queues . . . 44
5.16.3 Running the Test Version Software . . . 45

6 The Most Frequently Asked Questions 45
6.1 Why do I get malformed from address errors? . . . 45
6.2 It was working normally, then I get connection refused errors . . . 46
6.3 Job is not in print queue, but it gets printed! . . . 46
6.4 Job disappears and is never printed, but lpr works . . . 48
6.5 I get messages about bad control file format . . . 48
6.6 What is RFC 1179, the Line Printer Daemon Protocol? . . . 48
6.7 I want to replace lp, lpstat, etc, but my programs need them . . . 49
6.8 What are the drawbacks to LPRng? . . . 49

7 Client To Server Connection Information 50
7.1 Printer Name and Server IP Address . . . 50
7.1.1 Command Line -Pprinter@host . . . 50
7.1.2 Command Line -Pprinter . . . 50
7.1.3 PRINTER Environment Variable . . . 51
7.1.4 Default Printer From Printcap . . . 51
7.1.5 Default Printer When No Printcap . . . 51
7.1.6 LPD Default Printer . . . 51
7.2 LPD Server Runnning on Localhost . . . 51

8 Using the Printcap Database 51
8.1 Simple Client Printcap Entry . . . 52
8.2 Simple Server Printcap Example . . . 54
8.3 The Printcap Parsing Rules . . . 55

8.4 Displaying Printcap Information . . . 58
8.5 Remote Printer Using RFC1179 . . . 59
8.6 Remote Printer Using Socket API . . . 59
8.7 Parallel Printer . . . 60
8.8 A serial printer queue . . . 61
8.9 Single Printcap File for Large Installation . . . 61
8.10 Bounce queues . . . 62
8.11 Changing Job Formats . . . 62
8.12 LPR Filters Files . . . 63
8.13 Dynamic Routing . . . 63
8.14 Printer Load Balancing . . . 65
8.14.1 Multi-server print queue . . . 66
8.14.2 Checking Busy Status of Server Queues . . . 67
8.15 Locations of Printcap Files . . . 70
8.15.1 Separate Server and Client Printcap Files . . . 70
8.15.2 ALL Printcap Entry . . . 71
8.16 Management Strategies for Large Installations . . . 71
8.17 Using Programs To Get Printcap Information . . . 71
8.17.1 How to use NIS and LPRng . . . 71
8.17.2 How to use NIS and LPRng - Sven Rudolph . . . 73

9 LPRng checkpc Utility **75**
9.1 Maintenance . . . 75
9.2 Printcap Information . . . 76

10 Spool Queues and Files **76**
10.1 Spool Queue . . . 76
10.2 Queue Lock File . . . 76
10.3 Spool Control File . . . 76
10.4 Log and Status Files . . . 77
10.5 Job Files . . . 78
10.6 Job Hold File . . . 79
10.7 Job State . . . 81
10.8 Job Identifier . . . 81

11 Configuration File lpd.conf and Options **82**
11.1 Configuration File Format . . . 82
11.2 Configuration Information . . . 82

11.3 Configuration Options . . . 83
11.3.1 default_format . . . 83
11.3.2 default_permission=ACCEPT . . . 83
11.3.3 default_priority=A . . . 83
11.3.4 default_remote_host=localhost . . . 83
11.3.5 default_tmp_dir=/tmp . . . 83
11.3.6 domain_name=domain.name . . . 83
11.4 The Record Queue Name qq and force_queuename flags . . . 83
11.5 The check_for_nonprintable Flag . . . 84
11.6 The rg Restrict Use to Group Members Option . . . 84
11.7 The fx Allowed Formats Option . . . 84
11.8 Fixing Bad Control Files and Metacharacters . . . 85
11.9 Using the bk Option and Control File Filters . . . 85
11.10Maximum Copies . . . 86
11.11The minfree Minimum Spool Queue Space Option . . . 86
11.12Debugging . . . 86
11.13LPD Specific . . . 87
11.14Legacy Compatibility . . . 88
11.14.1The allow_duplicate_args Option . . . 88
11.14.2The class_in_status Options . . . 88
11.14.3The reverse_lpq_format Option . . . 88
11.14.4The return_short_status and short_status_length Options . . . 89
11.14.5The force_lpq_status Options . . . 89
11.14.6The ignore_requested_user_priority and force_fqdn_host Options . . . 89
11.14.7The lpr_bsd Options . . . 90

12 Job Processing 90

12.1 Opening the Output Device . . . 90
12.2 Printing Banner At Beginning . . . 93
12.3 Printing Job Files . . . 93
12.4 Printing Banner At End . . . 94
12.5 Normal Termination . . . 95
12.6 Abnormal Termination . . . 95
12.7 LPD Spool Queue Processing . . . 96

13 Filters 97

13.1 What are filters? . . . 97
13.2 What are print formats? . . . 97

13.3 OF Filter . . . 98
13.4 LPR -p format . . . 99
13.5 LPR binary (-l) format . . . 99
13.6 Filter Command Line Flags . . . 99
13.7 LPRng Supported Filters . . . 102
13.7.1 Filter Distribution Conventions . . . 102
13.8 lpf . . . 102
13.9 IFHP Filter . . . 102
13.9.1 Printer Capabilities . . . 103
13.10 Using your own filters . . . 104

14 Permissions and Authentication 104

14.1 Information for matching . . . 104
14.2 Permission Checks . . . 105
14.3 Match Procedure . . . 106
14.3.1 DEFAULT . . . 106
14.3.2 SERVICE . . . 106
14.3.3 USER . . . 107
14.3.4 REMOTEUSER . . . 107
14.3.5 HOST . . . 107
14.3.6 REMOTEHOST . . . 107
14.3.7 PORT . . . 107
14.3.8 IP . . . 107
14.3.9 REMOTEIP . . . 107
14.3.10 LPC . . . 108
14.3.11 SAMEUSER . . . 108
14.3.12 SAMEHOST . . . 108
14.3.13 SERVER . . . 108
14.3.14 FORWARD . . . 108
14.3.15 GROUP . . . 108
14.3.16 REMOTEGROUP . . . 108
14.3.17 CONTROLLINE . . . 109
14.3.18 AUTH . . . 109
14.3.19 AUTHUSER . . . 109
14.3.20 FWDUSER . . . 109
14.3.21 IFIP . . . 109
14.4 Permission File Location . . . 109

14.5 Example Permission File 110
14.6 Complex Permission Checking 110
14.7 More Examples 111

15 Accounting 111

15.1 Printer Accounting Reality Check 111
15.2 How HP Printers Implement Page Counters 114
15.3 Accounting Printcap Options 114
15.4 Accounting File 115
15.5 Remote Server Accounting 115
15.6 Using Filters For Accounting 116
15.7 Accounting Utility accounting.pl 116

16 Authentication and Encryption 117

16.1 Authentication 117
16.2 Identifiers 117
16.3 RFC1179 Protocol Extensions 117
16.4 Client Operations for Client To lpd Server Authentication 118
16.5 Server Operations for Client To lpd Server Authentication 119
16.6 lpd Server to Server Authentication 121
16.7 Permission Checking 122
16.8 Using PGP for Authentication 123
16.8.1 PGP Configuration 124
16.8.2 Permissions 124
16.8.3 Client Configuration 124
16.9 Using Kerberos 5 for Authentication 125
16.9.1 Kerberos Installation Procedure 125
16.9.2 Testing Transfers 126
16.9.3 Explicit Server Principal Name 127
16.10Using Kerberos 4 for Authentication 127

17 Status Monitoring and Logging 127

17.1 LPQ status reporting 127
17.2 LPQ Short Format (lpq -s) 127
17.3 LPQ Long Format (lpq default, lpq -l, lpq -L) 128
17.4 LPQ Verbose Format (lpq -v) 128
17.5 lpc status 129
17.6 Remote Logger Operation 129

17.6.1 Logger Network Communication . . . 129
17.6.2 Logger Messages . . . 130
17.6.3 Message Format . . . 130
17.6.4 Dump Messages . . . 131
17.6.5 LPD Messages . . . 131
17.6.6 Job Status Messages - UPDATE . . . 131
17.6.7 Printer Status Messages - PRSTATUS . . . 132
17.7 LPR -mhost%port and user logging support . . . 132

18 RFC 1179 - Line Printer Daemon Protocol 132
18.1 Ports and Connections . . . 132
18.2 Protocol Requests and Replies . . . 135
18.3 Job Transfer . . . 135
18.4 Data File Transfer . . . 137
18.5 Control File Contents . . . 138
18.6 LPQ Requests . . . 140
18.7 LPRM Requests . . . 140
18.8 LPC Requests . . . 140
18.9 Block Job Transfer . . . 143
18.10Authenticated Transfer . . . 143

19 Defective RFC1179 Implementations 143
19.1 OS2 Print Spoolers . . . 144
19.2 Serious Security Loophole . . . 144

20 Acknowledgements 144

21 Index To All The Configuration and Printcap Options 145

1 Introduction

Printing is one of the essential services provided by computer systems. Users want reliable and easy to use methods of printing that require a minimum amount of effort to used and understand. On single user systems with a directly attached printer they perceive that the printing process is simply a matter of *storing* or *spooling* a file, and then transferring it to the printer in a timely manner. In the classical *multi-user* systems, each user expects to share a common printer with one or more users; the print *spooling* system provides arbitration and sharing of the printer among the various users. In a *network* based multi-user system, there may be one or more printers shared by multiple users on many different systems. The print *spoolers* will need to cooperate to provide print services to the users in a simple an predictible manner.

The LPRng print spooler software was developed to a flexible and scalable approach to each of these printing problems. It has been used since 1988 in extremely demanding academic printing enviroments such as

University of Minnesota, MIT, and Rutgers, commerial companies such as Dow Jones and Abbot Pharmaceuticals, as well as being distributed with Linux, FreeBSD, and other systems. Each of these environments has a unique set of problems, demanding various configuration and administrative capabilities. For example, the simple single user system with a single or limited number of printers requires easy configuration and simple diagnostic procedures, while the network based printing system requires highly robutest error logging, authentication, and failover support.

This document is structured to first provide an overview of the LPRng printing system, and then a *quickstart* guide for single user systems with a simple printer. This includes printer hardware and Operating System specific setup descriptions. Next, more complex configurations are described, and their management and configuration are outlined. Next, an extremely detailed description of the LPRng print spooling components and their functionallity is provided. Finally, an index to the various features and functions of LPRng is provided.

1.1 What is LPRng?

LPRng is a print spooling system. It was designed to mimic the Berkeley (University of California - Berkeley) Line Printers (LPR) package, first found on Berkeley derivatives of the Unix operating system. LPRng will print a document with little or no knowledge of the content or special processing required to print the document on a stand-alone machine or in a distributed printing environment. New (as compared to Berkeley LPR) features include: lightweight `lpr`, `lpc` and `lprm` programs, dynamic redirection of print queues, automatic job holding, highly verbose diagnostics, load balancing queues; enhanced security (SUID not required in most environments), and easy configuration.

LPRng started life at the University of Waterloo in 1986 as PLP (Public Line Printer), a replacement for the original BSD LPD code. This was a one-shot effort by the author, Patrick Powell, to develop code that could be freely redistributed without the restrictions of the BSD/AT&T license and would allow non-license sites to fix and patch problems that they were having with print spooling. From 1988 to 1992, various individuals and groups added features, hacked, slashed, and modified the PLP code, coordinated largely by Justin Mason (<`jmason@iona.ie`>), who started the LPRng mailing list (see 1.2 (More Information)).

In 1992 while at San Diego State University Prof. Powell redesigned and reimplemented the PLP code and named the result LPRng. The goals of the LPRng project were to build a server system that was as close to user abuse proof as possible, that would provide services limited only by the inherent capacities of the support system, RFC1179 compliant, and with extensive debugging capabilities to allow quick and easy diagnostics of problems. Over the period from 1994-1999, LPRng users have suggested extensions, provided patches, and added facilities.

In early 1999 the code base for LPRng was reorganized in order to provide a common method for running on non-UNIX platforms such as Microsoft Windows NT, Apple Rhapsody, and embedded systems.

As a side effect of this work, many security problems that could develop were identified and steps taken to ensure that they were not present in LPRng. For example, LPRng clients such as lpr, lprm, lpc, and lpq run as ordinary users programs, and even the lpd server can run as a user once a network port has been opened. Bounds checking is performed on all input and formatting (for example, snprintf() rather than sprintf() is used), as well as other preventive measures where appropriate.

1.2 More Information

The main LPRng documentation is the LPRng-HOWTO, which is available in text, HTML, info, and PostScript format. Information about LPRng and the latest release can be found on the LPRng web page:

http://www.astart.com/LPRng.html

There is also a mailing list at `lprng@iona.com`. To subscribe, send an email to `lprng-request@iona.com`. The body should contain only the word 'subscribe'. To get off the list later on, repeat the same procedure, but use the word 'unsubscribe'.

Several presentations of LPRng and print spooling software have been made at the Large Scale Installation Administrator (LISA) conferences and are in the LPRng distribution and available on web sites. *LPRng - An Enhanced Printer Spooler System* was presented at the LISA95 conference, and is in the LPRng distribution as LPRng_DOC/LPRng-LISA95.ps. The slides for the *LISA97* and *LISA98* tutorials on *Network Printing and Print Spooling*.

1.3 Frequently Asked Questions

There is a list of Frequently Asked Questions that appear regularly on the LPRng mailing list. For your convenience, see 6 (The Most Frequently Asked Questions).

1.4 Copyright

1.5 Disclaimer

THE MATERIAL IN THIS SOFTWARE PACKAGE IS PROVIDED WITHOUT FEE AND AS-IS WITH NO WARRANTY REGARDING FITNESS OF USE FOR ANY PURPOSE. THE AUTHOR AND ALL CONTRIBUTORS ARE NOT LIABLE FOR ANY DAMAGES, DIRECT OR INDIRECT, RESULTING FROM THE USE OF INFORMATION PROVIDED IN THIS DOCUMENT.

1.6 Commercial Support

AStArt Technologies provides commercial support and enhancements for the LPRng and other network software. AStArt provides network and system consulting services for UNIX and NT systems, as well as real time and network software.

1.7 Web Site

Web Page:

http://www.astart.com/lprng.html

1.8 FTP Sites

The software may be obtained from
ftp://ftp.astart.com/pub/LPRng(Main site)

Mirrors:
ftp://ftp.sage-au.org.au/pub/printing/spooler/lprng (AU)
ftp://sunsite.ualberta.ca/pub/Mirror/LPRng/ (CA)
ftp://ftp.informatik.uni-hamburg.de/pub/os/unix/utils/LPRng (DE)
ftp://ftp.uni-paderborn.de/pub/unix/printer/plp/LPRng (DE)
ftp://ftp.iona.ie/pub/plp/LPRng (IE)
ftp://ftp.chembio.ntnu.no/pub/mirrors/LPRng (NO)
ftp://ftp.mono.org/pub/LPRng (UK)
ftp://ftp.cs.columbia.edu/pub/archives/pkg/LPRng (US)
ftp://ftp.cs.umn.edu/pub/LPRng (US)
ftp://ftp.iona.com/pub/plp/LPRng (US)
ftp://uiarchive.uiuc.edu/pub/packages/LPRng (US)

1.9 Mailing List

To join the LPRng mailing list, please send mail to *lprng-request@lprng.org* with the word 'subscribe' in the BODY.

The LPRng mailing list is archived on
http://www.findmail.com/list/lprng

1.10 PGP Public Key

The LPRng distributions have an MD5 checksum calculated, which is then signed with a PGP public key. Here is the key for validating the checksums:

```
Type Bits/KeyID    Date       User ID
pub  1024/00D95C9D 1997/01/31 Patrick A. Powell <papowell@astart.com>

-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: 2.6.3i

mQCNAzLygTQAAAEEANBW5fPYjN3wSAnP9xWOUc3CvsMUxjip0cN2sY5qrdoJyIhn
qbAspBopR+tGQfyp5T7C21yfWRRnfXmoJ3FVtgToAsJUYmzoSFY08eDx+rmSqCLe
rdJjX8aG8jVXpGipEo9U4QsUK+OKzx3/y/0aK4cizoWqKvy1l4lEzDsA2VydAAUT
tCdQYXRyaWNrIEEuIFBvd2VsbCA8cGFwb3dlbGxAYXN0YXJ0LmNvbT6JAJUDBRA0
XonoiUTMOwDZXJ0BAQ2cBAC7zU9Fn3sC3x0USJ+3vjhg/qA+Gjb5Fi1dJd4solc4
vJvtf0UL/1/rGipbR+A0XHpHzJUMP9ZfJzKZjaK/d0ZBNlS3i+JnypypeQiAqo9t
FVOOyUCwDfWybgAORuAa2V6UJnAhvj/7TpxMmCApolaIb4yFyKunHa8aBxN+17Ro
rrQlUGF0cmljayBBLiBQb3dlbGwgPHBhcG93ZWxsQHNkc3UuZWR1PokAlQMFEDLy
gTSJRMw7ANlcnQEBYBYD/0zTeoiDNnI+NjaIei6+6z6oakq070qFVx0FG3aP3kRH
WlDhdtFaAuaMRh+RItHfFfcHhw5K7jiJdgKiTgGfj5Vt3OdHYkeeh/sddqgf9YnS
tpj0u5NfrotPTUw39n6YTgS5/aW0PQf09dx7jVUcGeod1TGXTe9mIhDMwDJI4J14
=3Zbp
-----END PGP PUBLIC KEY BLOCK-----
```

2 Print Spooling Overview

A print spooler is a program that accepts *print jobs* (which are usually one or more files) from a program or network interface, stores them in a *spool queue*, and then sends them to a printer or another print spooler. Usually there are facilities to submit jobs, check on the current job status, remove jobs from spool queues, and perform administrative functions such as starting or stopping printing.

A print spooler is a client/server application. The client programs are used to submit jobs to the print spooler program which performs the actual printing operations. In order to carry out these operations, the server may need to use other programs to convert print job files into a format acceptible to a printer, or perform various accounting or administrative functions.

2.1 Overview

```
+---------+    +-----+    +-----+     +--------+    +---------+
| program | -> | lpr | -> | lpd |  -> | filter | -> | printer |
+---------+    +-----+  * +-----+     +--------+    +---------+
                  *    *     |
               printcap      V
                          +-----+     +--------+    +---------+
                          | lpd |  -> | filter | -> | printer |
                          +-----+     +--------+    +---------+

                           Figure 1
```

Figure 1 shows the flow of data between the individual components of the LPRng print spooling system. A progam (or user) will use the `lpr` program to send a file to the `lpd` server over a TCP/IP connection. The `lpd`server will store the file temporarily in a spool queue directory. The information needed by the `lpr` and `lpd` programs to carry out this activity is stored in the `printcap` (usually called the `/etc/printcap`) database file.

The `lpd` server sorts the queue entries and determines the print order. It will select a job to be printed, open a connection to the printer, and then use a *filter* program to convert the file contents into a format suiteable for the printer. If the file does not need conversion, then the `lpd` server will send the file directly to the printer.

The `lpd` server can also `forward` jobs to another print server over a network connection, optionally sending them through a filter as well. The destination server can in turn forward the job or send it to a printer.

The protocol or commands used to do this job forward and transfer are specified by 18 (RFC1179). This protocol specifies how the `lpr` client program sends a job to the `lpd` server, as well as how the `lpd` server forwards jobs to another server. In addition to job submission, RFC1179 specifies commands to obtain queue status, to remove jobs from the queue, and to start and stop print queues.

2.2 Sample Printcap Entry

As described in the 2 (Print Spooling Overview), the information in the `printcap` database is used control printing operations. While there is no RFC specifying its format or content, there is a strong *de facto* standard for its format. For a complete description of the `printcap` database see 8 (Using the Printcap Database). For a list of all of the `printcap` and configuration options see 21 (Index To All The Configuration and Printcap Options).

Here is a sample printcap:

```
lp:lp=psqueue@printserver.astart.com
```

This printcap information tells the *client* programs that when a client wants to print a job on the `lp` printer, that these jobs should be sent to the `psqueue` on host `printerserver.astart.com`. On the printserver, the following printcap entry is used by the `lpd` server to do the printing.

```
psqueue:server
  :lp=/dev/lp0
  :sd=/var/spool/lpd/psqueue
  :if=/usr/local/bin/psfilter
```

The `sd` (spool queue directory) entry specifies the directory where print jobs will be placed. The `lp` line specifies an actual output device, so the `lpd` server will open this device for printing. The `if` (input file filter) entry specifies the program to be used to translate the input file into a format compatible with the actual printer. Many of the times there is not need for conversion, and this entry is left out.

2.3 Print Server Configuration

The previous sections have given a very high level view of printing operations. In order to do printing the following programs and information must be established:

1. The printer itself, and the interface to the printer.
2. Client programs for use by users or other programs to send jobs to the print server and perform administrative functions.
3. A server program that runs on a host that accepts jobs for printing.
4. Printcap information to control the printing operations.
5. Filters that convert print jobs into formats compatible with printers.
6. System facilities such as spool queues or storage areas for jobs.

The following sections will cover each of these topics in turn. The initial sections assume that most users are setting up printers on small systems and require an extremely simple print capability. Later sections explore the various configurations that can be used to support large networks of print spoolers as would be found in large academic institutions or businesses.

3 Attaching Your Printer

When installing a printer, the first step should be to make sure that the printer is actually working independently of the software. The following section outlines some simple diagnostics that you can use to check printer connectivity.

3.1 Network Printers

The most flexible and highest throughput printer interface is via a network (TCP/IP) connection. Most high performance printers have a built in network interface, or you can attach them to a *printer server* box which provides a network interface. Network print boxes can be obtained from Hewlett Packard (*http://www.hp.com*), Lantronixs (*http://www.lantronix.com*), and Rose Electronics (*http://www.rosel.com*).

The network interface usually supports multiple network printing protocols. The most common are the LPD (RFC1179), Socket API, APPsocket, SMB, and Novell Netware interfaces. LPRng directly supports the LPD (RFC1179) and Socket API interfaces, and you can use the `smbclient` program from the 5.8 (Samba Software Package) for the SMB interface. There are no plans to support the Novell Netware interace, given the wide range of alternatives.

3.1.1 Socket API

The most powerful network connection is via the the Socket API. The most common Socket API, and a *de facto* standard, is the one used by Hewlett Packard on their JetDirect cards, which allows a TCP/IP connection to port 9100. This is a TCP/IP port on the printer that provides a direct connection to the print engine, similar to a serial or parallel port. This connection is usually full duplex and provides error messages and status information during printing.

The Socket API is extremely simple.

1. The user establishes a connection to port 9100. This connection may be refused if the printer is busy printing a job.
2. Once the connection has been established, the print job is sent over the connection. During this transfer, error messages may be returned over the data link.
3. After sending all the data, the sender should do a *half-close* of the connection. This tells the printer that no more data will be sent.
4. After finishing printing the job, the printer will close the connection.

You can use the *netcat* utility by Hobbit <`Hobbit@avian.org`> to test that this interface is available and working. If *ellipse.ps* is a test file, then: The simplest and easiest way to print a file to a network printer appears

```
  nc printer.ip.addr 9100 < file
Example:
  nc 10.0.0.25 9100 < ellipse.ps
```

3.1.2 LPD (RFC 1179) Protcol

The second most common network print protocol is the RFC1179 TCP/IP protcol. This is described in detail in later sections of this document, but basically consists of a simple set of command and responses.

1. A connection is made to TCP/IP port 515 on the printer. As for the Socket API, this connection may fail if the printer is busy printing another job or has another network connection open.
2. Once the connection has been established, a print request command is sent to the printer, and if printing is allowed, and acknowlegement will be received.
3. The files comprising the print job are sent to the printer.
4. The connection to the printer is closed.

During the job transfer no error information is returned, or other status information. This must be obtained by reconnecting to the printer and sending a print status request. This is a major weakness of the RFC1179 protocol.

When the LPRng software package has been installed, it is easy to check printing by using a command of the form:

```
  lpr -Plp@ipaddr -Dnetwork file
Example:
  lpr -Plp@10.0.0.25 -Dnetwork ellipse.ps
```

The `-Plp@ipaddr` option will cause the `lpr` program to make a direct connection to port 515 and send the print file using the RFC1179 protocol. The `-Dnetwork` debugging option will cause detailed network status to be displayed during this process.

In addition to the `lpr` program, the Perl `cheap_lpr` program can be used. This and other test programs are in the LPRng Distribution `UTILS` directory.

3.1.3 APPsocket

The APPsocket interface is supported by Tektronix and some other printer vendors. It is similar to the Socket API, with a couple of minor differences.

1. The printer has two ports for network connections: TCP port 9100 for TCP/IP stream connections and UDP port 9101 for UDP packet connections.
2. When a 0 length UDP packet or a UDP packet containing only `CR/LF` is sent to UDP port 9101, the printer will return a packet to the sender containing print status information. This information indicates the printers current status (busy, idle, printing) and any error conditions.
3. To send a job to the printer, a connection to port 9100 is made. This connection will be refused while the printer is busy or has a connection to another host.
4. When the TCP connection is established, the job can be sent over the TCP stream. When all of the job has been transferred, the connection should be *shutdown* for sending data by the sender, but remain open to received error messages or other information.
5. An end or job indication in the data stream will also act to terminate the connection. This means that if the PostScript CTRL-D (end of job) character is sent in a job, then the connection will be terminated.
6. Once all the data has been received and the job has finished printing, the connection will be terminated by the printer.

The LPRng IFHP filter program has support for the APPsocket interface. Also, there are Perl or other programs which can communicate with the printer. See 5.14 (Tektronix P450 and Family) for details.

3.2 Network Print Server Boxes

A "network print server" is usually a box (external model) or card in a printer (internal model) which has a network connection to a TCP network and software to implement a LPD print server. If it is an external model, The parallel or serial port of the printer is connected to the box, and the print server may support multiple printers. If it is an internal model, the server is usually nothing more than a Network Interface Controller and a ROM containing software that the microprocessor in the printer uses.

The print server may support multiple printing protocols, such as 6.6 (RFC1179) (TCP/IP printing using the LPD print protocol), Novell Printer Protocols, SMB print protocols, and Appletalk protocols. One of the observed problems with Network Print servers is that while they can usually support one protocol and one user at a time quite well, when you try to use multiple protocols and/or multiple users try to transfer print jobs to the printer, the printer may behave in a very odd manner. Usually this results in a printer failing to finish a job currently being printed, and unable to accept new jobs.

Several of the newer models of print servers have Simple Network Management Protocol (SNMP) agents built into them, and can provide detailed information about their internal functions. By using a SNMP manager such as SunNetmanage or HP-Openview, you can monitor your network printers activities.

I recommend that you use only a single protocol to send jobs to the printer. If you can, I also recommend that you use a print spooler and have only a single host system send a job to the printer.

My best advice on connecting to network printers is not to use the the built-in LPD server, but to use the direct TCP/IP connection to the print engine. Usually this is done to particular TCP/IP port on the printer. For the HP JetDirect and other HP products, this is usually TCP port 9100.

Once you have the direct connection, you can now use various filters to preprocess the print job, insert PJL and PCL commands, or convert text to PostScript or PCL for better print quality.

3.3 Parallel Printers

In most UNIX systems the printer port has the name `/dev/lpt`, `/dev/prn`, or something similar. On most systems the `dmesg` utility will print a list of IO devices found during system configuration. Use the following commands to get the information and scan for the device. You should also make sure that the printer device is available.

```
dmesg >/tmp/a
grep lp /tmp/a
ls /dev/lp*
```

Gordon Haverland <`haverlan@agric.gov.ab.ca`> supplied this little script, that will assist with this:

```
#!/bin/sh
#set -v -x              # uncomment for debugging
PATH=/bin:/usr/bin
printer=
for $printer in /dev/lp* ;
do
    echo PRINTER TEST to $printer 1>&2
    for i in 1 2 3 4 5 6 7 8 9;
    do
        echo PRINTER $printer $i > $printer;
    done
    echo -e \\r\\f > $printer
done
exit 0;
```

If your printer is connected to the device name you provided, then you should get a page of something out. If the output suffers from the “staircase” effect, you will see the numbers “marching” across the page, otherwise the numbers will all be in a single column.

3.4 Serial Printers

If your printer is attached by a serial line, then you may need to set the serial line characteristictics before sending the job to the printer. Here are a set of guidelines to following when attaching a serial port printer to a serial line.

1. Check to make sure that the line is not enabled for login. Logins are usually managed by the `getty` (BSD) or `ttymon` (Solaris, SystemV). Check your system documentation and make sure that these daemons are not managing the serial line.

2. Check the permissions and ownership of the serial line. For the most easy testing, set the permissions to 0666 (everybody can open for reading and writing). After you have made sure that you can send jobs to the printer, you might want to change the ownership of the serial line to the LPD server and change the permissions to 0600.

3. Make sure that you can print a test file on the printer via the serial port. This may require setting the line characteristictics and then sending a file to the printer. You should try to use 8 bit, no parity, with hardware flow control and no special character interpretation, and definitely no LF to CR/LF translation. The problem is that different versions of UNIX systems have different sets of stty(1) commands to do this. The following simple test script can help in this.

```
#!/bin/sh
# 9600, no echo, no CR
FLAGS= 9600 -raw -parenb cs8 crtscts
DEV= /dev/tty01
(stty $FLAGS; stty 1>&2; cat $1 ) <$DEV >$DEV
```

This shows using stty to set the flags, then to print the current settings, and then using cat a file to the output. If you attach a dumb terminal to the serial port, you can even use this script to ensure that input from the device is echoed to the output with the correct speed, parity, etc.

Experience has shown that serially connected printers are the least reliable and lowest speed. Where possible, it is stongly recommended that they be attached to a *network print box* which will provide a Socket API interace and handle the low level network to serial port protocol conversions.

4 Installing the Software

The basic components of the LPRng system are the executables and the database files. This section deals with generating and installing the executable files.

4.1 Files and Setup

The LPRng package consists of:

- *lpd* - the LPD print server program
- *lpr*, *lpq*, *lprm*, and *lpc* - client programs for printing, status queries, job removal, and server configuration respectively.
- */etc/printcap* print queue database file.
- */etc/lpd.conf* program configuration options.
- */etc/lpd.perms* permission information.

By default, the LPRng client programs are installed in `/usr/local/bin` and the `lpd` server in `/usr/local/sbin`. By tradition, UNIX systems have installed printer utilities in in widely varied locations such as `/usr/lib`, `/usr/bin`, `/usr/libexec`, `/usr/ucb`, `/opt`, `/usr/sbin`, and `/usr/etc/`. During installation the old print system files should either be renamed or removed.

The `8 (/etc/printcap)` file contains the printer database information. Information in this file can override the `/etc/lpd.conf` and default configuration information. The `11 (/etc/lpd.conf)` file contains configuration information for the server and client programs. The compiled in defaults should be suitable for most small user installations. The `14 (/etc/lpd.perms)` file contains the permission database information.

4.2 Source Code and Support Programs

1. Obtain the latest version of the LPRng source code from a 1.8 (LPRng FTP Site).
2. Obtain the GNU Zip (compression) program from one of the many *GNU Software Mirror Sites* and install it. See the directions in the GNU Zip distribution for details.
3. Obtain the GNU Make program from one of the many *GNU Software Mirror Sites.* and install it. See the directions in the GNU Zip distribution for details. This program will be referred to as `gmake` in the installation instructions.
4. Obtain an ANSI C compiler. The GCC compiler (from the *GNU Software Mirror Sites*) is strongly recommended.
5. Solaris Sparc and X86 Binaries for GCC and Make can be obtained from *http://smc.vnet.net/*.

4.3 Configuration and Installation

In the following discussion, the LPRng distribution will have the name `LPRng-<version>.tgz`.

Unpack, configure, compile, and install, and initialize the distribution using:

```
gunzip -c LPRng-<version>.tgz | tar xvf -
cd LPRng-<version>
# see discussion below for configuration options
./configure
gmake clean all
su   # you must do the following commands as root
gmake install
# if  you have not installed LPRng before,
# install default lpd.perms and lpd.conf file in /etc
if [ ! -f /etc/lpd.perms ]; then
    make default;
fi;
# update permissions,  create files needed for LPRng, check
# /etc/printcap file for problems.  Do as root:
./src/checkpc -f
```

4.4 Configuration Options and Problems with Compilation

The `configure` script will determine the type of system and establish a set of defaults for compilation and installation. The `LPRng/INSTALL` file contains detailed descriptions of the various configuration options and capabilities.

If you have problems compiling the package, you can try these things:

1. Compiler complains about missing files or has a large number of errors.
   Try `gcc` instead of your vendor's C compiler. This can be done either by setting the `CC` environment variable or using the `-with-cc` configure option.

   ```
   CC=gcc ./configure
      OR
   configure --with-cc=gcc
   ```

2. Missing libraries or include files.
   Usually this is caused when include files are in `/usr/local/include` and libraries are in `/usr/local/include` and these paths are not searched or used by the compiler. This can be fixed by setting the `CPPFLAGS` and `LDFLAGS` environment variables, or using the `-with-cppopts=` and `-with-ldopts=` configure options.

   ```
   CPPFLAGS="-I/usr/local/include -I/usr/include/kerberosIV" \
     LDFLAGS="-L/usr/local/lib -L/usr/lib/kerberosIV" \
     ./configure
      OR
   configure --with-cppopts="-I/usr/local/include -I/usr/include/kerberosIV" \
     --with-ldopts="-L/usr/local/lib -L/usr/lib/kerberosIV"
   ```

3. The default installation locations are not the desired ones.
   See the detailed discussions in the INSTALL file, especially the section dealing with the `-with-lpddir`, `-exec-prefix=`, `-bindir=`, `-sbindir=`, and other basic `configure` options.

The `configure` and `make` steps must be run on the target host, especially if the target host has a different version of the operating system. This is extremely important for SunOS or Solaris, where there tend to be changes in the system's include files between versions as well as support libraries.

Also read the notes for your OS in section 5 (System-dependent notes) for specific installation help (if any).

4.5 Advanced Configuration Options

While the default LPRng configuration will be suitable for most individual users, administrators of large sites or which need to support *lightweight print clients* will need to use the following configure options.

1. `-disable-force_localhost`
   The default LPRng configuration assumes that all printing will be done via a `lpd` print spooler running on the local host system. However, many larger sites prefer that all users do their printing via a few central servers, and do not run `lpd` servers on user systems. The `-disable-force_localhost` configuration will simply this type of operation by eliminating the need for a `/etc/lpd.conf` file to override the `force_localhost` option. Example:

   ```
   ./configure --disable-force_localhost
   ```

2. `-disable-lpd_bounce` The default LPRng configuration assumes that each `lpd` server will process and modify print jobs as they are forwarded from server to server. While this is usually the desired operation on small sites or on individual systems, this is usually not the desired operation on sites running central servers, as will requires distributing filter software to all the user workstations. Example:

   ```
   ./configure --disable-lpd_bounce
   ```

4.6 Printcap and lpd.conf files

The `/etc/printcap` file contains the definitions of print queues and other information used by LPRng. If your system does not have an `/etc/printcap` file, then the following is suitable for initial testing and configuration:

```
# test printcap file
lp:cm=Test Printcap Entry:
 :lp=/dev/null
 :sd=/usr/spool/lpd/lp
```

The `/etc/ifhp.conf` file contains settings that override the defaults provided at compile time. You will find a prototype or template `ifhp.conf` file in the LPRng distribution. You can install this by using:

```
astart > cd LPRng
astart > su
ASTART # make init
           or
ASTART # cp /etc/ifhp.conf /etc
ASTART # chmod 644 /etc/ifhp.conf
```

4.7 Security, Permissions, and CHECKPC

By default, the `lpd` server is run as a ROOT (user 0) process. (This is true not only for LPRng, but also for all other system processes which are started at boot time.) However, normally LPRng will do operations as a non-privileged user and group which is defined by the value of the `user` (default `daemon`) and `group` (default `daemon`) option in the `/etc/lpd.conf` configuration file or the compile time defaults in the `LPRng/src/vars.c` file.

The following steps must be taken in order to preserve system security:

1. Create a user and group `daemon` on the system. This user does not need login privileges, but will need a home directory if secure authentication such as Kerberos or PGP will be done.
2. The `/etc/lpd.conf` file should be owned by root (user 0), and should have read-only (0444) permissions.
3. The `/etc/printcap` file should be owned by root (user 0), and should have read-only (0444) permissions.
4. The spool and working directories used by LPRng should be owned by user `daemon`, group `daemon`, and have 0700 permissions (accessible only by user `daemon`).

4.7.1 Using CHECKPC

The `checkpc` program is used to make sure that the spool directories and files used by LPRng have the correct permissions and are in place. By default, `checkpc` will check permissions and report if there are any problems. You should run this as `root`. For example:

```
% astart > su
#>cd LPRng/src
#>./checkpc
Warning - No configuration file found in '/etc/lpd.conf:/usr/etc/lpd.conf'
Warning - No lpd only printcap file found in '/etc/lpd_printcap,/usr/etc/lpd_printcap'
Warning -  ** cannot open '/var/run/lpd.printer' - 'Permission denied'
Warning -  bad directory - /var/spool/lpd/lp
Warning -   Printer_DYN 'lp' spool dir '/var/spool/lpd/lp' needs fixing
```

In the above example, `checkpc` has discovered that the `/etc/lpd.conf` file is missing. This is not a serious problem if the system defaults are to be used, but you might want to put the default `LPRng/lpd.conf` file from the distribution in place.

The *lpd only printcap* message is usually of concern to administrators who wish to use some of LPRng's more exotic configuration options. It is possible to have separarte printcap databases for client and server programs. This is useful when printcap files get extremely large and cuts down substantially on system management problems.

The permission denied message for `/var/run/lpd.printer` is more serious, as the `lpd` server uses this as a lock file.

The *bad directory* message about the spool directory is usually caused by bad permissions or when the directory is missing.

The `checkpc -f` option causes `checkpc` to take action to rectify errors. You can see what is happening if you run it with the `-V` (vervose) option:

```
% astart > su
#>cd LPRng/src
# ./checkpc -f -V
LPRng version LPRng-3.6.1
 DaemonUID 1, DaemonGID 12
Using Config file '/etc/lpd.conf:/usr/etc/lpd.conf'
Checking for configuration files '/etc/lpd.conf:/usr/etc/lpd.conf'
Warning - No configuration file found in '/etc/lpd.conf:/usr/etc/lpd.conf'
Checking for printcap files '/etc/printcap,/usr/etc/printcap'
  found '/etc/printcap', mod 0100644
Checking for lpd only printcap files
     '/etc/lpd_printcap,/usr/etc/lpd_printcap'
Warning - No lpd only printcap file found in
     '/etc/lpd_printcap,/usr/etc/lpd_printcap'
LPD lockfile '/var/run/lpd.printer'
  checking '/var/run/lpd.printer' file
Names
 :lp=lp
All
 :lp
Printcap Information
lp
 :force_localhost
 :lp=lw4@astart4.astart.com
 :sd=/var/spool/lpd/lp
Checking printcap info
Checking printer 'lp'
 Checking directory: '/var/spool/lpd/lp'
  file 'control.lp', size 0 K, unchanged in 2 hours
  file 'status.lp', size 0 K, unchanged in 2 hours
  file 'status', size 0 K, unchanged in 2 hours
  file 'log', size 0 K, unchanged in 2 hours
  checking 'control.lp' file
  checking 'status.lp' file
  checking 'status' file
  cleaning 'status' file, 0 bytes long: no truncation
  checking 'log' file
  cleaning 'log' file, 0 bytes long: no truncation

# ./checkpc
Warning - No configuration file found in '/etc/lpd.conf:/usr/etc/lpd.conf'
Warning - No lpd only printcap file found in '/etc/lpd_printcap,/usr/etc/lpd_printcap'
```

As you can see, `checkpc` can not only print detailed information about your printing system, but it also fixes up the various problems.

4.7.2 Advanced Security Concerns

While `checkpc` will set permissions, there is always the problem with undetected errors in the LPRng software that, when exploited, could cause severe system problems. The most serious concern is that of gaining root (user 0) permissions.

One way to avoid this is to run *client* programs without root permission. This is the default operation for LPRng, in contrast to other print spooling software.

A remaining problem is the fact that on most UNIX platforms, `lpd` requires root permissions to open port 515. One option is to have the `lpd` server drop root permissions soon after binding to this port and before accepting any user commands. However, in order to be compatible with RFC1179, `lpd` must originate connections from a *reserved* port in the range 721-731, although in practice port 1-1023 seems to be acceptible.

If interoperability with existing print spoolers is not desired, then it is *trivial* to configure LPRng using the `lpd.conf` file or by modifying the compile time `lpd_port` value in the file or the `LPRng/src/vars.c` so that all the software will run as client programs. For example, in the `/etc/lpd.conf` file, you only need to change the indicated lines:

```
# Purpose: lpd port
#   default lpd_port=printer
lpd_port=2000
```

Now all the LPRng software will use port 2000 to transfer jobs and commands. You can also use this facility to establish a *private* set of print spoolers which can be used for testing.

4.8 Stopping Existing Spooler Software

The next step is to shut down and remove the existing print spooler, and test the functionality of the LPRng `lpd` programs. Unfortunately, this process is fairly system dependent, and requires a small amount of system expertise. In addition to these general directions, you should see the 5 (System specific notes) for your system.

4.8.1 SunOS and BSD Derived

This section provides instructions for systems that use the `lpd` print services, such as SunOS, BSD derived systems, and Linux based systems. These systems use an `lpd` print server, as does LPRng.

We first kill the currently running `lpd` process. While there may be a system shutdown script in the `/etc/rc.d/` directory or other location for this, do not bother using it.

```
# most BSD Systems
ps -auxw |grep lpd
# Kill it twice
kill (pid of lpd server)
#  This should report an error - if not then lpd did not terminate
kill (pid of lpd server)
#

Example:
```

```
astart % ps -axuw |grep lpd
papowell 23932  0.0  0.3  224  184  p3  S+  10:40AM  0:00.01 grep lpd
daemon  17763  0.0  0.2  448  120  ??  IWs  29Mar99  0:01.35 (lpd)
astart % kill 135
astart % kill 135
135: No such process
```

Next, you should remove or rename the existing print system executables. The following example shows how to use the `find` utility to track down candidates.

```
astart# find /usr -type f -name lp\*  -print >/tmp/candidates
astart# find /sbin -type f -name lp\*  -print >>/tmp/candidates
astart# cat /tmp/candidates
/usr/bin/lpunlock
/usr/bin/lpqall.faces
/usr/bin/lpq            <---- old
/usr/bin/lpr            <---- old
/usr/bin/lprm           <---- old
/usr/bin/lptest
/usr/doc/samba-1.9.18p10/examples/printer-accounting/lp-acct
/usr/man/man1/lpq.1
/usr/man/man1/lpr.1
/usr/man/man1/lprm.1
/usr/man/man1/lptest.1
/usr/man/man4/lp.4
/usr/man/man8/lpc.8
/usr/man/man8/lpd.8
/usr/sbin/lpc           <--- old
/usr/sbin/lpd           <--- old
/usr/sbin/lpf           <--- old
/usr/local/bin/lpc    <-- LPRng
/usr/local/bin/lpq    <-- LPRng
/usr/local/bin/lpr    <-- LPRng
/usr/local/bin/lprm   <-- LPRng
/usr/local/sbin/lpd   <-- LPRng
astart # mv /usr/bin/lpq  /usr/bin/lpq.old
astart # mv /usr/bin/lpr  /usr/bin/lpr.old
astart # mv /usr/bin/lprm /usr/bin/lprm.old
astart # mv /usr/sbin/lpc /usr/sbin/lpc.old
astart # mv /usr/sbin/lpd /usr/sbin/lpd.old
astart # mv /usr/sbin/lpf /usr/sbin/lpf.old
...
```

4.8.2 Solaris, HP, AIX, and SysVR4 Derived Systems

The original SysVR4 and other related systems did not have any support for RFC1179 network printing (Berkeley LPD). Support for this was added by various manufacture specific methods. Unfortunately, there are a wide range of possibilities.

The `lpsched` process (`/usr/lib/lp/lpsched/`) process performs many of the functions of the LPRng and BSD `lpd` server. On Solaris systems, it also stats the `lpNet` server that provides network print services. Unfortunately, no *simple* and reliable method of shutting down a running `lpsched` process and the associated network services has been found. However, it turns out to be very simple to *prevent* the services from being started.

First, you will need to locate the /etc/rc startup files that start system services. During system startup, a set of shell scripts stored in the /etc/rc.d diretories are executed. The individual startupfile files are usually links to a common one in the /etc/init.d directory. You first need to find the files containing the startup commands. This is done as shown below:

```
SUN # cd /
SUN # grep -l lpsched /etc/rc* /etc/rc*/* init.d/* init.d/*/* >/tmp/files
SUN # cat /tmp/files
/etc/rc0.d/K20lp
/etc/rc2.d/K20lp
/etc/rc2.d/S80lp
/etc/init.d/lp
># ls -l ‘ cat /tmp/files ‘
lrwxrwxr-x 1 root bin 1 Dec 29 23:39 /etc/rc0.d/K20lp -> ../../init.d/lp
lrwxrwxr-x 1 root bin 1 Dec 29 23:39 /etc/rc2.d/K20lp -> ../../init.d/lp
lrwxrwxr-x 1 root bin 1 Dec 29 23:39 /etc/rc2.d/S80lp -> ../../init.d/lp
-rwxr--r-- 5 root sys 460 Sep 1 1998 /etc/rcS.d/K39lp
```

Here is the contents of the typical script file, with the indicated modifications that should be made for testing

```
#!/sbin/sh

#### ADD THE FOLLOWING LINE TO EXIT EARLY
exit 0
#### THE REST IS THE USUAL SCRIPT
case "$1" in
'start')
    [ -f /usr/lib/lpsched ] && /usr/lib/lpsched ;;
'stop' )
    [ -f /usr/lib/lpshut ] && /usr/lib/lpshut ;;
*)
    echo "Usage: $0 { start | stop }"
    exit 1
esac
exit 0
```

Next, as for the BSD installation, we will find all of the printing related commands and rename them. You can either rename them one by one, or use the script method shown below. The minimum of the indicated files should be renamed.

```
SUN # find /usr -type f -name lp\* -print >/etc/printingfiles
SUN # cat /tmp/printingfiles
/usr/bin/lp          <---
/usr/bin/lpstat      <---
/usr/lib/lp/bin/lp.cat
/usr/lib/lp/bin/lp.set
/usr/lib/lp/bin/lp.tell
/usr/lib/lp/lpNet    <---
/usr/lib/lp/lpsched  <---
/usr/lib/lp/lpdata   <---
/usr/sbin/lpadmin    <---
/usr/sbin/lpfilter   <---
/usr/sbin/lpforms    <---
/usr/sbin/lpmove     <---
/usr/sbin/lpshut     <---
```

```
/usr/sbin/lpsystem   <---
/usr/sbin/lpusers    <---
/usr/ucb/lpc         <---
/usr/ucb/lpq         <---
/usr/ucb/lpr         <---
/usr/ucb/lprm        <---
/usr/ucb/lptest
SUN # for i in ‘ cat /tmp/printingfiles ‘ ; do
>  mv $i $i.old
>  done
```

Next, you find if there is a *cron* job scheduled by the file `/var/spool/cron/crontabs/lp` to periodically update and roll over error logs. If there is, you should (after having saved the file) remove it.

```
cp /var/spool/cron/crontabs/lp /etc/cron.crontabs.lp
```

Check the `/etc/inetd.conf` file for a line like:

```
printer stream tcp nowait root /usr/lib/print/in.lpd in.lpd
```

Comment out this line. This line is not present on all systems.

Now we must *reboot* the machine. You can use `reboot` if you are in a rush, and `shutdown` if you are not.

```
SUN # reboot
or
SUN # shutdown -y "Whooga! Whooga! Dive! Dive! System going down."
```

When the system reboots, check to make sure that the `lpd` server is not listening on port 515.

```
SUN # telnet localhost 515
Trying 127.0.0.1...
telnet: Unable to connect to remote host: Connection refused
```

If you do get a connection established then you must use `nlsadmin` to force the *tcpip listener* to release the port, as illustrated below.

```
SUN # nlsadmin -v tcp
lpd  \x00020203000000000000000000000000  ENABLED  \
  NORPC  root  NOMODULES  /var/spool/lp/fifos/listenBSD  #
0  \x00020ACE000000000000000000000000  ENABLED    \
  NORPC  root  NOMODULES  /usr/lib/saf/nlps_server  #
lp  NOADDR  ENABLED  NORPC  root  NOMODULES \
  /var/spool/lp/fifos/listenS5  #
SUN # nlsadmin -r lpd tcp
SUN # nlsadmin -r lp tcp
```

Once you disable this, you should try to reconnect to port 515. If you still cannot, then you have a problem and need to reboot once more.

4.9 Initial System Testing

We will now run the `lpd` executable in the `foreground` and `test` mode, and make sure that our system configuration is correct. It is best to do this with two screens or windows, as you will want to observe the output.

```
# > /usr/local/bin/lpd -F
Fatal error - Another print spooler is using TCP printer port
# > /usr/local/bin/lpd -F -D1
...
1999-04-05-10:02:37.755 astart10 [28903] lpd  Read_file_and_split: \
  cannot open file '/etc/lpd.perms' - No such file or directory
1999-04-05-10:02:37.758 astart10 [28903] lpd  Read_file_and_split: \
  cannot open file '/usr/etc/lpd.perms' - No such file or directory
1999-04-05-10:02:37.759 astart10 [28903] lpd  Build_printcap_info: \
  list->count 0, raw->count 3
1999-04-05-10:02:37.777 astart10 [28903] lpd  lpd: listening socket fd -6
Fatal error - Another print spooler is using TCP printer port
1999-04-05-10:02:37.782 astart10 [28903] lpd  Get_max_fd: getrlimit returns 64
1999-04-05-10:02:37.783 astart10 [28903] lpd  Get_max_fd: returning 64
1999-04-05-10:02:37.786 astart10 [28903] lpd  cleanup: done, doing killpg \
   then exit(0)
```

If you get the above error message, then you have either not killed off other the running `lpd` server or you are not starting the `lpd` server as ROOT. This is the most common error during setup. Correct the problem and then restart the server if neccessary. You should see the output indicated below:

```
# > /usr/local/bin/lpd -F -D1
1999-04-05-14:35:14.023 astart27 [2667] Waiting  lpd: LOOP START
1999-04-05-14:35:14.024 astart27 [2667] Waiting  Get_max_servers: getrlimit returns 256
1999-04-05-14:35:14.024 astart27 [2667] Waiting  Get_max_servers: returning 128
1999-04-05-14:35:14.025 astart27 [2667] Waiting  lpd: max_servers 128, active 0
1999-04-05-14:35:14.025 astart27 [2667] Waiting  lpd: starting select timeout 'yes', 600 sec
```

Now from another window do the following commands:

```
# > lpq -Plp@localhost
Printer: lp@astart
 Queue: no printable jobs in queue
# > lpq
Printer: lp@astart
 Queue: no printable jobs in queue
```

At this point your LPRng software has been installed and tested. You still need to set up *Startup Scripts* to automatically start it at boot time, and `/etc/printcap` entries for your printers.

4.10 Startup Scripts

The purpose of startup scripts is to automatically start the `lpd` print server at boot time. Again, the location and contents of these depend strongly on the version of the Operating System, and system vendor.

4.10.1 SunOS and BSD Derived

In most of these systems the startup script for `lpd` is already present in the `/etc/rc` files and only has to be modified. It can be found by using:

```
ASTART # grep -l lp /etc/rc* /etc/rc*/* /etc/rc*/*/*
/etc/rc
ASTART # more /etc/rc
```

```
...
if [ -f /etc/printcap ]; then
    echo -n ' printer';     /usr/sbin/lpd
fi
```

Modify this file so that path is to the LPRng `lpd` file.

4.10.2 Solaris, Linux, and SysVR4

These systems have individual startup files for each printing service. We need to update the startup files to reference the LPRng executables.

```
SUN # grep -l lp /etc/rc* /etc/rc*/* init.d/* init.d/*/* >/tmp/files
SUN # cat /tmp/files
/etc/rc0.d/K20lp
/etc/rc2.d/K20lp
/etc/rc2.d/S80lp
/etc/init.d/lp
># ls -l ` cat /tmp/files `
lrwxrwxr-x  1 root  bin  1 Dec 29 23:39 /etc/rc0.d/K20lp -> ../../init.d/lp
lrwxrwxr-x  1 root  bin  1 Dec 29 23:39 /etc/rc2.d/K20lp -> ../../init.d/lp
lrwxrwxr-x  1 root  bin  1 Dec 29 23:39 /etc/rc2.d/S80lp -> ../../init.d/lp
-rwxr--r--  5 root  sys  460 Sep 1 1998 /etc/rcS.d/K39lp
```

Modify the startup files so that they use the LPRng `lpd` executable:

```
#!/sbin/sh
case "$1" in
'start')
        [ -f /usr/local/bin/lpd ] && /usr/local/bin/lpd
        ;;
'stop')
    echo "Shutting down lpd: \c"
    kill -2 `cat /var/run/lpd*` >/dev/null 2>1;
        ;;

*)
        echo "Usage: $0 { start | stop }"
        exit 1
esac
exit 0
```

4.11 Replacing UNIX SystemV lp, lpstat Printing Services

Many UNIX utilities in the Solaris and HP UNIX environment use the UNIX System V `lp` and `lpstat` programs. It is almost impossible to modify the programs themselves, as many are *vintage* software that is unsupported or which would be too costly to update.

In order to support these applications, LPRng provides simulation for the `lp`, `lpstat`, and `clean` commands.

The LPRng `lpstat` command is a modified version of the `lpq` command, and accepts the `lpstat` command line options and tries to return status in an `lpstat` format.

If the `lpr` program is invoked with the name `lp`, it will simulate the `lp`options. Finally, if the `lprm` program is invoked with the name `cancel`, it will simulate the `lp`options. This can be done by using symbolic links or copying the programs.

Note that many of the `vintage` applications have fully qualified paths to the `lp` and `lpstat` executables, so it will be nessary to copy them to the original program locations.

```
# original - /usr/bin/lp
# original - /usr/bin/lpstat
cd /usr/local/bin
cp lpr /usr/bin/lp
cp lpstat /usr/bin/lpstat
cp lprm /usr/bin/cancel
```

See the man pages for lp, lpstat, and cancel in the LPRng/man directory. Not all the functions of the original programs are supported and these man pages should be installed to replace the original lp, etc, man pages.

5 System Specific Notes

The following are a set of suggestions and recommendations for specific systems.

5.1 Solaris 2.4, 2.5, 2.6, 2.7

The Sun Solaris operating system is derived from the System V UNIX baseline. Use the following installation procedure.

1. First, install the LPRng software and then rename all of the existing Solaris print facilities. See the 4 (Installation) section for details. You should especially look out for `lp`, `lpstat`, `lpsched`, `lpadmin`, and other executables used by Solaris for print support.
2. You may need to remove the `printer` entry in the `/etc/inetd.conf` file. Make sure it is removed.
3. *Reboot*. Don't try to be fancy and kill off processes, use the *nlsadmin* command, or other insanity. This is brutal, but appears to be necessary in order to ensure that the networking support is set up correctly.
4. When the system restarts, try using `lpq` to check to see if the `lpd` server is active.

James P. Dugal <`jpd@usl.ed`> has also makde the following suggestions as well.

```
From: "Dugal James P." <jpd@usl.edu>
To: lprng@iona.com
Subject: Re: [LPRng] start up trouble

Here are some more tips for Solaris:

1. If /var/spool/cron/crontabs/lp exists, remove it.

In fact, we actually test if /etc/init.d/lp exists on any newly-installed
system, and if so, we issue these commands:
        /etc/init.d/lp stop
        /usr/sbin/pkgrm -n SUNWpsu
        /usr/sbin/pkgrm -n SUNWscplp
```

```
        /usr/sbin/pkgrm -n SUNWpcu
        /usr/sbin/pkgrm -n SUNWpsr
        /usr/sbin/pkgrm -n SUNWpcr
        /bin/rm -f /var/spool/cron/crontabs/lp

Regards,
-- James Dugal, N5KNX           Internet: jpd@usl.edu
Associate Director              Ham packet: n5knx@k5arh.#lft.la.usa.noam
Computing Support Services      US Mail: PO Box 42770  Lafayette, LA  70504
University of Southwestern LA.  Tel. 318-482-6417       U.S.A.
```

5.2 Solaris, Newsprint and FrameMaker

The following is a guide to using LPRng and Sun Microsystems Newsprint by Christopher Hylands, Ptolemy Project Manager of the University of California.

The Sun Newsprint printer is actually an OEM version of the Textronix PhaserII; Sun Microsystems appears to have dropped support for Newsprint, and the recommended migration path is to buy a PostScript printer. If you want more information on using the Newsprint system, notes are available via `http://ptolemy.eecs.berkeley.edu/~cxh/lprng.html`.

Looking through the mailing list logs, it looks like everyone was having a hard time getting lprng to work with Sun's braindead newsprinters. I tried using ghostscript, but the fonts were, IMHO, ugly, so I spent a little time getting the newsprint fonts to work.

The key thing was to grab the file `/usr/newsprint/lpd/if` from a SunOS4.1.3 newsprint installation. If you cannot get this code, then the installation will be extremely difficult.

To install lprng on a Solaris2.x machine, you need to first stop the existing print services and install the startup scripts for LPRng. Note that if there is a local printer, you may have to also fix the permissions of the device. Typical commands are:

```
chown daemon /devices/sbus@1,f8000000/SUNW,lpvi@1,300000:lpvi0
```

We use the following simple `if` script.

```
#/bin/sh
# extremely simple filter script
/bin/cat
```

The Sparcprinters use licensed fonts from NeWSprint. To use the licensed fonts, you must have the lprng spool directory for the sparcprinter in the same location as spool directory of the brain dead Solaris lp system. If your printer is named xsp524, then this directory would be `/etc/lp/printers/xsp524`.

The printcap entry looks like:

```
sp524|524:
    :mx#0:sf:sh:sb:
    :lp=:rm=doppler:rp=xsp524:mx#0:
    :sd=/var/spool/lpd/sp524d:
    :lf=/var/spool/lpd/sp524d/log:
xsp524|Sun SPARCprinter NeWSprint printer:
    :mx#0:sf:sb:sh:rs:
    :lp=/dev/lpvi0:
    :sd=/etc/lp/printers/xsp524:
    :lf=/etc/lp/printers/xsp524/log:
```

```
:af=/var/spool/lpd/xsp524/acct:
:if=/usr/local/lib/newsprint/if:
```

The `/usr/local/lib/newsprint/if` was copied from `/usr/newsprint/lpd/if` in a SunOS4.x installation of the newsprint software. Unfortunately, the newsprint engine is so brain dead that it needs many environment variables set, so it is fairly difficult to come up with a clean script to start the engine. I made the following changes to the file.

1. First, set the path in the script. You may also need to change defaults to suit your preferences:

   ```
   PATH=/usr/ucb:/usr/bin:/etc:/usr/etc:/opt/NeWSprint/bin:/opt/NeWSprint/np/bin:
   PATH=$PATH:$NPHOME/pl.$ARCH/bin:$NPHOME/np/bin; export PATH
   ```

2. You will also need a `/etc/lp/printers/printername/.params` file. If you are using the same spooler directory as the directory that the Solaris lp system uses, then the .param file should appear there. If you are using a different spooler directory, then you will need to copy the .param file from elsewhere and edit it accordingly.

3. If you are going to move a license to a new printer, you should probably save the .param file in the old printer spooler directory. Run /opt/NeWSprint/bin/fp_install and remove the license from the old printer and assign it to the new printer. You could run /opt/NeWSprint/bin/rm_np_printer and remove the printer, but that will get rid of the .param file

4. FrameMaker under Solaris2.x uses the lp command. The fix is to edit $FMHOME/fminit/FMlpr and comment out the lp line and add an lpr line

   ```
   sunxm.s5.sparc)
       lpr -P"$PRINTER" "$FILE"
       #lp -c -d"$PRINTER" "$FILE"
   ```



```
Christopher Hylands, Ptolemy Project Manager  University of California
cxh@eecs.berkeley.edu                 US Mail: 558 Cory Hall #1770
ph: (510)643-9841 fax:(510)642-2739       Berkeley, CA 94720-1770
home: (510)526-4010 (if busy -4068)       (Office: 493 Cory)
```


5.3 Linux

At the time of this writing (Jan 1998), the three major Linux distributions (Slackware, Red Hat and Debian) carry an older version of LPRng. Users of those systems should download the latest stable release, and install that instead of the distributed binaries.

This is mostly important for Slackware 3.2 users, as this version installs LPRng by default. Patrick Volkerding changed the default back to BSD LPR in Slackware 3.3, as many users had experienced problems because they didn't realize they weren't using the BSD software.

Debian's `dselect` utility lets you choose between all packages. Amongst these are LPRng, as well as the traditional LPR software.

You have to make sure your kernel is configured correctly. The documentation for the kernel sources in `/usr/src/linux/Documentation/` and the `Kernel-HOWTO` will help you to generate a new kernel if needed. You will need to set the following options:

- Networking support (`CONFIG_NET`)
- TCP/IP support (`CONFIG_INET`)

- If your printer is connected to a parallel port, you will also need the ‘Parallel Printer Support’ (`CONFIG_PRINTER`). You can use this as a module if you want.
- For a serial printer, answer ‘Yes’ when asked if you want ‘Support for serial devices’ (`CONFIG_SERIAL`). This is also available as a module.

Once you have done this, the current releases of LPRng will install and run without problems. See the 4 (Installing the programs) section for details on how to install LPRng and deactivate the existing print support.

You may need to update the printcap file and filters. See 8 (Using the Princap Database) for details.

5.4 AIX

This information was supplied by *Dirk Nitschke*, as of August 1997, and describes how to install the LPRng package on a workstation running AIX 4.1.x and possibly 3.x.x as well. Dirk would be interested in any comments or corrections.

Printing on AIX systems is different. AIX provides a general queueing facility and printing is only one way to use it. You submit a print job to a print queue using one of the commands `qprt`, `enq`. You can use the BSD or System V printing commands `lpr` or `lp`, too. The qdaemon watches all (general) queues and knows how to handle your job. A (general) queue is defined in the file `/etc/qconfig`. The format of this file is different from the `printcap` format.

OK, how to replace the AIX printing system? There is no group `daemon` on AIX. Therefore you have to change the default group for file ownership and process permissions. We decided to use the `printq` group. The user `daemon` exists on AIX but we have chosen `lpd` as the user who runs `lpd` and all filters and owns the spooling directories. You can change the values for `group` and `user` in your `lpd.conf` file or in the sources `src/common/vars.c`. This is an example for `lpd.conf`:

```
# Purpose: group to run SUID ROOT programs
#   default group=daemon
group=printq
# Purpose: server user for SUID purposes
#   default user=daemon
user=lpd
```

Compile and install the LPRng package. Create your `printcap`, spooling directories, accounting and logfiles and so on. Don’t forget to use 9 (checkpc) to make sure that all the permissions are set correctly and the necessary files are created.

Then stop all print queues defined on your workstation. Use

```
# chque -q queuename -a "up = FALSE"
```

for this (yes, blanks around = are needed).

If you have local printers attached to your system you will have an `lpd` running. Stop this daemon using SMIT (Print Spooling, Manage Print Server, Stop the Print Server Subsystem). Choosing `both` also removes `lpd` from `/etc/inittab`. Maybe it’s faster to do this by hand:

```
# stopsrc -p’pid of /usr/sbin/lpd’
# rmitab "lpd"
```

Now delete all print queues (managed by qdaemon) defined on your system. You can use SMIT for this or the commands {mk,ch,rm}que, {mk,ch,rm}quedev, {mk,ch,rm}virprt. The SMIT fast path is `smit rmpq`.

To start the new `lpd` at system startup you have to add an entry to `/etc/inittab`:

```
# mkitab "lpd:2:once:/full/path/lpd"
```

Some work has to be done if have have a local printer attached to your workstation. You have to create a device file like `/dev/lp0`. The SMIT fast path for this is `smit mkdev`. Choose `Printer/Plotter` and then `Printer/Plotter Devices`. Now `Add a Printer/Plotter`. To create a parallel printer device select the following:

```
Plotter type:              opp Other parallel printer
Printer/Plotter Interface: parallel
Parent Adapter:            ppa0 Available
```

Now define the characteristictics of the device:

```
Port Number: p
```

(`p` is for parallel). Go to the field

```
Send all characters to printer UNMODIFIED   no
```

and select `yes`! We have had a lot of trouble with `no`. This is very important! Expect erroneous output if you choose `no`. If you have already created a device file, change the characteristictics! SMIT's fast path is `smit chdev`.

Finally remove all AIX printing commands like `qprt`, `lp`, `cancel`, `lpr`, `lprm`. You will find a lot of them in `/usr/bin`. Do not remove `enq` and friends if you want to use the general queueing facility.

Now you can start your new `lpd`.

5.5 Appletalk Support

Netatalk is used to communicate from TCP/IP to Appletalk printers and vice versa. The netalk distribution FAQ is at:

http://www.umich.edu/˜rsug/netatalk

There are two issues with using netatalk. The first issue has to do with printing to LPRng-served printers from Macs. The second issue has to do with printing from LPRng to network printers that speak AppleTalk.

5.6 Apple to LPRng Printing

After you have installed and gotten netatalk working, you can use the following configuration file to print from a Macintosh to an LPRng printer.

```
Date: Tue, 30 Sep 1997 00:04:58 -0400 (EDT)
From: Edan Idzerda <edan@mtu.edu>
To: lprng@iona.com
Subject: Re: [LPRng] Netatalk configuration file
> Somebody posted a very nice Netatalk papd.conf file
> that showed the setup for LPRng.  If anybody has this
> handy could you post it?
```

```
Well, *I* use:
Your 32 Character Printer Name:\
        :pr=|/your/path/to/lpr -Pprintername
        :ppd=/your/path/to/ppd/files/yourprinter.ppd
--
Edan Idzerda    <edan@mtu.edu>
System Administrator -- Michigan Technological University, Houghton MI USA
```

5.7 LPRng to Appletalk Printing

The netatalk package comes with a PostScript filter called `psf`. After compilation, it is in (e.g.) `netatalk-1.4b2/etc/psf` and will be installed in (e.g.) `/usr/local/atalk/etc/`. After installation, there will also be a directory `/usr/local/atalk/etc/filters` that contains `ifpap`, `ofpap`, et al. These are just symlinks to `psf`, and `psf` will do the appropriate thing based on how it was invoked. If there's a 'pap' in the name, `psf` uses AppleTalk to talk to the printer. See psf(8) for more information.

A printcap entry for a network printer looks like the following:

```
dave|Dave's 32 Character Printer Name
    :sd=/var/spool/lpd/dave
    :if=/usr/local/atalk/etc/filters/ifpap
    :of=/usr/local/atalk/etc/filters/ofpap
    :lp=/var/spool/lpd/dave/null
```

There are three caveats to using netatalk this way.

1. The first is that `psf` (i.e., the filters) needs to run as root. You can accomplish this in one of two ways. The first requires LPD to run the programs as root:

   ```
   # ROOT_CFLAG=-DROOT_PERMS_TO_FILTER_SECURITY_LOOPHOLE
   ```

   The filter lines then need to look like the following:

   ```
       :if=ROOT /usr/local/atalk/etc/filters/ifpap
       :of=ROOT /usr/local/atalk/etc/filters/ofpap
   ```

   The alternative is to make `psf` SETUID root. To minimize the risk, you may want to make `psf` executable only by group `daemon`. (I haven't tested the first option. I'm currently using the second option.)

2. The second caveat is that each network printer needs a `.paprc` in its spool directory. For instance, `/var/spool/dave/.paprc` looks like this:

   ```
   Dave's 32 Character Printer Name:LaserWriter@Your AppleTalk Zone
   ```

   See psf(8) and pap(1) for more information.

3. The third caveat concerns the lp line in the printcap entry. For a single printer, this can be `/dev/null`. If the host spools to more than one AppleTalk printer, you shouldn't use `/dev/null` for lp. You should use `mknod` to create a null device for each printer. See psf(8).

```
Chad Mynhier <mynhier@cs.utk.edu>
Lab Engineer, CS Department
University of Tennessee, Knoxville
```

5.8 SAMBA Support

From the *http://www.samba.org* Web Site.

5.8.1 What is SMB

This is a big question.

The very short answer is that it is the protocol by which a lot of PC-related machines share files and printers and other information such as lists of available files and printers. Operating systems that support this natively include Windows NT, OS/2, and Linux and add on packages that achieve the same thing are available for DOS, Windows, VMS, Unix of all kinds, MVS, and more. There is no reason why Apple Macs and indeed any Web browser should not be able to speak this protocol, and current development (in which the Samba team is heavily involved) is aimed at exactly that. Alternatives to SMB include Netware, NFS, Appletalk, Banyan Vines, Decnet etc; many of these have advantages but none are both public specifications and widely implemented in desktop machines by default.

The Common Internet Filesystem is what the new SMB initiative is called. For details watch *http://www.samba.org*.

WHAT CAN SAMBA DO?

Here is a very short list of what samba includes, and what it does.

- a SMB server, to provide Windows NT and LAN Manager-style file and print services to SMB clients such as Windows 95, Warp Server, smbfs and others.
- a Netbios (rfc1001/1002) nameserver, which among other things gives browsing support. Samba can be the master browser on your LAN if you wish.
- a ftp-like SMB client so you can access PC resources (disks and printers) from unix, Netware and other operating systems
- a tar extension to the client for backing up PCs

For a much better overview have a look at the web site at *http://www.samba.org* and browse the user survey.

Related packages include:

- smbfs, a Linux-only filesystem allowing you to mount remote SMB filesystems from PCs on your Linux box. This is included as standard with Linux 2.0 and later.
- tcpdump-smb, a extension to tcpdump to allow you to investigate SMB networking problems over netbeui and tcp/ip.
- smblib, a library of smb functions which are designed to make it easy to smb-ise any particular application.

FTP SITE and WEB SITE

The main anonymous ftp and web site for this software is found using *http://www.samba.org*.

As well as general information and documentation, this also has searchable archives of the mailing list and a user survey that shows who else is using this package. Have you registered with the survey yet? :-)

5.8.2 Samba and LPRng

The SAMBA code is very easy to configure. See the SAMBA documentation for details, but you only need to modify the samba.conf file and put in the pathnames of the LPRng facilities. The following is a sample.

```
From: Sascha Ottolski <alzhimer@cs.tu-berlin.de>
Subject: Re: [LPRng] lprng-3.2.6 and smb on Linux
webnut@conc.tds.net said:

 I have samba sending print from Win95 machines to LPRng.  The key to
 making it work is in the samba.conf file in the [global] section:

[global]
    printing = lprng
    print command = /usr/local/bin/lpr  -P%p %s -r
    lpq command   = /usr/local/bin/lpq  -P%p
    lprm command  = /usr/local/bin/lprm -P%p %j
    printcap name = /etc/printcap
    load printers = no
[printers]
   comment = All Printers
   path = /tmp
   browseable = no
   printable = yes
   guest ok = no
   writable = no
   create mode = 0700

Note: the path= value specifies the spool directory where
the print files are temporarily stored.  This should NOT
be the LPRng spool directory,  but some other directory that
Samba has write permissions for.

From: "Pascal A. Dupuis" <dupuis@lei.ucl.ac.be>
Subject: Re: LPRng-3.2.10 and SAMBA
```

I include the smbprint script used to send stdin to a NetBEUI printer.

```
#!/bin/sh -x
# This script is an input filter for printcap printing on a unix machine. It
# uses the smbclient program to print the file to the specified smb-based
# server and service.
# For example you could have a printcap entry like this
#
# smb:lp=/dev/null:sd=/usr/spool/smb:sh:if=/usr/local/samba/smbprint
#
# which would create a unix printer called "smb" that will print via this
# script. You will need to create the spool directory /usr/spool/smb with
# appropriate permissions and ownerships for your system.
#
# The /usr/spool/smb/.config file should contain:
#   server=PC_SERVER
#   service=PR_SHARENAME
#   password="password"
#
```

```
# Set these to the server and service you wish to print to
# In this example I have a WfWg PC called "lapland" that has a printer
# exported called "printer" with no password.
#
# E.g.
#   server=PAULS_PC
#   service=CJET_371
#   password=""
# Should read the following variables set in the config file:
#   server, service, password
config_file=.config
eval `cat $config_file`
# echo "server $server, service $service" 2>&1
(
# NOTE You may wish to add the line `echo translate' if you want automatic
# CR/LF translation when printing.
#       echo translate
    echo "print -"
    cat
) | /usr/local/bin/smbclient "\\\\$server\\$service" \
   $password -U $server -N -P
# comment preceeding line and uncomment following
# to just test for correct filter working
#) | cat > /dev/null
```

5.9 Printer Specific notes

This is a small collection of miscellaneous notes about printers and applications.

5.10 HP Deskjet

```
From: jarausch@igpm.rwth-aachen.de (Helmut Jarausch)
Subject: Re: Using gs (ghostscript) as a filter?
To: lprng@iona.com
Cc: Rick Gaine <rgaine@nbcs.rutgers.edu>
Sender: majordomo-owner@iona.com
Reply-To: lprng@iona.com
>>
>> Hello All:
>>
>> I would like to use LPRng 3.1.4 with an HP LaserJet 4P.  I'd like to be
>> able to use gs to convert PostScript files so that I can print them on my
>> HP 4P.  Can I do this with LPRng?  If so, could someone semd me a printcap
>> entry?  I'd appreciate it.  I am not sure how I will be cconnecting the
>> printer yet, but I am thinking either serial or network.  Probably serial
>> though.  Thanks for any help.
```

This printcap works for my Deskjet:

```
djps
    :cm=Local Deskjet(GhostScript)
    :sd=/var/spool/djps:sf:sh:mx#0
    :lp=/dev/plp
    :if=/usr/local/bin/LPRng/ps_to_deskjet:
```

and this is the script /usr/local/bin/LPRng/ps_to_deskjet

```
#!/bin/sh
nice -19 /usr/local/bin/gs -sDEVICE=cdj550 -sPAPERSIZE=a4 \
    -sOutputFile=- -q -r300 -
```

```
Helmut Jarausch
Lehrstuhl f. Numerische Mathematik
Institute of Technology
RWTH Aachen
D 52056 Aachen, Germany
```

5.11 HP LaserJet IIISiMX

```
> From majordomo-owner@iona.com Mon Aug 31 11:17:26 1998
> To: lprng@lprng.org
> Subject: [LPRng] problems printing PS-level2 jobs on LJIIIsi's...
> Date: Mon, 31 Aug 1998 15:06:22 -0400
> From: "John Saroglou" <johny@yorku.ca>
>
> Greetings...
>
> I'm wondering if someone got around the problem of printing
> Postscript(R) Level 2  jobs on Laser Jet IIIsi printers.
> Our printers are direct network printers talking to a
> print server running solaris 2.6 and lprng-3.5.1.
>
> Is there a fix (possible drivers?) for such problem?
>
> Thanks in advance.
```

The LaserJet IIISi does not support PostScript level 2, only level 1 (really, it is called 3SiMX). The Windows (you are under Windows, right?) HP driver for 3Si/3Si MX PostScript should produce only PS level 1.

Beware: latest version of Adobe Windows PS driver produces *ONLY* PS level 2. So if you have (or receive) level 2 files, read them using ghostscript and print a screen dump :-) or as a bitmap.

You can also convert them into PDF (using either Adobe distiller or ghostscript ps2pdf) then use *acroread* to print the result. *Acroread* can produce either level 1 or level 2 PostScript. THE definite solution!

This trouble has nothing to do with the way they are connected, it is only a driver problem. I believe that HP had once a PS level2 update, but the price was so high that buying a new printer was a better solution!

```
Bertrand
--
| Bertrand DECOUTY              | mailto:Bertrand.Decouty@irisa.fr   |
| IRISA - INRIA (Atelier)       | PHONE : 0299847346 / 0299847100    |
| Campus de Beaulieu            | FAX   : +33 (0) 299842534          |
| F-35042 Rennes Cedex - FRANCE | http://www.irisa.fr/               |
```

5.12 HP JetDirect Interface

The HPJetDirect card can be configured through the front panel or through a set of network files. Here is a summary of the methods used from UNIX systems, or when you are desperate, to configure the printer.

5.12.1 Setting Up IP Networking and Address

You can set the network address from the front panel. Reset the printer, put it in offline mode. and then use the MENU, +-, SELECT keys as follows:

```
MENU  -> MIO MENU (use MENU to display MIO MENU)
ITEM  -> CFG NETWORK=NO*
+     -> CFG NETWORK=YES
ENTER -> CFG NETWORK=YES*
ITEM  -> TCP/IP=OFF* (use ITEM to display TCP/IP)
+     -> TCP/IP=ON
ENTER -> TCP/IP=ON*
ITEM  -> CFG TCP/IP=NO* (use ITEM to display TCP/IP)
+     -> CFG TCP/IP=YES
ENTER -> CFG TCP/IP=YES*
ITEM  -> BOOTP=NO*
    (Enable BOOTP if you want to - see below)
ITEM  -> IP BYTE 1=0*
    This is IP address MSB byte.
    Use +- keys to change value, and then ENTER to change
    Use ITEM keys to get IP BYTE=2,3,4
ITEM  -> SM BYTE 1=255*
     This is the subnet mask value
    Use +- keys to change value, and then ENTER to change
    Use ITEM keys to get IP BYTE=2,3,4
ITEM  -> LG BYTE 1=255*
    This is the Syslog server (LoGger) IP address
    Use +- keys to change value, and then ENTER to change
    Use ITEM keys to get IP BYTE=2,3,4
ITEM  -> GW BYTE 1=255*
    This is the subnet gateway (router) IP address
    Use +- keys to change value, and then ENTER to change
    Use ITEM keys to get IP BYTE=2,3,4
ITEM  -> TIMEOUT=90
     This is the connection timeout value.  It puts a limit
    on time between connections.  A value of 10 is reasonable.
```

5.12.2 BOOTP Information

If you have a bootp server, you can put this information in the bootptab file. To use this, you must enable the bootp option on the printer. The T144 option specifies a file to be read from the bootp server. This file is read by using the TFTP protocol, and you must have a TFTPD server enabled. Here is a sample bootptab entry.

```
# Example /etc/bootptab: database for bootp server (/etc/bootpd).
# Blank lines and lines beginning with '#' are ignored.
#
# Legend:
#
#       first field -- hostname
#                       (may be full domain name)
#
#       hd -- home directory
#       bf -- bootfile
```

```
#       cs -- cookie servers
#       ds -- domain name servers
#       gw -- gateways
#       ha -- hardware address
#       ht -- hardware type
#       im -- impress servers
#       ip -- host IP address
#       lg -- log servers
#       lp -- LPR servers
#       ns -- IEN-116 name servers
#       rl -- resource location protocol servers
#       sm -- subnet mask
#       tc -- template host (points to similar host entry)
#       to -- time offset (seconds)
#       ts -- time servers
#
# Be careful about including backslashes where they're needed.  Weird (bad)
# things can happen when a backslash is omitted where one is intended.
#
peripheral1:
:hn:ht=ether:vm=rfc1048:
:ha=08000903212F:
:ip=190.40.101.22:
:sm=255.255.255.0:
:gw=190.40.101.1:
:lg=190.40.101.3:
:T144="hpnp/peripheral1.cfg":
```

If you are using the T144 option, you will need to create the configuration file. The sample configuration file from the HP Direct distribution is included below.

```
#
# Example HP Network Peripheral Interface configuration file
#
# Comments begin with '#' and end at the end of the line.
# Blank lines are ignored.  Entries cannot span lines.

# Name is the peripheral (or node) name.  It is displayed on the peripheral's
# self-test page or configuration plot, and when sysName is obtained through
# SNMP.  This name can be provided in the BOOTP response or can be specified
# in the NPI configuration file to prevent the BOOTP response from overflowing
# the packet.  The domain portion of the name is not necessary because the
# peripheral does not perform Domain Name System (DNS) searches.  Name is
# limited to 64 characters.

name: picasso

# Location describes the physical location of the peripheral.  This is the
# value used by the interface for the MIB-II sysLocation object.  The default
# location is undefined.  Only printable ASCII characters are allowed.
# Maximum length is 64 characters.

location: 1st floor, south wall

# Contact is the name of the person who administers or services the peripheral
```

```
# and may include how to contact this person.  It is limited to 64 characters.
# This is the value used by the interface for the MIB-II sysContact object.
# The default contact is undefined.  Only printable ASCII characters are
# allowed.  Maximum length is 64 characters.

contact: Phil, ext 1234

# The host access list contains the list of hosts or networks of hosts
# that are allowed to connect to the peripheral.  The format is
# "allow: netnum [mask]", where netnum is a network number or a host IP
# address.  Mask is an address mask of bits to apply to the network number
# and connecting host's IP address to verify access to the peripheral.
# The mask usually matches the network or subnet mask, but this is not
# required.  If netnum is a host IP address, the mask 255.255.255.255 can
# be omitted.  Up to ten access list entries are permitted.

# to allow all of network 10 to access the peripheral:
allow: 10.0.0.0  255.0.0.0

# to allow a single host without specifying the mask:
allow: 15.1.2.3

# Idle timeout is the time (in seconds) after which an idle
# print data connection is closed.  A value of zero disables
# the timeout mechanism.  The default timeout is 90 seconds.

idle-timeout: 120

# A community name is a password that allows SNMP access to MIB values on
# the network peripheral.  Community names are not highly secure; they are
# not encrypted across the network.  The get community name determines which
# SNMP GetRequests are responded to.  By default, the network peripheral
# responds to all GetRequests.  The get community name is limited to 32
# characters.
#
# For hpnpstat and hpnpadmin, the community name can be stored in
# /usr/lib/hpnp/hpnpsnmp.

get-community-name: blue

# The set community name is similar to the get community name.  The set
# community name determines which SNMP SetRequests are responded to.  In
# addition, SetRequests are only honored if the sending host is on the
# host access list.  By default, the network peripheral does not respond
# to any SetRequests.  The set community name is limited to 32 characters.
#
# The set community name can come from /usr/lib/hpnp/hpnpsnmp
# if it is the same as the get community name.  We recommend that the
# set community name be different from the get community name though.

set-community-name: yellow

# SNMP traps are asynchronous notifications of some event that has occurred.
# SNMP traps are useful only with network management software.  Traps are
# sent to specific hosts and include a trap community name.  Up to four
```

```
# hosts can be sent SNMP traps.   The trap community name is limited to
# 32 characters.  The default name is public.

trap-community-name: red

# The SNMP trap destination list specifies systems to which SNMP
# traps are sent.  Up to four IP addresses are allowed.  If no
# trap destinations are listed, traps are not sent.

trap-dest: 15.1.2.3
trap-dest: 15.2.3.4

# The SNMP authentication trap parameter enables or disables the sending
# of SNMP authentication traps.  Authentication traps indicate that an SNMP
# request was received and the community name check failed.  By default,
# the parameter is off.

authentication-trap: on

# The syslog-facility parameter sets the source facility identifier that the
# card uses when issuing syslog messages.  Other facilities, for example,
# include the kernel (LOG_KERN), the mail system (LOG_MAIL), and the spooling
# system (LOG_LPR).  The card only allows its syslog facility to be configured
# to one of the local user values (LOG_LOCAL0 through LOG_LOCAL7).  The
# selectable option strings, local0 through local7 (configured to LOG_LOCAL0
# through LOG_LOCAL7, respectively) are case insensitive.  The default
# syslog-facility for the card is LOG_LPR.

syslog-facility: local2

# This parameter allows the card to treat hosts on other subnets as if the
# hosts were on the card's subnet.  This parameter determines the TCP
# Maximum Segment Size (MSS) advertised by the card to hosts on other subnets
# and affects the card's initial receive-window size.  The card will use a
# TCP MSS of 1460 bytes for local hosts, and 536 bytes for a non-local host.
# The default is off, that is, the card will use the maximum packet sizes
# only on the card's configured subnet.
#
# The configuration utility does not allow access to this parameter.  If you
# want to configure it, you must manually edit the NPI configuration file
# and add it to the bottom of the entry for the network peripheral.

subnets-local: on

# This parameter affects how the card handles TCP connection requests from
# the host.  By default, the JetDirect MPS card will accept a TCP connection
# even if the peripheral is off-line.  If this parameter is set to "on", then
# the card will only accept a TCP connection when the peripheral is on-line.

old-idle-mode: off
```

5.12.3 Paper Tray Selection

Be careful with your paper tray selection. You should configure the printer, using the front panel switches, to select the FIRST paper tray. See your printer documentation on this. Unfortunately, different models of HP printers have different methods of handling paper trays.

5.13 Lexmark Printers

Some Lexmark printers do not send *end of job* status back unless configured to do so. Here is what is needed to force this.

```
Date: Wed, 21 Jan 1998 18:25:50 -0600 (CST)
From: Matt White <whitem@bofh.usask.ca>
To: lprng@iona.com
Subject: Re: [LPRng] ifhp with Lexmark Optra N printer

On Wed, 21 Jan 1998, Simon Greaves wrote:

> Apologies in advance if this is way off mark, but we've been evaluating a
> commercial print charging package (Geomica) which works by talking to the
> printer in what I think is a similar way to the ifhp filters. Lexmarks are
> currently a big headache because they seem to fail to return the message
> that they have finished printing which screws things up somewhat. In our
> case, it is believed to be a problem with the Lexmark firmware which they
> are looking into.

There is a fix for that...it is originally from the Lexmark 4039 series,
but it still works on the Optra S 1650 machines that we have (and should
work on the rest of the optra line).  Just send this little chunk of
postscript to the printer once:

-----------snip----------
%! Postscript utility file to set the 4039 printer into synchronous mode
serverdict begin 0 exitserver
statusdict begin true setenginesync end
-----------snip----------

Basically, it causes the printer to wait until it is finished printing
before actually reporting that it is done.  I've got 3 Optra S printers
running with ifhp right now with no extra options (just defaults).

--------------------------------------------------------------------
- Matt White                         whitem@arts.usask.ca          -
- Network Technical Support          http://arts.usask.ca/~whitem   -
- College of Arts & Science          University of Saskatchewan     -
--------------------------------------------------------------------
```

5.14 Network Base Tektronix Phaser Printers

The `ifhp` filter supports the 3.1.3 (APPsocket) protocol used by Tektronix. You will need a printcap similar to the following:

```
phaser:
```

```
:sd=/var/spool/lpd/%P
# need a dummy device for output
:lp=/dev/null
# You need to specify the IP address of the printer's network interface
#  In this example it is 10.0.0.1 - replace with the correct value
# The filter will actually open the connection.
:if=/usr/local/lib/filter/ifhp -Tdev=10.0.0.1%9100,model=tek
```

5.15 Duplex Printing

Duplex printing is when you print on both sides of a page. Some printers which do duplex printing require that you send them special commands to force this mode. This is usually done by the FILTERS. The IFHP filter makes a stab at sending the PJL or PostScript commands to the printer. Many people have reported problems doing duplex printing, so here is a check list.

1. Make sure you have enough memory for the worst case print job. Usually the printer has to rasterize both pages before it can produce an impression. It may require much more memory than you expect.
2. Check your printer manual to discover the EXACT form of the `enter duplex mode` command and make sure that either the command is part of the job (PJL language at the start of the job, postscript header, etc), or that the filter generates the correct form. Note there is a PostScript Printer Description file (PPD) for most printers that support PostScript, and they even have the PJL and PostScript code for this in the PPD file.
3. It has been observed that even with what would apparently be sufficient memory, that many duplex jobs print 'oddly', that they are not aligned on the same side in the same way, etc etc. This may not be the fault of the software, but of the support for duplex operation.
4. Read the IFHP documentation, and create a configuration section in the `ifhp.conf` file for your printer.

I know this is painful, but until there is a uniform way to get the correct commands extracted from either PPD or some other database then this appears to be the only way to do it.

Patrick Powell

5.16 Test Version and Portability Testing

The LPRng code has the ability to run as non-setuid software, and to use the non-default TCP/IP ports for communication. This facility allows a *Test Version* to be run in parallel with the normal LPRng software.

To simplify testing and portability issues, a simple test version of the spool queues and jobs has been supplied with the LPRng distribution. These queues can be placed in a suitable location (`/tmp` is common) and the LPRng software tested.

The test version of the software will use the `LPD_CONF` environment variable to specify the location of the configuration file. It will read this configuration file on startup and use the values to override the normal defaults. Since a user could maliciously set up their own configuration files with values that could compromise system security, it is strongly recommended that the test version is not made SETUID root. In fact, the LPRng code will chatter messages when the LPD_CONF ability is enabled and it is run as root.

5.16.1 Compiling the Test Version

Edit `src/Makefile`, and uncomment the indicated line. Then run `make` to regenerate the distribution.

```
#### ****** TESTING AND SECURITY LOOPHOLE ******************************
# Define GETENV to allow the LPD_CONFIG environment
#  variable to be used as the name of a configuration file.  In non-testing
#  systems,  this is a security loophole.
#CF := $(CF) -DGETENV
```

5.16.2 Setting Up The Test Version Spool Queues

The LPRng TESTSUPPORT directory contains a set of shell scripts and files that need to be installed in the appropriate directory. The following steps are used.

1. First, you need to set up your HOST environment variable to the fully qualified domain name of your host and your USER environment variable to your user name. This is done in order to get values to put into the Test Version configuration files.
2. In the TESTSUPPORT directory, edit the `Makefile`, and specify the location of the `Test Version` spool queues. The default location is `/tmp`; since on most systems these files are deleted or are available to everybody, a more secure location should most likely be used. **DO NOT USE THE RAW TESTFILE DIRECTORY**. These files need to be copied and placed in another directory.
3. The LPD_CONF environment variable should be set to the location of the installed `lpd.conf` file.
4. In the TESTSUPPORT directory, run `make`. This will copy and install the necessary files.

Example:

```
CSH:
  setenv HOST {fully qualified domain name};
  setenv USER ‘whoami‘
  setenv LPD_CONF /tmp/LPD/lpd.conf
  set path=( /tmp/LPD $path )
  unsetenv PRINTER
 Example:
    setenv HOST astart1.astart.com
    setenv USER papowell
    setenv LPD_CONF /tmp/LPD/lpd.conf
    set path=( /tmp/LPD $path )
    unsetenv PRINTER
Bourne Shell:
  HOST={fully qualified domain name}; export HOST;
  USER='whoami'; export USER
  LPD_CONF=/tmp/LPD/lpd.conf.$HOST; export LPD_CONF
  PATH=/tmp/LPD:$PATH; export PATH
  PRINTER=; export PRINTER
 Example:
    HOST=astart1.astart.com; export HOST
    USER=papowell; export USER
    LPD_CONF=/tmp/LPD/lpd.conf.$HOST; export LPD_CONF
    PATH=/tmp/LPD:$PATH; export PATH
    PRINTER=; export PRINTER
cd TESTSUPPORT
make
```

5.16.3 Running the Test Version Software

Set your current directory to the location of the compiled `Test Version` executables. Execute the various executables using `./cmd`, or set **. as the first entry in the PATH** . If it is not the first entry, then the standard system executables will be used.

1. Run `./checkpc`. this will print out the various values for the spool queues in the `Test Version` setup. If the `t1`, `t2`,... spool queues are not displayed, make sure that the LPD_CONF environment variable is set correctly and that you are using the `Test Version` executable.
2. Run `./checkpc -f`. This will fix up the (deliberately introduced) problems in the spool queues.
3. Next, run `./lpd -F` in one window, and then run `./lpq -a` in another window. This will check that the server is working.
4. You can now amuse yourself by sending jobs, setting up permissions checking, and other chores.
5. When everything appears to be working correctly, you can then remove the `Test Version` flag from the `src/Makefile`, recompile, and install the LPRng software.

6 The Most Frequently Asked Questions

In this section, the Most Frequently Asked Questions have been placed, together with their answers. You may notice that some questions have the same answer, but the symptoms appear differently.

Some of these answers will reference other material in this FAQ, or the LPRng man pages.

6.1 Why do I get malformed from address errors?

This is the number one question asked by most LPRng users who try to use LPRng with network printers or other systems supporting 6.6 (RFC1179) printing. For details about LPRng and RFC1179, see 18 (RFC1179 and LPRng).

The `malformed from address` error is usually reported when trying to send a print job from LPRng to other BSD LPR or RFC1179 LPR implementations, or with network connected printers that have a built in LPR server. This is due to the following RFC1179 rule:

> Servers originate a connection from ports in the range 721-731.

WHY? These are a subset of the 'reserved' ports in UNIX, and normal users cannot open connections from them. This provides a small amount of security from UNIX users on the host 'spoofing' a server.

IMPLICATION: in order to do use a reserved port, the program must have root privileges. This means the LPR, LPD, LPQ, etc., programs must be installed SUID root. This can open up a can of worms with regard to security, but LPRng has been designed to take as much paranoid care as possible to avoid problems.

WHAT TO DO:
When installing LPRng, you will need to install the executables SUID root. In the `src/Makefile`, you can remove the comment from the line

```
PERMS=SUID_ROOT_PERMS
```

and then do `make install`. This will install the executables SUID, and owned by root.

6.2 It was working normally, then I get connection refused errors

This message usually appears when you have been sending a large number of jobs to a network printer or a remote system. The reason for this is a combination the above port 721-731 restriction and the TCP/IP timeouts. For details, see 18 (RFC1179 and LPRng), but here is a quick explanation.

A TCP/IP connection is usually specified as between `srchost:srcport, desthost:destport`, although in practice the order of source (src) and destination (dest) is not important.

When a connection is established, each end of the connection exchanges the necessary flow control and error control information. When a connection is terminated, each end of the connection will not accept another connection from the same `host:port` that was previously active for a specified timeout period, usually 10 minutes.

Some TCP/IP implementations go further: they will not allow **ANY** connection to be **originated** (via the `bind()` system call or API) from a port that was active, or accepted on a port that was active for this timeout period.

Now let us see what happens when we have a client program, which must originate a connection on port 721-731, connect to the server, which waits for a connection on port 515. We first try to make a connection from host:port `1.1.1.1:721` to `1.1.1.2:515`. The first time that we make the connection (or the first connection) we succeed. We can transfer a file, etc., and then close the connection. When we try to reconnect from `1.1.1.1:721` to `1.1.1.2:515` we get an error such as "address already in use" or "connection refused".

Luckily, we can use port 722 to originate a connection, and we can connect from `1.1.1.1:722` to `1.1.1.2:515`. We continue on, until we come to port 731, and then we need to wait for our timeouts.

SOLUTION:

It appears that most RFC1179 implementations do not check for the exact port range 721-731, but only that the connection originates from a reserved port, i.e. - in the range 1-1023. You can extend the range of ports used by LPRng by changing the

```
originate_port=721 731
```

value in the defaults (`LPRng/src/common/defaults.c`) file or in the `lpd.conf` file. I recommend the following:

```
originate_port=512 1022
```

This is, in fact, now the default in LPRng software. If you get the infamous `malformed from address` error message from your spooler, then you will have to set originate_port=721 731, and live with a delayed throughput.

6.3 Job is not in print queue, but it gets printed!

In the original BSD LPD implementation, the LPR program copied users files to a special spool queue directory, and then caused the LPD server to peek in the directory and print the files.

This type of operation required spool directory space, special SETUID programs, and a slew of headaches in system security and management.

The LPR, LPQ, and other user programs in the LPRng suite use TCP/IP connections and transfer jobs directly to a LPD server running on a remote host, or even the local host if appropriate. Note that this type of operation does not require a LPD server to run on each local machine. In fact, you can have a single host system performing all of your printing. This type of operation is very similar to a central mail server versus individual systems, each having their own mail server and queues.

However, some users require or want their jobs to be spooled on the local host system, and then transferred to the remote printer. This is usually the case when some type of processing (filtering) is needed in order to print the job correctly. There are several methods that can be used to force this.

Method 1: Explicit Printer Address

You can force a job to be sent directly to the `pr` serviced by the LPD server on `host` by using the form:

```
lpr -Ppr@host file
```

You can also set the `PRINTER` environment variable to a similar form, and get the same effect:

```
PRINTER=pr@host; export PRINTER;
lpr file
```

Method 2: User and Server Printcap Entries

If you want to have the benefits of a printcap file, i.e. - you can use aliases or abbreviations for the names of printers, then here is a couple of hints. First, the LPRng software scans the `/etc/printcap` file for printcap entries, combining information for the same printer into a single entry. Information found later in the printcap file will override earlier information. In addition, you can tag entries as either being used for all utilities or just for the LPD server. Here are a couple of examples:

```
# for all utilities
pr:lp=pr@host
# just for LPD
pr:server
  :lp=/dev/lp
# more information
pr:check_for_nonprintable@
# --- final result for LPR
pr:lp=pr@host:check_for_nonprintable@
# --- final result for LPD
pr:lp=/dev/lp:check_for_nonprintable@
```

As you can see, the `server` keyword indicates that the printcap entry is only for the printer. The LPR utility will send the job to the host, while the LPD server will print it on `/dev/lp`.

Note that the `lp=...` information overrides the `:rp:` (remote printer) and `:rm:` (remote machine) fields if they are present.

Method 3: Force sending to server on `localhost`

The `force_localhost` printcap or configuration flag forces non-LPD applications to send all requests and print jobs to the server running on the local host.

This method is similar to the previous one, but has the benefit that it can be configured as a global (i.e. - applies to all printers) rather than printer specific. You can put this in the `/etc/lpd.conf` file for general application, or have a printcap entry of the following form:

```
# for all utilities
pr:lp=pr@host:force_localhost
```

The LPD server will ignore the `force_localhost` flag, and send jobs to the `pr` queue on the `host` machine. However, the LPR, LPQ, etc., utilities will send their requests to the server running on the local host.

6.4 Job disappears and is never printed, but lpr works

This is a rather disconcerting problem, and usually occurs when sending jobs to either a network printer or a nonconforming 6.6 (RFC1179) print spooler. For details about LPRng and RFC1179, see 18 (RFC1179 and LPRng), but here is a quick explanation.

An LPD job consists of a control file, which contains information about the job, and one or more data files. RFC1179 is silent on the order that jobs are sent; however some implementations REQUIRE that the data files be sent first, followed by the control file.

SOLUTION:

Set the `send_data_first` flag in the printcap for the particular printer, or in the `lpd.conf` configuration file. This is:

```
:send_data_first:  (printcap)
send_data_first    (lpd.conf)
```

Note that some printers/servers INSIST on the control file first; You can clear the flag using `send_job_first@` if you need to.

6.5 I get messages about bad control file format

RFC1179 describes a set of fields that MAY appear in the control file. It is silent if other ones can appear as well. Unfortunately, some implementations will reject jobs unless they contain ONLY fields from a very small set. In addition, RFC1179 is silent about the ORDER the fields can appear.

LPRng quite happily will accept jobs from poor or nonconforming RFC1179 spooler programs, and fix them up to be comformant.

If you are sending jobs to one of a non-conforming spooler, you can force LPRng to send jobs with only the fields described in RFC1179 by setting the the `:bk:` (BacKwards compatible) flag in the printcap for your printer.

6.6 What is RFC 1179, the Line Printer Daemon Protocol?

RFC1179 defines a standard method by which print jobs can be transferred using the TCP/IP protocol between hosts. The standard was developed by simply detailing the way that a version of the BSD LPD software did its job.

From the RFC Introduction:

> RFC 1179 describes a print server protocol widely used on the Internet for communicating between line printer daemons (both clients and servers). RFC1179 is for informational purposes only, and does not specify an Internet standard.

Having said this, the RFC then goes on to describe the protocol used by a particular implementation of LPD. The problem was that the RFC did not provide any way to put extensions to the operations into the system, and failed to specify such interesting details as the order in which print jobs and their components could be transferred.

Comment by Patrick Powell <`papowell@astart.com`> :

> Since 1988, there have been a large number of print spooling systems developed which claim RFC1179 conformance, but which are mutually incompatible.

Rather than live with the limited capabilities of the RFC1179 standard, LPRng has extended them by adding capabilities to perform remote control of print spoolers, encrypted and authenticated data transfers, and other operations missing from the RFC1179 specification. However, great effort was made to be backwards compatible with older and other LPD based systems.

LPRng was developed in order to be able to both accept and provide interactions with these systems. It does so by allowing various options to be used to *tune* how print jobs would be exchanged. Currently, LPRng can be configured to send and receive print jobs between a vast number of the existing spooling systems. It is flexible enough to act as a gateway between non-compatible systems, and has provisions to transform jobs from one format to another in a dynamic manner.

For a detailed explanation about LPRng and RFC1179, see 18 (RFC1179 and LPRng).

6.7 I want to replace lp, lpstat, etc, but my programs need them

LPRng was designed as a replacement the BSD printing system. As such, it inherited its command names and options from the latter. As you might know, System V uses a totally different set of commands, incompatible with the BSD ones.

The good news is that the LPRng binaries include an emulation for the System V commands. (See 4.11 (lp Simulation) for details. Briefly, you create links to the appropriate programs, and invoke them by the link names. *Actually, these links are installed by default in recent versions.*

```
ln -s lpr lp
ln -s lpq lpstat
ln -s lprm cancel
```

If you make these links, calling `lp`, `lpstat` and `cancel` will give you a (partial) SVR4 emulation. They have their own man pages, which you should read if you need the emulation.

Since it is a **partial** emulation, you shouldn't expect everything to work. In particular, I would guess that any script which relies on the output format of one of your system binaries will break. Again, see 4.11 (lp Simulation) for more details or additional suggestions.

6.8 What are the drawbacks to LPRng?

There are many reasons to run LPRng, and most of them are related to the extra features it has, compared to vanilla BSD LPR. A short list is given in section 1.1 (What is LPRng?). (A more elaborate description can be found in the LPRng package itself.)

On the other hand, there are also reasons **not** to switch to LPRng:

- Switching from a System V system will require some work. On the other hand, getting System V printing to work correctly for you may be even more work.
- You don't need any of the enhanced features, and are not worried about security issues. On the other hand, you may not have known about them.
- While there are many resources and books devoted to the old BSD printer daemon, documentation for LPRng is rather limited. On the other hand, you get lots of diagnostics and error messages when things go wrong.

7 Client To Server Connection Information

The LPRng software is a true set of client/server applications. The LPRng clients, `lpr, lprm, lpq,` and `lpc` connect to a `lpd` server using a TCP/IP connection. This means that you must have TCP/IP networking enabled on your workstation to use LPRng.

However, you do not need to have an external network connection to the Internet. For most single system users, the `lpd` server is running on the same workstation as the client program, and the clients will simply talk to the `localhost`.

7.1 Printer Name and Server IP Address

Options used:

- `PRINTER` *Environment variable*
- `force_localhost` *force clients to send requests to localhost*
- `default_printer_when_unknown` *used by LPD when printer name not in printcap*

When an LPRng client such as `lpr`, `lpq`, `lprm`, or `lprc` needs to communicate with a print server, the only information they normally need is:

1. The spool queue or *remote printer* to be used in requests to the `lpd` print server. This is sometimes referred to as the *printer* or *print queue* name.
2. The IP address or hostname of the *print server* or *remote server*. This is sometimes called the *remote host* or simply *server*.
3. Options that control *how* jobs or commands are transferred to the server host. These options might include encryption and client side job filtering. We will discuss these capabilities later.

LPRng has several ways to specify the *printer queue* and *server* information.

7.1.1 Command Line -Pprinter@host

The `-P printer@host` option specifies both the print queue and server.

```
lpr -Plaser@10.0.0.1
lpq -Plp@myserver
```

When used as a command line option, the printcap database will not be consulted for other options or information. This allows LPRng clients to function without a printcap database, and without a `lpd` print server running on their local host. However, options set in the `/etc/lpd.conf/` and the compile time defaults will still be used.

7.1.2 Command Line -Pprinter

This form will cause the LPRng clients to look in the `/etc/printcap` for a printcap entry with the name or alias `printer` and use the information in that printcap entry. We will discuss the format of the printcap entry in a moment. Example:

```
lpr -Plp
```

7.1.3 PRINTER Environment Variable

If no command line option is specified, the LPRng clients will check for a `PRINTER` environment variable value and will use it as though specified as a `-P$PRINTER` command line option.

If the $PRINTER value has the form `printer@host` the print queue will be `printer` on server `host`. and not consult the printcap database If the $PRINTER value has the form `printer` then the printcap will be searched for a `printer` printcap entry. For example:

```
export PRINTER=laser@10.0.0.1; lpr
export PRINTER=pr; lpr
```

7.1.4 Default Printer From Printcap

If you do not specify a printer on the command line or in the `PRINTER` environment variable, then LPRng will search the printcap and use the first valid printcap entry as the printer.

7.1.5 Default Printer When No Printcap

If you do not have a `/etc/printcap` file, then LPRng will use the `default_printer` and `default_remote_host` *fallback* values set in the `/etc/lpd.conf` file or by the compile time defaults.

Using the fallback values is usually not a desirable event and may indicate that you have a misconfigured host, so the fallback values are usually set by administrators to `missingprinter@localhost` to provoke an annonying message for users.

7.1.6 LPD Default Printer

For completeness, there is even a `default_printer_when_unknown` configuration entry for use by LPD when it is given a printer name not in its printcap database. By default, it will look up this name in the database and treat the job as though it was sent to this printer.

7.2 LPD Server Runnning on Localhost

Workstations for personal use or in extremely simple configurations will always run an `lpd` server on the *localhost*. This is the most common situation for the majority of new users, and the default LPRng installation sets the default value of the `force_localhost` configuration parameter to **TRUE** or to 1. The `lpr` `lpq` `lprm` and `lpc` will connect to `localhost` (usually IP address 127.0.0.1) unless explicitly overridden by the command line `-Pprinter@host` argument.

Larger sites or organizations which want to use a central print server to handle multiple printers or printer sharing may want the clients to connect directly to the server. In this case the system administrator should set `force_localhost@` in the `/etc/lpd.conf` or modify the compile time default. See 8.1 (Simple Client Printcap Entry) for ways to change this.

8 Using the Printcap Database

As described in the 2 (Print Spooling Overview), the heart of the LPRng system is information in the `/etc/printcap` file. The printcap information specifies:

1. The print queues available to users.

2. How client programs communicate with the `lpc` print server.
3. The configuration, location, and other information for each print queue on the print server.
4. How the `lpd` server processes jobs in each print queue.

In order to explain a complex subject, we will start with a set of simple printer configurations, and explain the purpose and effect of each entry in the printcap.

For details about individual printcap options, see the `printcap(5)` man page from the LPRng distribution, or use the 21 (Index To All The Configuration and Printcap Options) to find a specific printcap option and its effects.

8.1 Simple Client Printcap Entry

Options used:

- `rm=`*remote host (machine)*
- `rp=`*remote printer*
- `lp=`*destination printer information*
- `client` client only printcap FLAG

I'll use this simple example to explain the basics of the LPRng printcap format and introduce some of the LPRng network configuration options. Here is a simple printcap file used to provide client programs (`lpr, lprm,` etc) with *remote printer* and *server* information.

```
# printer lp1
lp1|printer1
  :rm=localhost
# printer lp2 with continuation
lp2:\
  :lp=pr@10.0.0.1:client
# printcap lp3, to printer pr, with overrides
lp3:rp=pr:rm=hpprinter.astart.com
  :force_localhost@
# Simplest possible printcap entry - defaults for everything
lp4
```

1. Lines starting with a `#` sign are comments, and all leading and trailing *whitespace*, i.e. - spaces, tabs, etc, are ignored. Empty lines are ignored as well.
2. A printcap entry starts with the printcap entry **name**, followed by one or more *aliases*, followed by one or more options. In the above example we have three printcap entries: `lp1` with an alias `printer1` and `lp2`, `lp3`, and `lp4` with no aliases.
3. Aliases start with the | character and options with the `:` character; tabs and spaces before and after the | or `:` characters and at the start and end of lines are ignored. You can use backslash (tt/\/) at the end of a line to create a multi-line value for an option. The backslash will cause the next line to be appended to the current line; watch out for comments and ends of printcap entries if you use this facility. As you can see from the example, there is no `Name` printcap entry - this is part of the `cm` option on the previous line.
4. Options take the form of a keyword/value pair, i.e.-

```
:option=value
:option#value   (legacy, not advised for new systems)
:option
:option@
```

5. Option names are case insensitive, but option values are not. While `Ts` and `ts` are the same option name, `ts=Testing` and `ts=testing` have their case preserved. A string or integer value is specified by `option=value` or `option#value`.

6. The use of the legacy `option#value` form is **NOT** recommended as some preprocessors and database systems will treat # as the start of a comment and delete the remainder of the line. This has caused great consternation for sysadmins who wonder why their NIS distributed printcap entries have been mysteriously truncated.

7. If you want to set a string option to *empty* value, use `option=`. The `option` will set it to `1`. If an option value contains a colon, then use the C (or Perl or Tck/Tk) string escape `\072` to represent the value.

8. Boolean options are set TRUE (1) if no value follows the keyword and FALSE (0) by appending a `@`. For example `sh` will set `sh` to TRUE and `sh@` to FALSE.

There may be multiple options on the same line, separated by colons. However, this does make the file less readable. The next tip was supplied by James H. Young <`jhy@gsu.edu`>:

> My personal preference for readability is to always put each option on its own line. Putting each option on its own line is worth the trouble even though it detracts from the usability of certain grepping techniques when trying to maintain these types of files.

Now let's examine the first printcap entry in detail. It is reproduced here for convenience:

```
# printer lp1
lp1|printer1
  :rm=localhost
```

1. We start with a comment, followed by the printcap entry name and and alias. Aliases are useful when you want to refer to a single printer or print queue by different names. This can be useful in advanced printcap and print queue setups. By default, the remote printer name is the printcap entry name.

2. The `rm` (remote machine or host) option specifies the name or IP address of the `lpd` host running `lpd`. In this example the remote host is `localhost` or the machine that the client is running on and we assume that the `lpd` server is running on the localhost. Thus, we would communicate with printer `lp1@localhost`.

Let's look at the next printcap entry:

```
# printer lp2 with continuation
lp2:\
  :lp=pr@10.0.0.1:client
```

1. The `lp2` printcap entry illustrates the use (and abuse) of the \ continuation. If you think about this, we have really defined a printcap entry of the form:

   ```
   lp2: :lp=pr@10.0.0.1:client
   ```

Luckily, LPRng ignores empty options like `: :`. While it is strongly recommended that \ be avoided it may be necessary for compatiblitye with other system utilities.

2. The `lp=pr@10.0.0.1` option is an alternate way to specify a remote queue and server. If the `force_localhost` default is being used, then the LPRng clients will ignore the `10.0.0.1` address and still connect to `pr@localhost`. There is further discussion about this in the next section.

3. The `client` option explicitly labels client only printcap information. The `lpd` server will ignore any printcap with the `client` option. When constructing complex printcaps, this option is used to keep ensure that you have consistent printcap information.

The following printcap entry shows how to override the `force_localhost` default, and force the LPRng clients to connect directly to a remote server:

```
lp3:rp=pr:rm=hpprinter.astart.com
  :force_localhost@
```

1. The `rp=` (remote printer) remote print queue name to used when sending commands to the `lpd` print server.

2. The `force_localhost@` option is an example of a *flag* option. The `@` sets the option value to 0 (false). We set `force_localhost` to false, which now allows the LPRng clients to connect directly to the specified remote printer. In this example, the `hpprinter.astart.com` could be a HP LaserJet Printer with a JetDirect interface, which supports the RFC1179 protocol.

3. One disadvantages of sending a job directly to a printer using the above method is that `lpr` program will not terminate or exit until all of the files have been transferred to the printer, and this may take a long time as the printer processes the files as they are received.

Now let's look at the last printcap entry:

```
# Simplest possible printcap entry - defaults for everything
lp4
```

The last example is the simplest possible printcap entry. This will cause LPRng clients to use the default values for everything. The printer will be `lp4`, i.e. - the name of the printcap, and the server will be `localhost` if `force_localhost` is set, or the value of the `default_remote_host` configuration option if it is not.

8.2 Simple Server Printcap Example

Options used:

- `cm=`*comment for status*
- `if=`*default job file filter*
- `lf=`*log file*
- `mx#`*maximum job size*
- `lp=`*output device*
- `sd=`*spool directory file*

- `sh` *suppress headers (banners)*
- `sf` *suppress form feeds between files*

The previous section discussed printcap entries for use by the client programs. Now we will discuss printcap entries for use by the `lpd` server. In simple configurations or when we have the `force_localhost` option enabled we can use the same printcap for both LPRng clients and the `lpd` server.

```
# Local ASCII printer
lp1|printer
  :server
  :cm=Dumb printer
  :lp=/dev/lp1
  :sd=/var/spool/lpd/lp1
  :lf=log:af=acct
  :if=/usr/local/sbin/lpf
  :mx#0:sh:sf
```

1. The printcap entry name is `lp1`. This information will be displayed when requesting status information using the `lpq` program.
2. The `printer` alias. This allows a single spool queue to have multiple names.
3. The `server` corresponds to the `client` flag and indicates this printcap entry is for the `lpd` server only.
4. The `cm` field supplies a information field for `lpq` (printer status) output.
5. The `lp` value specifies the destination file, device or remote spool queue to which data is sent. In this example it is the device `/dev/lp1`. By default, IO devices are opened for *write-only* operation.
6. The `sd` specifies the *spool directory* where print job files are stored until they are printed.
7. The `lf` and `af` options specify the names for the log and accounting files, respectively. These have the default values `log` and `acct` respectively, and if not absolute pathnames are relative to the spool queue directory. A log file is highly recommended. If these files don't exist, they will not be created, and no logging or accounting will be done. You will need to create them manually (e.g., by using `touch`) or by using the `9 (checkpc)` program.
8. The `if` option specifies a filter program to be used for jobs with the `f` or default job format format. Filters and print formats are discussed in section 13 (Filters). The `lpf` filter translates `LF` (line feed) to `CR/LF` (carriage return/line feed) sequences, eliminating *staircase* output.
9. `mx` indicates the maximum file size for a print job. Specifying 0 means that there is no limit.
10. The `sh` *suppress headers* flag will suppress printing banner pages.
11. The `sf` *suppress form feeds* flag will suppress form feeds between the files of a multi-file print job.

8.3 The Printcap Parsing Rules

In this section, we will discuss the remaining tricky parts of the LPRng printcap database: combined client and server printcaps, host specific printcap entries, and the `tc` *include* facility.

The following is a complete description of how a printcap file is processed:

1. When processing a printcap file, the LPRng software reads and parses each entry individually. Leading whitespace is removed. Lines starting with `#` and blank lines are ignored.

2. Lines ending with \ will have the \ discarded, and all lines of a printcap entry are joined by removing the line separators (`\n`) and replacing them with a space.
3. The printcap entry is parsed, and the printcap name, aliases, and options are determined. Colons `:` act as option separators, and leading and trailing whitespaces are removed.
4. Options are sorted and expect for the `tc=...` option only the last option setting is retained.
5. Client programs will discard a printcap entry with a `server` option and server programs will discard a printcap entry with a `client` options.
6. The `oh` (**o**n this **h**ost) option specifies a list of IP addresses and mask pairs or glob strings which are used to determine if this printcap entry is valid for this host (see discussion below).
7. After the above processing, if there is an existing termcap entry with the same name, the two sets of options are combined, with the last option setting retained except for the `tc` entries which are combined.
8. When a printcap entry is actually used, the printcap entries listed by the `tc` include option are extracted and combined in order. (This allows include entries to appear after the referring printcap entry.) Then printcap options will be combined with the included ones. This has the effect that the options specified in the printcap entry will override the ones from the `tc` included entries.
9. Finally, each string printcap option with a *%X* value has *%X* replaced by the following values. Unspecifed values will not be modified.

```
%P          printcap name
%h          short host name  (host)
%H          fully qualified host name  (host.dns.whatever)
%R          remote printer (rp value)
%M          remote host (rm value)
%D          date in YYYY-MM-DD format
```

10. When parsing multiple printcap files, these are processed in order, and all of their printcap entries are combined according to the above procedures. The `tc` resolution and *%X* expansion is done after all the files have been processed.

The following examples show how to use the above rules to your advantage. You can combine both client and server printcap information in a single file as well as dividing a printcap entry into several parts. Here is an example:

```
# seen by both client and server
lp1:lp=lp@pr1:mx#100
lp1:sd=/usr/local/spool/lp1:mx#0
# seen only by client
lp2:lp=lp@pr2:client
# seen only by server
lp2:lp=/dev/lp:server
```

1. Printcap entries with the same name are combined. The first printcap entry, `lp1`, the information is seen by both client and server. The next printcap entry, with the same name `lp1`, will be combined with the second one. The order of options is important - the entries are scanned in order and an option will have the last value set. Thus, after having read both the `lp1` printcap entries, both client and server will have:

```
lp1:lp=lp@pr1
  :mx#0
  :sd=/usr/local/spool/lp1P
```

2. The `lp2` has a client and server version. This is recommended when complex printcaps on multiple hosts and servers are used. Thus, the LPRng clients will see:

```
lp1
  :lp=lp@pr1
  :mx#0
  :sd=/usr/local/spool/lp1P
lp2
  :client
  :lp=lp@pr2
```

   and the server will see:

```
lp1
  :lp=lp@pr1
  :mx#0
  :sd=/usr/local/spool/lp1P
lp2
  :lp=/dev/lp
  :server
```

If you have multiple printers of the same type whose configuration is almost identical, then you can define a set of *tc* only printcap entries containing common information and use the `tc` include facility.

By convention, all printer names start with an alphanumeric character and contain only alphanumeric values, underscore (`_`) or hyphen (-). A printcap entry starting with punctuation such as period (`.`) or at (`@`) is processed by LPRng but can only be used with the `tc` include facility. For example:

```
.hp:
  :sd=/usr/local/spool/%P
  :mx#0:sf:sh
hp1:tc=.hp,.filter
  :lp=lp@10.0.0.1
hp2:tc=.hp,@filter
  :lp=lp@10.0.0.2
@filter
  :if=/usr/local/bin/ifhp
  :lpd_bounce
```

1. The `.hp` and `@filter` printcap enties will not be used as real printcaps by LPRng, but can be referenced by the `tc` printcap include facility. After `tc` include processing is completed, the printcap information would resemble:

```
hp1
  :lp=lp@10.0.0.1
  :if=/usr/local/bin/ifhp
  :lpd_bounce
  :mx#0
  :sd=/usr/local/spool/%P
  :sf
  :sh
hp2
```

```
        :lp=lp@10.0.0.1
        :if=/usr/local/bin/ifhp
        :lpd_bounce
        :mx#0
        :sd=/usr/local/spool/%P
        :sf
        :sh
```

2. The *%X* processing will replace `%P` with the printcap name, so we would have:

```
    hp1
      :lp=lp@10.0.0.1
      :mx#0
      :sd=/usr/local/spool/hp1
      :sf
      :sh
    hp2
      :lp=lp@10.0.0.1
      :mx#0
      :sd=/usr/local/spool/hp1
      :sf
      :sh
```

When administring a large number of printers over a large area, it is sometimes desireable to have a *default* printer for each host. This default printer may be different for each host, and can be selected by using the `oh` entry. The `oh` value is a list of the following entries

```
IP/n      - address + mask length    10.0.0.0/8
IP/IP     - address + mask           10.0.0.0/255.0.0.0
vvv       - glob for hostname        pc*.org.com
```

The LPRng software will determine the hostnames and IP addresses assigned to the host and then check to see if there is a match in the listed hostnames or IP addresses. If there is a match, the printcap entry will be used. If not, then the entry will be discarded. For example:

```
lp:oh=*.admin.org.com,10.0.0.5,10.2.0.0/16:lp=pr1@server1
lp:oh=*.eng.org.com:lp=hp@server2
```

1. In the above example, if our host name is `booster.admin.org.com`, then we would use `lp=pr1@server1`, as the `*.admin.org.com` glob pattern would match our host name.
2. if our host name is `booster.dev.org.com` and our IP address is 10.2.0.1, then we would use `lp=pr1@server1`, as the `10.0.0.0/16` ip address would match.

8.4 Displaying Printcap Information

If you are generating complex printcap entries, you might need to find out exactly what the LPRng servers or clients will actually see. The LPRng software has several diagnostic tools to help you. The most simple to use is the `lpc` program.

The `lpc client all` and `lpc server all` commands will display the printcap information that the LPRng clients and `lpc` server would see when executing on the host. For example:

```
#> lpc client all
lp1
  :lp=lp@pr1
  :mx#0
  :sd=/usr/local/spool/lp1P
lp2
  :client
  :lp=/dev/lp
```

8.5 Remote Printer Using RFC1179

Options used:

- `lp=`*destination*
- `rm=`*remote host (machine)*
- `rp=`*remote printer (machine)*

You can have the `lpd` server forward jobs to another server or print which supports the RFC1179 procol by using the following printcap:

```
# Simplest
remote|Remote Printer
   :lp=raw@server
# historical
remote:
  rp=raw:rm=server
# Sometimes you have to connect to a non-standard port
special:lp=lp@server%2000
```

1. If the `lp` printer entry is present, it will override the `rm` and `rp` printer entries.
2. The `lp=pr@host` format specifies that the output device is actually a remote spool queue, and jobs should be transferred using RFC1179 protocol.
3. By default, LPRng will attempt to **sanitize** all jobs that it originates or forwards. This sanitization will result in an RFC1179 compliant `control file`, and will not modify any of the job information.

8.6 Remote Printer Using Socket API

If the spool queue destination is a remote printer supporting the Socket API, then you can have LPRng open a connection directly to the printer. These include the older Apple printers with TCP/IP support and the HP JetDirect supported printers.

```
# Simplest
remote
   :lp=10.24.2.3%9100
   :sh:sf
```

1. The `lp=server%port` or `lp=IPaddr%port` format specifies that `lpd` should open a TCP/IP connection to the remote host and simply transfer verbatum the files to be printed.
2. The `sh` and `sf` will prevent `lpd` from trying to generate banner pages or put form feeds between jobs.

While this is the simplest printcap, it is also the most dangerous as there is nothing to prevent a malformed job from being sent to the printer. The next printcap example is much more robust:

```
# Simplest
remote
   :lp=10.24.2.3%9100
   :of=/usr/local/lib/ifhp
   :if=/usr/local/lib/ifhp
   :sh:sf
```

1. This version will use the `ifhp` filter to precondition the printer and to process jobs. See 13.9 (IFHP Filter) for details. The `ifhp` filter will perform the appropriate printer resets, translate job information, and ensure correct printer operation in the presence of errors. It will also produce voluminous error messages and logging information.

8.7 Parallel Printer

The parallel printer printcap is very simple.

```
# parallel printer
lp:
   :lp=/dev/lpr
   :sh:sf
```

1. The `lp=/dev/lpr` specifies that `lpd` should open the device for APPEND and simply transfer job files to it.
2. The `sh` and `sf` will prevent `lpd` from trying to generate banner pages or put form feeds between jobs.

If you discover that UNIX print jobs result in a *staircase* appearance, then you need to force your printer to do `LF` (linefeed) to `CR/LF` (carriage return/line feed) translation, or do the translation yourself.

```
# Simple parallel printer
lp:
   :lp=/dev/lpr
   :if=/usr/local/bin/lpf
   :sh:sf
```

By using the `if=...lpf` filter, the job will be passed through the `lpf` filter, which will do the `LF` to `CR/LF` translation.

If you have a more complex printer that handles PostScript, PCL, and PJL, then you will need to use the more powerful `ifhp` filter:

```
# Simple parallel printer
lp:
   :lp=/dev/lpr
   :ifhp=model=hp4,status@
   :of=/usr/local/bin/ifhp
   :if=/usr/local/bin/ifhp
   :sh:sf
```

See 13.9 (IFHP Filter) for details. This entry will specify that the printer is an HP4, and that no status information is available. This is usually the case with a parallel port.

8.8 A serial printer queue

Options used:

- `rw` *device opened RW flag*
- `br#`*serial port bit rate*
- `stty=`*stty options for serial port configuration*

The following is a typical printcap for a serial printer:

```
# Local Serial ASCII printer
lp2
  :lp=/dev/ttya
  :rw
  :cm=Serial printer
  :sd=/var/spool/lpd/lp2
  :stty=9600 -echo -crmod -raw -oddp -evenp pass8 cbreak ixon
  :if=/usr/local/sbin/lpf
  :mx#0:sh
```

Let's examine the new options:

1. A serial port is usually *bidirectional*, and printers will report errors back to the host computer. The `rw` flag will cause the printer port to be openned `read-write`, and the `lpd` server will report status information.
2. The `sy` option specifies the `stty(1)` flags and line speed needed to configure the serial line (See 3.4 (Serial Printers) for details).
3. The legacy `br` (bit rate) option can be used to specify the line speed as well.

8.9 Single Printcap File for Large Installation

One of the major problems faced by administrators of large sites is how to distribute printcap information. They would like to have a single printcap file either distributed by a file server (NFS) or by some other method such as `rdist`. By using the `server` and `oh` tags, information for the specific sites can be separated out. For example:

```
#/etc/printcap file
pr1:lp=pr1@serverhost1:oh=*.eng.site.com,130.191.12.0/24
pr2:lp=pr1@serverhost1:oh=*.eng.site.com,130.191.12.0/24
pr1:lp=pr2@serverhost2:oh=*.admin.site.com
pr2:lp=pr2@serverhost2:oh=*.admin.site.com
pr1:server:oh=serverhost1.eng.com:lp=/dev/lp:tc=.common
pr2:server:oh=serverhost2.admin.com:lp=/dev/lp:tc=.common
.common:sd=/usr/local/lpd/%P
```

The above example has some interesting effects. The `pattern` is used as a *glob* pattern and is applied to the fully qualified domain name (FQDN) of the host reading the printcap file. For example, `*.eng.site.com` would match host `h1.eng.site.com` but would not match `h1.admin.site.com`. Thus, the effects of the first couple of entries would be to specify that the `pr1` and `pr2` printers on the `eng` hosts would be `pr1@serverhost1`, and on the `admin` hosts would be `pr2@serverhost2`,

Also, the lpd daemons on `serverhost1` and `serverhost2` would extract the additional information for `pr1` and `pr2` respectively, overriding the common `lp` entries.

8.10 Bounce queues

Options used:

- `lpd_bounce` *lpd filters and then forwards*
- `bq=`*destination for filtered job*
- `bq_format=`*format of filtered job*

When the destination of a spool queue is another spool queue the job is simply forwarded without any modifications. However, sometimes it is essential that the job be modified before forwarding, as when the remote spool queue is actually a printer, and jobs need to be converted to the format acceptable by the remote printer or banner pages added.

The `lpd_bounce` flag marks a spool queue as a bounce queue. `Lpd` will perform all of the usually job processing steps, such as banner generation, filtering files, etc, but saves the output to a file. This file is then sent to the destination print queue for further processing.

```
# Simple example of a bounce queue
bounce:lp=bounce@bouncehost
bounce:server
    :lp=lp@remote
        :lpd_bounce
        # LEGACY
        #bq=lp@remote
    :sd=/usr/spool/lpd/%P
    :if=/usr/local/bin/lpf
    :vf=/usr/local/bin/lpf
    :bq_format=l
    # uncomment ab if you want banner
    #ab
```

Some comments:

1. The `lpd_bounce` option marks the job as a bounce queue, and the `lpd` server will process the job through the appropriate filter programs.
2. The legacy `bq=host` has the same effect as `lpd_bounce`. This option is retained for compatibility with previous versions of LPRng. It is recommended that this option not be used.
3. The printcap has filter specifications for different job formats. These are the programs that will be used by LPRng to process the job.
4. The `bq_format` specifies the job format for the output file sent to the remote spool queue. If not specified, it defaults to `l` (literal or binary).
5. The `ab` (always print a banner) flag will force a banner to be added to the job. The banner generation is done as discussed in 12.2 (Banner Printing).

8.11 Changing Job Formats

Options used:

- `translate_format=`*format of filtered job*

A rarely encountered problem is when a job is printed with one format but for compatiblity needs to be processed with another. The simple `translate_format=vlxf` option will rename format `x` files to `f` format.

This can be used to resolve problems with PC based sofware, which spools jobs using the `v` format. Unfortunately, many RCF1179 print spoolers do not understand the `v` format and mishandle the job. A simple forwarding queue with the following entries will rename `v` format to `l` (binary) format.

```
lp
  :sd=/usr/spool/lpd/%P
  :translate_format=vl
  :lp=lp@printerserver
```

8.12 LPR Filters Files

Options used:

- `lpr_bounce` *lpr does filtering*

Some users would like the advantages of the filtering and processing capabilities of a lpd daemon without running a lpd daemon on their system. By having the `lpr` program process the job by passing it through the various filters and then send the output of the filters as the print job you can get the desired effect.

```
# Simple example of an lpr_bounce entry
bounce
  :lpr_bounce
  :lp=lp@remote
  :if=/usr/local/bin/lpf
```

The `lpr_bounce` flag, if present in the printcap entry, will force `lpr` to process the job using the specified filters and send the outputs of the filters to the remote printer for further processing.

8.13 Dynamic Routing

Options used:

- `destinations=`*destinations for jobs*
- `router=`*router program*

LPRng has the ability to route a job to one or more destinations in a dynamic manner. This is not the same as *load balancing*, as the destinations are hard coded and not able to be changed. This is accomplished by having a `router` filter generate a set of destinations. Here is a sample printcap entry:

```
t2|Test Printer 2
    :sd=/var/spool/LPD/t2
    :lf=log
    :destinations=t1@server1,t1@server2,t1@localhost
    :router=/usr/local/LPD/router
```

When a job arrives at the `lpd` server, the 'router' filter is invoked with the standard filter options which include the user, host, and other information obtained from the control file. STDIN is connected to a temporary copy of the control file, and the CONTROL environment variable is set to the value of the actual control file itself.

The routing filter exit status is used as follows:

- 0 (JSUCC) - normal processing
- 37 (JHOLD) - job is held
- any other value - job is deleted from queue

The router filter writes to STDOUT a file specifying the destinations for the job. The destinations entries in this file file have the following format. Entry order is not important, but each destination must end with the 'end' tag.

```
dest (destination queue)
copies (number of copies to be made)
priority (priority letter)
X(controlfile modifications)
end
```

Example of router output:

```
dest t1@localhost
copies 2
CA
priority B
end
dest t2@localhost
CZ
priority Z
end
```

In this example, two copies of the job will be sent to the t1 and t2 spool queue servers. The Class (C letter value) and job priority information will be rewritten with the indicated values.

If routing information is specified by the router filter the job will be sent to the default destination.

LPQ will display job information in a slightly different format for multiple destination jobs. For example:

```
Printer: t2@astart2 'Test Printer 2' (routed/bounce queue to 't1@astart2.astart.com')
  Queue: 1 printable jobs in queue
 Rank  Owner/ID        Class Job Files                           Size Time
active  papowell@astart2+707 A 707  /tmp/hi                         3 10:04:49
 - actv papowell@astart2+707.1 A 707 ->t1@localhost <cpy 1/2>       3 10:04:49
 -      papowell@astart2+707.2 A 707 ->t2@localhost                 3 10:04:49
```

The routing information is displayed below the main job information. Each destination will have its transfer status displayed as it is transferred. By convention, the job identifier of the routed jobs will have a suffix of the form .N added; copies will have CN added as well. For example, papowell@astart2+707.1C2 will be the job sent to the first destination, copy two.

Routed jobs can be held, removed, etc., just as normal jobs. In addition, the individual destination jobs can be manipulated as well. The LPC functionality has been extended to recognize destination jobids as well as the main job id for control and/or selection operations.

The optional `destinations` entry specifies the possible set of destinations that the job can be sent to, and is for informational purposes only. In order for LPQ/LPRM to find the job once it has passed through LPD, LPQ/LPRM uses the list of printers in the `destinations`, and loop over all the names in the list looking for the "job" that you are interested in. If there is no `destinations` information, the `bq` information will be used.

Lars Anderson <lsa@business.auc.dk> supplied this example (slightly edited):

This script will attempt to distribute print jobs evenly on 2 printers hpl5a and hpl5b when sending to hpl5bounce.

```
hpl5bounce|for PLP/LPRng software - network based HP Jetdirect card:
        :lpd_bounce
        #default
        :rp=hpl5b
        :destinations=hp5a,hp5b
        :router=/usr/local/admscripts/bouncer.pl
hpl5a|for PLP/LPRng software - network based HP Jetdirect card:
        :lp=hpl5a%9100
        :tc=@hplcommon
hpl5b|for PLP/LPRng software - network based HP Jetdirect card:
        :lp=hpl5b%9100
        :tc=@hplcommon
# Common settings
@hplcommon:
        :sd=/var/spool/lpd/%P
        :rw:sh:ps=status
        :fx=flp
        :if=/usr/local/lib/filters/ifhp -Tbanner=on
        :of=/usr/local/lib/filters/ofhp -Tbanner=on
```

The perl script `bouncer.pl` looks like this:

```
#!/usr/bin/perl
#
# Script for printjob loadsharing
#   This is static, not dynamic balancing
#
# Printqueues to check
$printer1="hpl5a\@localhost";
$printer2="hpl5b\@localhost";
# obtain number of jobs in each printqueue
$lpq1=`/usr/local/bin/lpq -s -P$printer1`;
$lpq2=`/usr/local/bin/lpq -s -P$printer2`;
$lpq1=~ (/(\d+) jobs?/); $numjobs1=$1;
$lpq2=~ (/(\d+) jobs?/); $numjobs2=$1;
if ($numjobs1 == 0) {
    print "dest $printer1\nCA\nend\n";
    exit;
}
if ($numjobs1 > $numjobs2) {
    print "dest $printer2\nCA\nend\n";
    exit;
}
print "dest $printer1\nCA\nend\n";
```

8.14 Printer Load Balancing

In a large site, you could have several equivalent printers, which will be used by many people. The reason for this is, of course, to increase the printer output by enabling several jobs to be printed at once.

LPRng supplies mechanisms to define a 'virtual' printer for such a set of real printers. If properly set up, print jobs will be distributed evenly over all printers.

8.14.1 Multi-server print queue

Options used:

- `ss=`*queue served by printer*
- `sv=`*printers where jobs are sent (servers)*

A multi-server print queue is one that feeds jobs to other queues. The main queue `sv=q1,q2,...` printcap entry specifies the names of the printers that will be sent jobs. These printers must have their spool queues on this LPD server.

Servers that are fed jobs have a `ss=`*mainqueue* printcap entry. This informs the `lpd` server that the queue operates under the control of the *mainqueue* print queue, and is fed jobs from it.

During normal operation, when the `lpd` server has a job to print in the *mainqueue*, it will check to see if there is an idle *service* queue. If there is, it will transfer the job to the service queue spooling directory and start the service queue printing activities.

Users can send jobs directly to the individual printers serving a queue.

The next example (and the comments underneath) was supplied by John Perkins <`john@cs.wisc.edu`> (slightly edited).

Here's how I've set up a bounce queue that feeds 6 LaserWriters:

```
laser|pi|Room 1359 LaserWriters
    :lp=laser@server.com
laser|pi|Room 1359 LaserWriters
    :server
    :lf=/usr/adm/laser-log
    :sv=laser1,laser2,laser3,laser4,laser5,laser6
    :sd=/usr/spool/laser
@commonlaser
    :sd=/usr/spool/%P
    :rw:mx#0:sh
    :lf=/usr/adm/laser1-log
    :if=/s/lprng/lib/filters/cappsif
    :of=/s/lprng/depend/cap/bin/papof
    :ss=laser
    :fx=fdginpt
laser1|pi1|Room 1359 LaserWriter #1
    :lp=laser1@server.com
laser1|pi1|Room 1359 LaserWriter #1
    :server
    :lp=/dev/laser1
    :tc=@commonlaser
laser2|pi2|Room 1359 LaserWriter #1
    :lp=laser2@server.com
laser2|pi2|Room 1359 LaserWriter #2
    :server
    :lp=/dev/laser2
    :tc=@commonlaser
```

and so on for the other 4 `laser`N queues.

This will forward a job from `laser` to `laser`N, once one of those queues is available. It will hold jobs in the "`laser`" queue until one of the other queues is empty.

Even though the queues are not meant for direct use, people can print directly to individual queues. This allows a specific load sharing printer to be used. If you wanted to *hide* the load sharing printers, i.e. - not allow direct spooling to them, then you would simply remove the non-server entries from the printcap.

8.14.2 Checking Busy Status of Server Queues

Options used:

- `check_idle=`*check for idle printer program*

The previous section outlined how LPRng uses the `sv` and `ss` flags to indicate that the server spool queue has multiple destination queues. However, there is a problem when the actual printer being served by the destination queue is a remote device, and can be busy or offline.

The `check_idle` option specifies a program that is invoked by the `lpd` server to determine if the spool queue device is available.

The program is invoked with the standard filter options, STDIN and STDOUT connected to `/dev/null`, and STDERR to the error log.

The program should make a connection to the remote device or system and should determine that the remote device is available for use, and then exit with the following status.

```
Key      Value   Meaning
JSUCC    0       Successful - printer is idle
JABORT   non-zero Printer is not accepting jobs
```

If the printer is accepting jobs but is temporarily busy, the program should poll the printer until it becomes free, only exiting when it is available for use. If the printer is not accepting jobs, the program should exit with a non-zero exit code.

The following is a sample printcap entry, showing how the `check_idle` facility can be used.

```
pr:
  :lp=laserjet%9100
  :check_idle=/usr/local/filters/remote_check lp@laserjet
  :if=/usr/local/filters/ifhp
```

The following perl program shows how to generate a query to the remote printer by simulating an `lpq` query and checking for returned status.

```
#!/usr/local/bin/perl
# Usage:
#  remote_check printer@host[%port] [-options]
#   -Tflag[,flags]*
#  flag
#    debug  - turns debugging on
#    long   - use long status format
#
# query the remote printer whose name is passed on the command line
#
```

```
# Note that -Txxx options are passed AFTER the printer
use English;
use IO::Socket;

my $JSUCC = 0;
my $JABORT = 33;
my $JNOSPOOL = 38;
my $JNOPRINT = 39;

my $debug = 0;
my $optind;

# pull out the options
my($key,$value,$opt,$long,$opt_c);

$printer = $ARGV[0];

for( $i = 1; $i < @ARGV; ++$i ){
    $opt = $ARGV[$i];
    print STDERR "XX opt= $opt\n" if $debug;
    if( $opt eq '-c' ){
        $opt_c = 1;
    } elsif( ($key, $value) = ($opt =~ /^-(.)(.*)/) ){
        if( $value eq "" ){
            $value = $ARGV[++$i];
        }
        ${"opt_$key"} = $value;
        print STDERR "XX opt_$key = " . ${"opt_$key"} . "\n" if $debug;
    } else {
        $optind = $i;
        last;
    }
    print STDERR "XX opt_P = $opt_P\n" if $debug;
}

$long = 0;  # short

if( defined($opt_T) ){
    print STDERR "XX CHECK_REMOTE opt_T=$opt_T\n" if $debug;
    if( $opt_T =~ /debug/ ){
        $debug = 1;
    }
    if( $opt_T =~ /short/ ){
        $long = 1;
    }
    if( $opt_T =~ /long/ ){
        $long = 0;
    }
}

print STDERR "XX CHECK_REMOTE " . join(" ",@ARGV) . "\n" if $debug;

if( !defined($printer) or $printer =~ /^-/ ){
    print STDERR "$0: no printer value\n";
    exit( $JABORT );
```

```
}

while( checkstatus( $printer, $long ) ){
    print STDERR "XX CHECK_REMOTE sleeping\n" if $debug;
    sleep(10);
}

exit $JSUCC;

sub checkstatus {
    my ($printer,$long) = @_;
    my ($remote,$port);
    my ($count, $socket, $line);

    if( $long ){
        $long = 4;
    } else {
        $long = 3;
    }
    if( $printer =~ /@/ ){
        ($printer,$remote) = $printer =~ m/(.*)@(.*)/;
    }
    $remote="localhost" unless $remote;

    if( $remote =~ /%/ ){
        ($remote,$port) = $remote =~ m/(.*)%(.*)/;
    }
    $port = 515 unless $port;
    print STDERR "XX CHECK_REMOTE remote='$remote',"
        . " port='$port', pr='$printer', op='$long'\n" if $debug;

    $socket = getconnection( $remote, $port );

    $count = -1;
    # send the command
    printf $socket "%c%s\n", $long, $printer;

    while ( defined( $line = <$socket>) && $count < 0 ){
        chomp $line;
        print STDERR "XX CHECKREMOTE '$line'\n" if $debug;
        if( $line =~ /printing disa/ ){
            print STDERR "XX CHECKREMOTE printing disable\n" if $debug;
            exit $JNOPRINT;
        } elsif( $line =~ /spooling disa/ ){
            print STDERR "XX CHECKREMOTE printing disable\n" if $debug;
            exit $JNOSPOOL;
        } elsif( $line =~ /([0-9]*)\s+job.?$/ ){
            $count = $1;
            print STDERR "XX CHECKREMOTE $count jobs\n" if $debug;
        }
    }
    close $socket;
    if( $count < 0 ){
        print STDERR "CHECKREMOTE cannot decode status\n";
        exit $JABORT;
```

```
    }
    return $count;
}

sub getconnection {
    my ($remote,$port) = @_;
    my ($socket);
    print STDERR "XX CHECK_REMOTE remote='$remote', port=$port\n" if $debug;
    $socket = IO::Socket::INET->new(
        Proto => "tcp",
        PeerAddr => $remote,
        PeerPort => $port,
        );
    if( !$socket ){
        print STDERR "CHECK_REMOTE IO::Socket::INET failed - $!\n";
        exit $JABORT;
    }
    $socket->autoflush(1);
    $socket;
}
```

The example of the previous section can be modified now so that it uses the `check_idle` facility. The master queue will send jobs only to the server queue queues which report idle status.

```
laser1|pi1|Room 1359 LaserWriter #1
    :server:check_idle=/usr/local/lib/filters/remote_check pr@laser1
    :lp=laser1%9100
    :tc=@commonlaser
laser2|pi2|Room 1359 LaserWriter #2
    :server:check_idle=/usr/local/lib/filters/remote_check pr@laser1
    :lp=laser2%9100
    :tc=@commonlaser
```

8.15 Locations of Printcap Files

Options used:

- `printcap_path`=*printcap file locations*
- `lpd_printcap_path`=*additional server printcap file locations*

The `printcap_path` and `lpd_printcap_path` configuration options (see 11 (lpd.conf(5))) specify a set of paths for the printcap information. Client programs use only `printcap_path` and the `lpd` server uses both `printcap_path` and `lpd_printcap_path`. The path names can be separated with whitespace, commas, semicolons, or colons. The default values are:

```
printcap_path      /etc/printcap /usr/etc/printcap
lpd_printcap_path  /etc/lpd_printcap /usr/etc/lpd_printcap
```

8.15.1 Separate Server and Client Printcap Files

Since only the LPD server uses the printcap file specified by the `lpd_printcap_path`, you can place server specific information there. This allows you to have a common printcap file for clients and an additional one for the lpd servers.

8.15.2 ALL Printcap Entry

The `all` printcap name and `all` option is reserved to provide a list of printers available for use by the spooling software. This is a desperation, last ditch, back to the wall option for administrators with systems that do not have ways to provide a list of printcap entries. The 'all' printcap entry has the form:

```
all:all=pr1,pr2,...
```

The value of the `all` option should be a list of printcap names whose values will then be extracted.

8.16 Management Strategies for Large Installations

One very effective way to organize print spooling is to have a small number of print servers running a `lpd` daemon, and to have all the other systems send their jobs directly to them. By using the above methods of specifying the printer and server host you eliminate the need for more complex management strategies.

However, you still need to inform users of the names and existence of these printers, and how to contact them. One method is to use a common `/etc/printcap` file which is periodically updated and transfered to all sites. Another method is to distribute the information using the NIS or some other database. LPRng has provided a very flexible method of obtaining and distributing database information: see 8.17 (Using Programs To Get Printcap Information) for details.

8.17 Using Programs To Get Printcap Information

In the `lpd.conf` file you can specify:

```
printcap_path=|program
```

This will cause the LPRng software to execute the specified program, which should then provide the printcap information. The program is invoked with the standard filter options, and has the name of the printcap entry provided on `STDIN`. The filter should supply the printcap information on `stdout` and exit with a 0 (success) error code. By convention, the printcap name 'all' requests a printcap entry that lists all printers.

This technique has been used to interface to the Sun Microsystem NIS and NIS+ databases with great success. By having the invoked program a simple shell script or front end to the `nismatch` or `ypmatch` programs, the complexity of incorporating vendor specific code is avoided.

8.17.1 How to use NIS and LPRng

This note is based on material sent to the `lprng@lprng.org` mailing list by Paul Haldane <`paul@ucs.ed.ac.uk`>.

```
# From: Paul Haldane <paul@ucs.ed.ac.uk>
# To: lprng@lprng.org
# Subject: Re: Problem using plp with NIS
#
```

We generally don't use NIS for printcap files (we've moved to hesiod) but I can show you what we've done in the past.

The input to NIS is a normal printcap file:

```
# Classical printcap entry
lp23a|lp23|lp|main printhost printer - KB, EUCS front Door:\
        :lp=lp23a@printhost:\
        :sd=/usr/spool/lpr/lp23a:

#lprng printcap entry
lplabel|lpl|TEST - Labels printer:
        :lp=:rm=printhost:rp=lplabel:
        :sd=/usr/spool/lpr/lplabel:
        :rg=lpadm:mx#1:
```

To build the NIS printcap.byname map we add the following to the NIS makefile (along the other bits and pieces that the makefile needs to know about a new map).

```
PRINTCAP=$(DIR)/printcap
#PRINTCAP=/etc/printcap
# warning : [  ] is actually [<space><tab>] in the script
printcap.time: $(PRINTCAP) Makefile
  if [ -f $(PRINTCAP) ]; then \
    sed < $(PRINTCAP) \
      -e 's/[   ][  ]*$$//' -e '/\\$$/s/\\$$/ /' \
    | awk '$$1 ~ /^#/{next;} $$1 ~ /^[:|]/ {printf "%s", $$0; next;} \
        {printf "\n%s", $$0 }' \
    | sed -e 's/[   ]*:[  ]*:/:/g' -e 's/[  ]*|[  ]*/|/g' \
      -e '/^[   ]*$$/d' > .printcap.$$$$; \
    cat .printcap.$$$$; \
    if [ $$? = 0 -a -s .printcap.$$$$ ]; then \
      awk <.printcap.$$$$ '{ FS=":"; OFS="\t"; } { \
          n = split($$1, names, "|"); \
          for (i=1; i<=n; i++) \
              if (length(names[i]) > 0 \
              && names[i] !~ /[ \t]/) \
                  print names[i], $$0; \
      }' | $(MAKEDBM) - $(YPDBDIR)/$(DOM)/printcap.byname; \
      awk <.printcap.$$$$ '{ FS=":"; OFS="\t"; } { \
          n = split($$1, names, "|"); \
          if (n && length(names[1]) > 0 && names[1] !~ /[ \t]/) \
              print names[1], $$0; \
      }' | $(MAKEDBM) - $(YPDBDIR)/$(DOM)/printcap.bykey; \
      rm -f .printcap.$$$$; \
      touch printcap.time; echo "updated printcap"; \
    fi \
  fi
  @if [ ! $(NOPUSH) -a -f $(PRINTCAP) ]; then \
      $(YPPUSH) printcap.byname; \
      $(YPPUSH) printcap.bykey; \
      touch printcap.time; echo "pushed printcap"; \
  fi
```

To specify that you want YP database rather than file access, use the following entry in your `/etc/lpd.conf` file:

```
printcap_path |/usr/local/lib/pcfilter
```

Put the following shell script in /usr/local/lib/pcfilter

```
#!/bin/sh
#/usr/local/lib/pcfilter
read key
ypmatch "$key" printcap.byname
```

8.17.2 How to use NIS and LPRng - Sven Rudolph

```
Date: Wed, 11 Sep 1996 00:11:02 +0200
From: Sven Rudolph <sr1@os.inf.tu-dresden.de>
To: lprng@lprng.org
Subject: Using :oh=server: with NIS
```

When I use a cluster-wide printcap, two entries for each printer will appear, e. g.:

```
---------- start of /etc/printcap snippet
lp1
 :lp=lp1@server
lp2
 :lp=lp2@server
lp1
 :server:oh=servername
 :sd=/var/spool/lpd/lp1
 :lp=/dev/lp1
 :sh:mx#0
---------- end of /etc/printcap snippet
```

When I create a NIS map out of this, the printer name is used as a key and must be unique. So NIS' makedbm decides to drop all but the last entry for each printer. This makes the printer on the clients unavailable. I solved this by a hack where the second entry is called lp1.server and the NIS client script has to request the right entry.

1. Assumptions Perl is available at the YP server in /usr/bin/perl . A Bourne Shell is available at all clients in /bin/sh The printcap that is to be exported is in /etc/printcap . The printcap is written in the new format.

   In the examples the printer is called lp1 .

2. Add the following to your YP Makefile (/var/yp/Makefile) on the YP server :

   ```
   ---------- start of /var/yp/Makefile snippet
   PRINTCAP  = /etc/printcap
   printcap: $(PRINTCAP)
       @echo "Updating $@..."
       $(CAT) $(PRINTCAP) | \
           /usr/lib/yp/normalize_printcap | $(DBLOAD) -i $(PRINTCAP) \
           -o $(YPMAPDIR)/$@ - $@
       @if [ ! $(NOPUSH) ]; then $(YPPUSH) -d $(DOMAIN) $@; fi
       @if [ ! $(NOPUSH) ]; then echo "Pushed $@ map." ; fi
   ---------- end of /var/yp/Makefile snippet
   ```

   (These lines are for Debian GNU/Linux, other systems might require other modifications)

3. Install the programs `match_printcap` and `normalize_printcap` to `/usr/lib/yp`. normalize_printcap is only required on the YP server. The `normalize_printcap` processes only the LPRng printcap format.

```
---------- start of /usr/lib/yp/normalize_printcap
#! /usr/bin/perl
$debug = 0;
$line = "";
$new = "";
while (<>) {
    chomp;
    next if ( /^\s*\#.*/ );
    s/^\s*$//;
    next if ( $_ eq '' );
    print "new: " . $_ . "\n" if $debug;;
    if (/^\s/) { # continuation line
        $line = $line.$_;
        print "continued: $line\n" if $debug;
        next;
    } else {
        $line =~ s/\s+\:/:/g;
        $line =~ s/\:\s+/:/g;
        $line =~ s/\:\s*\:/:/g;
        print "line: $line\n" if $debug;
        push(@lines, $line) if $line;
        $line = $_;
    }
}
$line =~ s/\s+\:/:/g;
$line =~ s/\:\s+/:/g;
$line =~ s/\:\s*\:/:/g;
push(@lines,$line) if $line;
@lines = sort(@lines);
foreach $line (@lines) {
    ($printers) = split(/\:/,$line);
    @printers = split(/\|/,$printers);
    foreach $printer (@printers) {
      $num{$printer}++;
      push(@allprinters,$printer);
      print "allprinters: @allprinters\n" if $debug;
      print $printer."_".$num{$printer}."\t$line\n";
    }
}
@pr = keys %num;
print "printers @pr\n" if $debug;
if ($#allprinters >=0) {
    print "all_1\tall:all=".join(",",@pr)."\n";
}
---------- end of /usr/lib/yp/normalize_printcap
```

The result of processing the sample printcap file is:

```
lp1_1 lp1:lp=lp1@server
lp1_2 lp1:server:oh=servername:sd=/var/spool/lpd/lp1:lp=/dev/lp1:sh:mx#0
lp2_1 lp2:lp=lp2@server
all_1 all:all=lp1,lp2
```

Observe that each of the real printer entries has a key consisting of the printer name with a numerical suffix. This leads to the following method of extracting the printcap information using `ypmatch`:

```
---------- start of /usr/lib/yp/match_printcap
```

```
#!/bin/sh
read p
n=1
while ypmatch "${p}_${n}" printcap 2>/dev/null; do
    n=`expr $n + 1`
done
---------- end of /usr/lib/yp/match_printcap
```

4. Now test the YP arrangement:

```
$ cd /var/yp; make # this should create the printcap map
$ ypcat printcap # should provide the whole normalized printcap
$ echo lp1 |/usr/lib/yp/match_printcap # yields lp1 printcap
```

5. Add the printcap_path entry to `/etc/lpd.conf`:

```
printcap_path=|/usr/lib/yp/match_printcap
```

6. Test the use of the printcap path entry:

```
$ lpq -Plp1 # shows the status of lp1
```

7. Restart the lpd server and check to see that it accesses the right printcap information. Use the same `lpq` command, and then try `lpc printcap lp1`.

9 LPRng checkpc Utility

The `checkpc` (check printcap file) is one of the most useful utilities in the LPRng package.

It will read all the configuration files, printcap files and tests whether devices are set up correctly. Optionally, it will also set the permissions for spool directories and device files. Additionally, it will truncate the accounting and log files to a maximum size. Another use for `checkpc` is to remove old entries from queue directories.

For a new installation, you will want to run

```
checkpc -f -V
```

to set the permissions right. The `-f` flag instructs the program to correct file permissions. If you don't run this as `root`, you'll receive a warning about that fact and any **chown(2)** calls will (most likely) fail.

The program reports everything it changes. Since it isn't too clever about some things (visit the man page), you should keep an eye on the output, and run it again if needed. If it keeps failing, change the permissions yourself.

9.1 Maintenance

Later, you will want to use `checkpc` for the daily maintenance of your system. I have this line in user `lp`'s crontab:

```
32 5 * * * checkpc -t 10K -A3 -r >/dev/null 2>&1
```

This job will:

1. truncate all log and accounting files to 10KB (`-t 10K`). Actually, it will keep the last 10K from the file, starting on a complete line.

2. remove all stale files older than three days (`-A3 -r`).

I'm redirecting output to `/dev/null`, because `checkpc` is a little noisy to my taste. (But too noisy is better than too silent :)

9.2 Printcap Information

You can use `checkpc -V -P` to examine printcaps and tell you what they contain. This is identical to the `lpc server all` operation, but with a higher level of verbosity.

10 Spool Queues and Files

When files are accepted by the `lpd` server for printing, they are stored in a spool queue directory, together with other files controlling the print operation. This section describes these files and how the LPRng software uses them.

For descriptive purposes, we will use the following printcap entry as a guide:

```
pr|alias
  :sd=/var/lpd/pr_public
  :cd=/var/lpd/pr
```

10.1 Spool Queue

- `sd=`*Spool queue directory name*

The `sd` option in the printcap entry specifies the spool queue directory. If there is no `sd` entry or value, then the printer can only be used by the clients such as `lpr` to locate the destination for a print job. All information, files, etc., for a print queue is stored in the spool directory.

10.2 Queue Lock File

When the `lpd` server starts printing, it will fork individual worker processes to service each queue. To prevent multiple processes from working on the same queue, a printer lock file with the cannonical spool queue name is used. In our example, the lock file would be: `/var/lpd/pr/pr`.

The process ID of the currently active printer is stored in the lock file. By reading the lock file and testing to see if the process is still active, programs such as `lpq` can determine queue activity.

Similarly, the worker process may need to create other processes to assist it. These in turn will create lock or temporary files in the spool directory as well.

10.3 Spool Control File

The spool control file is used to control the operations of the spooler, and is in the spool or control directory. The file name has the form `control.`*`printer`*; in our example, the control file would be: `/var/lpd/pr/control.pr`.

The `lpc` program sends spool control requests to the `lpd` daemon, which updates the control file and then signals the appropriate spool server processes that an update has been performed. The control file contents have the form:

```
key value
```

The following keys and their values are currently supported.

```
printing_disabled   0 or 1              disable printing of jobs in queue
spooling_disabled   0 or 1              disable placing jobs in queue
holdall             0 or 1              hold jobs until released
redirect            printer             transfer jobs to indicated printer
class                   glob expression         print only jobs whose class matches glob
expression
server_order        printer name list   preferred order of printer use
debug               debugging options   debugging and tracing
```

The `printing_disabled` and `spooling_disabled` are managed using the `lpc start`, `lpc stop`, `lpc enable` and `lpc disable` commands. Similary, `holdall` is enabled and disabled by `holdall` and `noholdall` commands respectively. When holdall is enabled, jobs placed in the print queue will be held until they are explicitly released for printing by an `lpc release` command.

The `redirect` entry is used to redirect or transfer jobs which are spooled to this queue to another queue, and is managed by the redirect command. The `lpc redirect off` removes the redirect entry from the control file.

The `class` entry is similar in operation to the `holdall`, but allows jobs whose class identification matches the glob expression to be printed. This can be useful when you have special forms or paper required for a print job, and want to run only these jobs when the paper is in the printer.

The `server_order` entry is created and updated for a multiple printer queue. It records the order in which printers should next be used for normal print operations. This allows *round robin* use of printers, rather than having all jobs printed to the first printer in the list of printers.

The `debug` entry is set by the `lpc debug` command, and is used to enable or disable debugging and tracing information for a spool queue. This facility is for diagnostic purposes only.

10.4 Log and Status Files

- `create_files` *create log, accounting and status files*
- `lf=`*log file name (default: log)*
- `max_log_file_size#` *maximum log file size (Kbytes)*
- `min_log_file_size#` *minimum log file size (Kbytes)*
- `max_status_line#` *maximum status line length (characters)*
- `max_status_size#` *maximum status file size (Kbytes)*
- `min_status_size#` *minimum status file size (Kbytes)*
- `ps=`*filter status file name (default: status)*

During operation, the `lpd` server records the current printing operations in the `status.`*`printer`* file. For our example, this would be `/var/lpd/pr/status.pr`. In order to prevent this file from growing too large, the server will periodically truncate the file. You can force creation of these files by setting the `create_files` option. The `max_status_size` configuration or printcap option sets the maximum size (in Kbytes) of the status file; if the file exceeds this, only the last `min_status_size` bytes or 25% of the maximum size (default if not specified) will be preserved.

Similarly, the server logs its operations in the log file specified by the `lf` (log file) option (default is `lf=log`). The `max_log_file_size` value (default 0) specifies the maximum length of the log file in Kbytes. If this value is non-zero, then the log file is truncated to `min_log_file_size` bytes or 25% of the maximum file size. Again, the last portion of the log file is preserved. If the `max_log_file_size` value is 0, then the log file grows without limit.

Some filters require an additional filter status file that they use for recording additional filter status or other operational information. The `ps` names this file, and it is passed to a print filter using the `$s` option (see 13.6 (Filter Command Line Flags)).

The log file (`lf`) is opened and used as the STDERR output for filters and debugging information from the `lpd` server.

When reporting status information, the length of line returned can be a problem. The `max_status_line#79` option restricts the status line to a maximum of 79 characters.

10.5 Job Files

- `longnumber` *long job number*
- `default_priority=`*default job priority*

A print job consists of a control file and one or more data files. 6.6 (RFC1179) specifies the general format of these files and how they are to be transfered between servers. LPRng has extended the contents of the control files and the transfer protocol to provide a more powerful set of features, but has extensive provisions for backwards compatibility with non-LPRng software. A sample control file is shown below:

```
Hastart4.astart.com
J/tmp/file1 /tmp/file2
CA
Lpapowell
Ppapowell
fdfA002230astart4.astart.com
N/tmp/file1
UdfA002230astart4.astart.com
fdfB002230astart4.astart.com
N/tmp/file2
UdfB002230astart4.astart.com
```

The first part of the control file contains general information generated by the `lpr` or other spooling program. The information lines start with an uppercase letter or digit. Some other spooling systems also start information lines with various punctuation marks such as underscores (_) or periods (.).

Following this are a set of entries about each of the various files to be printed. These lines start with a lower case letter, followed by the print file name. The lower case letter is the *format* to be used to process the file. See 12.3 (print file formats) for more information about its use.

The entries marked with * are used only by LPRng. `N` and `U` lines are associated with a print file. The `N` line is the original name of the print file. The `U` line originally was used to indicate that the named file was to be unlinked after printing. This information is now ignored by LPRng. These lines are always grouped with a print file entry.

The names of control and data files follow a very strict pattern. Control files have the format `cfX`*number*`host`, where X is an upper case letter, *number* is (usually) a 3 digit number, and **host** is the host name. 6.6

Key	Meaning	Generated By
A	identifier *	LPRng internal
C	class	lpr -C class
D	date	lpr
H	originating host	lpr
I	indent	lpr -i indent
J	jobname	lpr -J jobname (default: list of files)
L	bnrname	lpr -U username
N	filename	(see text)
M	mailname	lpr -m mailname
P	logname	lpr
Q	queuename	lpr -Q
R	accntname	lpr -R accntname
S	slinkdata *	lpr
T	prtitle	lpr -T prtitle
U	unlnkfile	(see text)
W	width	lpr -w width
Z	zopts *	lpr -Z zopts
1	font1	lpr -1 font1
2	font2	lpr -2 font2
3	font3	lpr -3 font3
4	font4	lpr -4 font4

(RFC1179) restricted the total length of the control file name to 32 characters; LPRng has a much looser limit.

Data file names must follow the same pattern as the control file name, and have the format `dfX`*`number`*`host`. The X can be in the range A-Za-z, allowing at most 52 data files for a job. The *number* and **host** must be identical to the corresponding control file.

By convention, LPRng uses the X of the control file name to set a priority for the job. A job with control file name `cfA...` will have *lower* format than a job with format `cfB...`, and so forth. The `lpr` program uses the first letter of the class name or an explicit priority indication to set the letter value. If none of these are specified, then the `default_priority` value from the configuration or printcap entry is used.

The job number is usually a 3 digit value. However, in systems where a large number of jobs are spooled and need to be kept for printing at scheduled times, this can lead to problems. The `longnumber` option will use 6 digit job numbers. This must be used with care when operating with non-LPRng software.

10.6 Job Hold File

Associated with each control file is a hold file that has additional information controlling the printing operations. The entries in this file have the form:

```
key [value]
```

The following is an example of a hold file:

```
server 0
subserver 0
attempt 3
error cannot open printer
```

```
hold 0
priority 0
remove 0
routed 0
```

The `server` and `subserver` entry records the process ID of the server process and the subserver process that is printing the job. The `attempt` field records the total number of attempts to print the job. The `error` field records any error that would prevent the job from being printed. This information is reported by the `lpq` program.

The `hold` field is non-zero when the `lpc hold` command is used to explicitly prevent the job from being printed; `lpc release` will clear the field and allow the job to be printed.

The `priority` field is modified by the `lpc topq` command and is used to provide an overriding priority to printing the file.

The `remove` field is non-zero when the file has been printed and should be removed.

The `routed` field is used to indicate that there is routing information present in the hold file, and that special handling is needed. The routing information is provided by a 8.13 (routing filter). The information is recorded by information in the hold file. The following is an example of routing information. Normally this information is stored in a URL escaped format, with one line per destination, but for clarity this has been broken out into plain text form:

```
active 0
attempt 0
done 0
hold 0
priority 0
remove 0
routed 880892602
route 1
  dest t1
  ident papowell@astart4+705.1
  error
  copies 1
  copy_done 0
  status 0
  active 0
  attempt 0
  done 0
  hold 0
  sequence 0
  priority B
  CB
  end
route 2
  dest t1
  ident papowell@astart4+705.2
  error
  copies 0
  copy_done 0
  status 0
  active 0
  attempt 0
  done 0
```

```
hold 0
sequence 1
end
```

Routing information lines start with `route` followed by individual routing entry information. The `route dest`, `copies`, `priority`, and `Xnnnn` entries are derived from the output of the router program; other fields are used during the printing process. The `copy_done` records the numbers of copies done, while the `done` records that the entry has been completed. The `status` is the process ID of the server process doing the printing.

The output from route filter that generated the above file was:

```
dest t1
copies 1
priority B
CB
end
dest t1
end
```

10.7 Job State

Options used:

- `ah` *Automatically hold jobs*

A job can be in the following state:

1. Initial. This is the state during job submission. Jobs in the initial state do not have any status displayed for them.
2. Held. Once a job is submitted, it can either be printed or *held*. The `ah` printcap option specifies that all jobs are automatically held on submission. The `lpc release` and `lpc redo` command will cause these jobs to be printed and the `lprm` command can remove these jobs.
3. Active. The job is being processed for printing or transfer to another queue.
4. Pending. Jobs which can be printed but are not active. This can be due to the printer being busy or the job **class** not being printed.
5. Error. Jobs which have encountered an error during printing. The `lpc release` and `lpc redo` command will cause these jobs to be printed and the `lprm` command can remove these jobs.
6. Done. Jobs which have completed printing, but which are not yet removed from the print queue. See the `12.5 (save_when_done)` flag for more information. The `lprm` command can remove these jobs.

Normally the job sequences is initial, pending, active, and done. However, a job may be put in the error state by problems processing the job or by actions of the `lpc` command.

10.8 Job Identifier

Options:

- `use_identifier` *put job identifier in control file*

For each job in a spool queue, the LPRng software creates a unique identifier. This identifier is recorded in the control file `A` line. It can be used by the various client programs for identifying jobs, and is displayed by the `lpq` program as status information.

11 Configuration File lpd.conf and Options

The values in the LPRng configuration file (default: `/etc/lpd.conf`) override the compile time default values for printcap options. See the 21 (Index To All The Configuration and Printcap Options) for an index to a complete list of configuration options.

11.1 Configuration File Format

The LPRng distribution contains a template `lpd.conf` file which can be installed as `/etc/lpd.conf`. The configuration file has the following format:

```
# lpd.conf generated from  on Wed Apr  7 07:59:48 PDT 1999

#   The values in this file are the default values.
#   If you modify the file,  set the value to something other than the default
#   For example, '# default force_localhost' means the 'force_localhost' option
#    change this to 'force_localhost@' to have the opposite effect.

# Purpose: always print banner, ignore lpr -h option
#   default ab@
# change ---
ab
# Purpose: query accounting server when connected
#   default achk@
# Purpose: accounting at end (see also af, la, ar, as)
#   default ae=jobend $H $n $P $k $b $t
# Purpose: name of accounting file (see also la, ar)
#   default af=
...
```

The options are set exactly as for a printcap file, but do not require a leading colon (:).

To change the default value of an option, remove the comment character and edit the entry as shown in the above example.

To force the `lpd` server to use the new options, use the `lpc reread` command.

11.2 Configuration Information

In order to protect system security, the `/etc/lpd.conf` and `/etc/printcap` files should be read only.

For testing purposes when LPRng is compiled with the `-DGET_ENV` option LPRng uses the value of the `LPD_CONF` environment variable as the path to the `lpd.conf` file. **This is a security loophole, and should not be used when running SETUID ROOT or as ROOT.**

11.3 Configuration Options

The following variables are used to set default behavior for the LPRng software, or are commonly used for configuration of LPRng operations.

11.3.1 default_format

Default format for printing. Usually, `default_format=f`, but setting it to `default_format=l` will cause all files spooled by lpr to be spooled as binary files.

11.3.2 default_permission=ACCEPT

The default permissions to use when checking for printing or other permissions.

11.3.3 default_priority=A

The default priority for a print job.

11.3.4 default_remote_host=localhost

The default `lpd` server host.

11.3.5 default_tmp_dir=/tmp

The default directory for temporary files.

This option can be used to specify a default printer. If no value is given (default), the first printer in the printcap file will be used when no printer is specified.

11.3.6 domain_name=domain.name

You will only need to set this if LPRng can't determine your hosts domain name itself. This is usually a desperation option when DNS or some other database system is not available. The value of the $USER environment variable will be used as the return address.

11.4 The Record Queue Name qq and force_queuename flags

Options used:

- `qq` *Insert queue name into control file*
- `use_queuename` *(alias for qq)*
- `force_queuename=` *Queuename to be used*

The printcap information consists of the printer name and aliases; when a job is spooled to a printer alias, it is actually spooled to the main printer entry.

The `qq` use queuename option or its alias `use_queuename` tells LPRng to record the queue name that a job was queued to, and make it available to other software for processing. The `force_queuename=...` entry forces this name to be used. This capability has some interesting possibilities, as shown below.

```
pr1_landscape|pr1_portrait|pr_raw:lp=pr@host:qq
```

If a job is printed using `lpr -Ppr1_landscape`, then `pr1_landscape` will be recorded as the spool queue name by the LPRng software.

Later, when the job is processed by a filter, the filter will be invoked with a `-Qpr1_landscape` command line option. The filter can use the name of the queue to enable say, landscape, portrait, or raw orientations.

```
john|tom|frank:lp=pr@host:force_queuename=office
```

This printcap entry forces the queuename to be `office`; this information could be used by a central routing facility to process the information is a suitable manner.

11.5 The check_for_nonprintable Flag

Options used:

- `check_for_nonprintable` *LPR checks for non-printable file*
- `ml=` *minimum number of printable characters*

Normally, `lpr` checks an `f` format file for non-printable characters (i.e., escape characters) at the start of the print file. Disabling this check allows you to print executable files, etc., which can cause extreme abuse of your printer.

Disabling can be done on a single printcap basis, or you can do this on a global basis by modifying the configuration information (see 11 (lpd.conf)).

The `ml` value specifies the number of characters that are to be checked. Clearly, if it is 0, none will be checked.

11.6 The rg Restrict Use to Group Members Option

Options used:

- `rg=`*Restricted group list*

The `rg` value specifies a list of groups. If this value is present use of a printer or operation is restricted to only users in a particular group.

This was a wimpy attempt to do restrictions on print facilities. The -Ppr@host option overrides this check, unless the `rg` value is put in the LPRng defaults.

However, it does provide a simple tool to have clients do some form of permissions checking that only the `lpd` server could normally do.

11.7 The fx Allowed Formats Option

Options used:

- `fx=`*supported formats for printing*

The `fx` option restricts the formats supported by a spool queue. The `lpr` program uses these to check if a requested format is supported. By default, all formats are supported.

11.8 Fixing Bad Control Files and Metacharacters

Options used:

- `safe_chars=`*additional safe characters for control file*

RFC1179 defines a simple protocol and standard for print jobs to be interchanged between print spooling systems. Unfortunately, there were some major mistakes in not specifying the exact form that text would take when placed in the control file.

In addition, there are some simple coding errors that have been made, but due to their wide distribution in major vendors software, need to be accommodated. See 11.14.3 (reverse_lpq_format) for an example.

By default, LPRng will brutally convert a non-conforming RFC1179 control file into one that is acceptible to most, if not all, existing RFC1179 implementation.

In order to prevent problems with LPRng ruthlessly purges all characters but upper and lower case letters, spaces, tabs, and `-_.@/:()=,+-%` from the control file, replacing suspicious characters with '_'.

For some installations, the default set of safe characters may be overly restrictive. For example, *vintage* software may generate files with `#` characters in the `J` line of the control file. The replacement of this character may cause other things to stop working.

The `safe_chars` option allows the user to specify an additional set of safe characters in the `lpd.conf` configuration file(s). For example, `safe_chars=#"` would allow the `#` and `"` characters to appear in the control file.

In addition, LPRng will ruthlessly regenerate control file entries and data file names so that they are compliant with all known RFC1179 implementations.

11.9 Using the bk Option and Control File Filters

Options:

- `bk` *Berkeley compatible control file*
- `control_filter=`*Control file filter*

One of the more serious problems is when a print spooler (LPR) program does not generate print jobs in a manner compatible with a remote system.

While LPRng performs checks for improper implementations of RFC1179, it will try to accept a job even under the most severe abuse of the protocol. However, other spoolers are not so forgiving.

Some spoolers require that the contents of the control file be in **exactly** the order that the original 1988 BSD LPR software generated them. While some entries can be missing, all the entries present in the file must be in an explicit order.

The `bk` (Berkeley LPD compatible control file) option causes LPR and LPD to reformat the control file, removing objectionable entries. The control file of a job being sent to a remote printer will have its control file entries restricted to letters in (and the same order) as HPJCLIMWT1234.

However, there are some very odd commercial implementations that require *more* information than is present. To assist with this, the `control_filter` option can be used. This specifies a program that will process the control file before it is sent to a remote destination. See 13 (Filters) for details on filter operation, and 12.1 (Control Filters) for details on the point during job processing when the `control_filter` is used.

The `control_filter` program is run with the standard set of filter options. STDIN is attached to the control file and the STDOUT will be used as the control file value sent to the remote host.

The `control_filter` can rewrite the control file. The only restriction is that it cannot modify the **names** or delete datafiles. Here is a small snip of PERL code that shows how to rewrite the control file:

```
# you need to get PERL to do a 'dup' call on FD 0
$status = 0;
@cf_lines = <STDIN>;
# mess about with the control file
foreach $line (@cf_lines) {
   # or whatever you want
   print STDOUT $line;
}
exit $status;
```

The exit code of the `control_filter` is used to determine whether to proceed in processing. See 7 (Errorcodes) for details.

11.10 Maximum Copies

Options used:

- `mc#`*maximum copies*

The `mc` value specifies the maximum number of copies of a job that can be printed on a printer using the `lpr -Knn` or `lpr -#nn` option.

11.11 The minfree Minimum Spool Queue Space Option

Options used:

- `minfree=`*Size in Kbytes*

If this value is non-zero, then the `lpd` receiving server checks to see that there is the specified number of Kbytes of file space available before accepting a job.

11.12 Debugging

Options used:

- `debugging=`*debugging options*
- `full_time` *full or extended time format*
- `ms_time_resolution` *millisecond time resolution*
- `syslog_device=`*syslog alternative device*
- `use_date` *put date information in control file*
- `use_info_cache` *cache printcap and other information*

The LPRng software has a very powerful debugging capability. Since most printing problems occur on remote systems where it is impossible to run debuggers, and since most systems do not do core dumps of SETUID ROOT programs, the LPRng software provides a very verbose set of log file trace messages.

First, serious errors or other information are logged using the *syslog()* facilities. If these are not present on a system, then the messages are logged to the device specified by `syslog_device`.

For client programs, the debugging options are specified on the command line and output is directed to STDERR. For the `lpd` server, debugging commands can be specified on the command line OR as the `db=options` printcap value. Output is directed to the log file (`lf` option value, default log).

A typical debug entry has the format `2,network+1,database`. This sets the general debugging level to 2, network debugging to 1 and the database debugging level to the default. The following debugging options and levels are supported.

- nnn - general purpose debugging level
- network - network debugging
- database - database debugging
- receive - job or command reception debugging
- print - detailed job printing debugging

The `full_time` flag forces the logging and other information which has timestamps to have a full (year, month, day, etc.) timestamp. The `ms_time_resolution` flag forces millisecond time resolution in the time stamp. The `use_date` flag forces a date value to be placed in a control file if there is none.

The `use_info_cache` (default ON) causes `lpd` to cache printcap and configuration information. This is desirable except when trying to change values in printcap files and test the results. By using `use_info_cache@` in the configuration information, you can get immediate responses. Also, see 18.8 (lpc reread) for another method.

11.13 LPD Specific

Options used:

- `ipv6` *use IPV6 Network facilities*
- `lockfile=`*lpd server lock file*
- `report_server_as=`*server name for status reports*
- `spool_dir_perms#`*spool directory permissions*
- `spool_file_perms#`*spool file permissions*
- `spread_jobs#`*job number spread*

These options are usually LPD specific. For example, the `ipv6` specifies that the IPV6 protocol, rather than IPV4 will be used. In future versions, this may not be necessary.

The `lockfile` specifies the location of the lock file used by the `lpd` server.

The `spool_dir_perms` and `spool_file_perms` (default 0700 and 0600 respectively) values are the (numeric) permissions for the spool directory and spool files.

The `spread_jobs` option is obsolete. The `spread_jobs` option was a desperation fix to handle difficulties with the arrival of a large number of jobs with the same or close job number. The LPD server would fork children, each of whom tried to lock the job files. The spread value randomly chose a new number in the range about the original job number. However, it is still preserved for legacy systems which still have problems with file locking.

The `report_server_as` option allows an administrator to masquerade a server with another name. This could be useful if various load sharing activities are being carried out, or if there are problems reconfiguring DNS to cause the correct server name to be reported.

11.14 Legacy Compatibility

The following arguments have been provided for compatibility with legacy systems.

11.14.1 The allow_duplicate_args Option

Options used:

- `allow_duplicate_args` *allow lpr to have duplicate arguments*

Some users would like duplicate LPR and LPRM command line arguments to override earlier ones, i.e. - `lpr -a x -a y` should be equivalent to `lpr -a y`

The `allow_duplicate_args` option allows the various client programs to have duplicate arguments. The last specified argument on the command line will override previous values.

11.14.2 The class_in_status Options

Options used:

- `class_in_status` *show class name in status*

Setting the `class_in_status` option causes the class name rather than priority to be displayed in the status information.

11.14.3 The reverse_lpq_format Option

Options used:

- `reverse_lpq_format=` *reverse LPQ status format for specified remote systems*

Various Solaris and other System V implementations support an RFC1179 interface to remote printers. Unfortunately, there is a problem in that when they send a status request, the status format is reversed. That is, when LONG status format is wanted, they send SHORT, and vice versa.

The `reverse_lpq_format=` specifies a list of printers or IP addresses for which the `lpd` server will return LONG status when SHORT is requested, and vice versa. For example:

```
reverse_lpq_format=*.eng.com,130.192.0.0/16
```

will cause hosts whose Fully Qualified Domain Name (FQDN) ends in `eng.com` or from subnet `130.192.0.0` to have reversed status returned.

11.14.4 The return_short_status and short_status_length Options

Options used:

- `return_short_status=` *return short LPQ status for specified remote systems*
- `short_status_length#` *short LPQ status length in lines*

In order to be compatible with non-LPRng client programs, some administrators would like `lpd` to return a short or brief status to normal status queries.

The `return_short_status=` specifies a list of printers or IP addresses for which the `lpd` server will return an abbreviated status when LONG status is requested. For example:

```
return_short_status=*.eng.com,130.192.0.0/16
short_status_length#3
```

will cause hosts whose Fully Qualified Domain Name (FQDN) ends in `eng.com` or from subnet `130.192.0.0` to get only 3 lines of detailed status returned.

11.14.5 The force_lpq_status Options

Options used:

- `force_lpq_status=` *force LPQ status format for specified remote systems*

In order to be compatible with non-LPRng client programs which are totally unpredictable, this allows the administrator to specify the format for LPQ status when requests arrrive.

The `force_lpq_status=` specifies a list of formats and printers or IP addresses for which the `lpd` server will return status in the sepcified format. The entry has the format `KEY=list;KEY=list...` where `KEY` is `s` for short and `l` for long format, and list is a list of hosts or IP addresses. For example:

```
force_lpq_status=s=pc*.eng.com,130.192.12.0/24,l=sun*.eng.com
```

will cause hosts whose Fully Qualified Domain Name (FQDN) matches `pc*eng.com` or from subnet `130.192.12.0` to get short status returned and hosts which match `sun*.eng.com` get long status.

11.14.6 The ignore_requested_user_priority and force_fqdn_host Options

Options used:

- `ignore_requested_user_priority`*prevent users from queue jumping*
- `force_fqdn_hostname`*force FQDN hostname in control file*

Some students... um... users... will request a high priority for their job in order to jump the queue of waiting jobs. This option will cause the `lpd` server to ignore the requested user priority. However, the `topq` operation will still be effective.

Similarly, some print spoolers do not put a FQDN host name in their control file. The `force_fqdn_hostname` flag will cause `lpd` to put a FQDN host name in.

11.14.7 The lpr_bsd Options

This will force the `lpr -m` (send mail to user) option not to take an arguement, as in the BSD lpr.

12 Job Processing

Much of the flexibility of the LPRng software is obtained from the ability to control the details of each step of job processing. The following section details each step in the processing of a job, and explains the printcap options used to control each operation.

Assume the `pr` printcap entry has the form:

```
pr
    :lp=/dev/lp  OR  :lp=rp@rm
    :sd=/var/spool/lpd/pr
    :lf=log
    :of=/usr/local/bin/lpf
    :if=/usr/local/bin/lpf
```

Assume that we have used the following command to print a set of files.

```
lpr -Ppr file1 file2
```

This will create a control file in the `/var/spool/lpd/pr` directory with the following contents (this is an example - in practice there may be minor differences between the example and an actual control file):

```
Hastart4.astart.com
J/tmp/file1 /tmp/file2
CA
Lpapowell
Ppapowell
fdfA002230astart4.astart.com
N/tmp/file1
UdfA002230astart4.astart.com
fdfB002230astart4.astart.com
N/tmp/file2
UdfB002230astart4.astart.com
```

12.1 Opening the Output Device

Options used:

- `achk` *Accounting check at start*
- `af=`*Accounting File*
- `ar` *Remote printer accounting enabled*
- `as=`*Accounting at start*
- `connect_grace#` *Time between jobs*
- `connect_interval#` *Connection interval*
- `connect_timeout#` *Connection timeout*

Format	Meaning
`/pathname`	Absolute pathname of IO device
`pr@host`	transfer to `pr` on remote `host`
`host%port`	open a TCP/IP connection to port on host. host can be name or IP address
`|filter`	run the filter program; it STDIN will be used as device

- `control_filter=`*Control file filter*
- `ff` *form feed*
- `fo` *form feed on open*
- `la` *Local printer accounting enabled*
- `ld=`*leader on open (initialization string)*
- `lk` *Lock IO device*
- `lp=`*IO device pathname*
- `nb` *Nonblocking device open*
- `network_connect_grace#` *Time between jobs*
- `of=`*of filter*
- `retry_econnrefused#` *Retry if open failed*
- `retry_nolink#` *Retry if open failed*
- `rm` *the remote machine to send the job to*
- `rp` *the remote print queue to send the job to*
- `rw` *device opened RW flag*
- `server_tmp_dir=`*temporary directory*

Sequence of Operations:

1. During the server operations, it will try to create temporary files in the print queue spool directory. If this is not desirable, it will create them in the `server_tmp_dir` directory.
2. If the accounting file specified by `af` exists, it is opened (af_fd) and the af_fd is passed as file descriptor 3 to all filters. If the `af` value has the form `af=|/program` then the program is started and the program STDIN is used as af_fd. If the `af` value has the form `af=host%port`, then a TCP/IP connection to the corresponding port on the remote host is made and the port used as af_fd. In the latter two cases, the filter STDIN (file descriptor 0) is actually opened read/write, and is used when information is needed from the accounting filter or remote server. See 15.3 (Accounting Printcap Options) for more information on the LPRng accounting support.
3. If the `connect_grace` value is non-zero and the server is opening a device or `network_connect_grace` is non-zero and a network connection is being made, the server will pause the specified time. This is to accommodate devices which need a recovery time between jobs.
4. The `lp` option is checked to determine the type of IO device.

5. The IO device specified by `lp` is opened write-only or read-write if the `rw` flag is true, and the resulting file descriptor is io_fd. If the `nb` flag is set, a non-blocking open will be done as well. If the `lk` (lock device) flag is true, the device will be locked against use by other LPD servers.

6. If a `host%port` combination, a TCP/IP connection will be opened to the remote port and the connection will be used as io_fd.

7. If a filter program is specified, the filter program will be run and the STDIN of the filter will be used as the device file descriptor.

8. If a `rp@rm` combination, or none of the above combinations are true and the `rm` and `rp` values are non-zero, then the job will be transferred to a remote printer. The type of operation will be a job transfer, rather than printing operation.

9. If the `connect_timeout` value is non-zero, a timeout is setup for the device or socket open. If the device or connection open does not succeed within the timeout, then the open operation fails.

10. If a connection is to a network address (i.e. - `connect()` system call) and the connection attempt fails with an `ECONNREFUSED` error, if the `retry_econnrefused` flag is set then the connection attempt is retried, but this time using an alternative port number. See 18 (RFC1179) for details. This is repeated until all of the possible originating port numbers are exhausted.

11. If the open or connect operation fails, and the `retry_nolink` flag is set, then the server will pause for a minimum of `connect_grace` plus a multiple of `connect_interval` seconds based on the number of attempts before retrying the open operation. Note that the interval may increase as the number of attempts increases.

12. If printing a job and the `of` filter is specified, it is created with its STDOUT (fd 1) attached to the io_fd. Its stdin (of_fd) will be used in the steps listed below. If there is no `of` filter, then the of_fd value will be the io_fd descriptor.

13. If transferring a job and the `control_filter` option is specified, then the program specified by the `control_filter` value will be run. It will have its STDIN set to the control file, and its STDOUT output will be used as the new value of the control file to transfer to the remote host. See 13.6 (Filter Command Line Flags) for details of options passed to the control filter, and 7 (errorcodes) for the exit codes of the filter.

14. If `la` (local accounting) is true and we are printing a job or `ar` (remote accounting) is true and we are transferring a job, the `as` value is examined. If it is a filter (program) specification, then the program is started with its STDIN attached to `/dev/null` and STDOUT to the io_fd, STDERR to the error file, and file descriptor 3 to the accounting file descriptor af_fd. The lpd program will wait until it terminates, and examine the error code for action, as for the filters (see 7 (errorcodes) below). If it is a string, then it is interpreted, the escape sequences replaced with the appropriate information, and written to the accounting file.

15. If the `achk` (accounting check) flag is set, a line is read from the accounting filter af_fd file descriptor. This line should be `accept`, otherwise the job processing terminates with a JFAIL indication.

16. If the operation is a job transfer, the operation proceeds as outlined in 18 (RFC1179), and then the 12.5 (Normal Termination) operations are carried out.

17. If the operation is a print operation and the `ld` (leader on open) value is provided, the string is translated (escapes removed) and written to the of_fd file descriptor.

18. If the `fo` (form feed on open) flag is true, then the `ff` (form feed) string is translated (escapes removed) and written to the of_fd file descriptor.

12.2 Printing Banner At Beginning

Options used:

- `ab` *Always print banner (default FALSE)*
- `be=`*End banner generator program*
- `bl=`*Short banner line format*
- `bp=`*Banner generator program*
- `bs=`*Start banner generator*
- `hl` *Banner (header) Last*
- `of=`*Banner and File Separator Filter*
- `sb` *Short banner (default FALSE)*
- `sh` *Suppress header (banners) (default FALSE)*

Sequence of Operations:

1. If the `sh` (suppress header) flag is true, no banner is printed, and the actions in this section are skipped.
2. If the `hl` (header last) flag is true, the banner is printed at the end of the job, and the actions in this section are skipped.
3. If the user does not want banner pages and uses `lpr -h` then the `L` line in the control file will be missing. This is a signal not to print a banner. If there is no `L` line in the control file and `ab` (always print a banner) is false (the default), then no banner is printed. If there is no `L` line and `ab` is true, then ANONYMOUS is used.
4. LPRng generates banners by using either a banner printing program specified by `bp`, tt`bs`, or `be` options. For historical compatibility, if the `sb` flag is set, then the `bl` (banner line) contents are sent directly to the of_fd; By default the `bl` value is: `bl=$-'C:$-'n Job:  $-'J Date:  $-'t` (See 13.6 (Filter Command Line Flags) for details.) This will get translated to:
   `papowell:A Job:  file1 file2 Date:  Thu Nov 27 23:02:04 PST 1997`
5. If `bs` (start banner) program is specified, then it is used to generate a banner, otherwise if the `bp` (banner) program is specified, then it is used to generate a banner. If no program is available, we skip the banner generation. The banner generator program is started with the normal command line flags (see 13.6 (Filter Command Line Flags)), with its STDOUT attached to the of_fd descriptor. The short banner string described in the previous step is written to the STDIN. The banner printer is responsible for generating a banner appropriate to the printing device.

12.3 Printing Job Files

Options used:

- `Xf=`*Format Filter*
- `if=`*Default F Format Filter*
- `pr=`*pr formatting program*

- `send_job_rw_timeout`=*print job read/write timeout*
- `send_query_rw_timeout`=*status query operation read/write timeout*
- `sf` *Suppress FF Print File Separators*

Sequence of Operations: for each job in listed in the control file, the following operations are done in turn.

1. If there is an `of` filter present, the suspend string `\031\001` is written to of_fd and the no further action is taken until the of filter is suspended.
2. The control file line for the job is examined, and the first letter of the data file specification is used as the format.
3. If the format is `p`, the job is first processed by the program specified by the `pr` program, and the program output used as the print file.
4. If the format is `f`, `l`, or `p` then the `if` filter is used, otherwise the keyword `Xf` is used. Note that certain formats such as `p, a, l`, may not be used as formats.
5. The filter program is started with an appropriate set of command line options (see 13.6 (Filter Command Line Flags)), and with its STDOUT attached to the printing device (io_fd), STDERR to the log file (`lf`), and file descriptor 3 to the accounting fd af_fd.
6. When doing a read/write operation to a device or remote system, a timeout can be specified. When doing a print or job transfer operation, the `send_job_rw_timeout` value is used. When doing a status or query operation, the `send_query_rw_timeout` value is used. If a write or write operation does not complete within the specified timeout seconds, then we have an error condition and job processing or the query operation is terminated with JFAIL status. If the timeout value is 0, then no timeout is done.
7. `lpd` will then wait for the filter to exit. The exit status can be as follows:

```
Key      Value   Meaning
JSUCC    0       Successful
JFAIL    1, 32   Failed - retry later
JABORT   2, 33   Abort - terminate queue processing
JREMOVE  3, 34   Failed - remove job
JHOLD    6, 37   Failed - hold this job
Other            Abort - terminate queue processing
```

8. If the filter exit status was JSUCC (0), or no error indicated, then processing will continue otherwise the job termination takes (see 12.6 (Abnormal Termination)).
9. If the `of` filter is present, then it is reactivated with a `kill -CONT` signal.
10. If the `sf` (suppress FF print file separators) is false, then the `ff` (form feed) string will be interpreted and sent to the of_fd.

12.4 Printing Banner At End

Options used:

- `hl` *Header (Banner) Last*

The actions taken in this step are identical to those for the 12.2 (Printing Banner At Beginning), with the exception that the `be` (end banner program) is used to select the banner generation program rather than the `bs` (start banner program).

If we have `hl` true, then we print a banner at the end of the job, rather than start.

12.5 Normal Termination

Options used:

- `fq` *Form Feed on Close*
- `la` *Local Printer Accounting*
- `tr=`*Trailer on Close*
- `ae=`*Accounting at end*
- `save_when_done` *Save when done*

Sequence of Operations:

1. If we are printing and the `fq` flag is set and the `sf` (suppress interfile FF) flag is set, then the `ff` (form feed) string will be interpreted and sent to the of_fd.
2. If we are printing, the `tr` (trailer) string will be interpreted and sent to the of_fd.
3. If printing and the `la` (local printer accounting) flag is set or transferring a job and the `ar` (remote accounting) flag is set, the `ae` is examined and accounting is done as described for the `14 (as) field.`
4. If the `of` filter is present, its STDIN is closed, and the `lpd` server waits for it to exit. The exit status is used as described above.
5. The device (io_fd) is closed.
6. The job is marked as completed in the spool queue.
7. If the `save_when_done` flag is not specified, the job is removed.

12.6 Abnormal Termination

Options used:

- `mail_from=`*Mail from user name*
- `mail_operator_on_error=`*Mail to operator on error*
- `send_try#` *Maximum Print or Transfer Attempts*
- `save_on_error` *Do not delete on error*
- `send_failure_action=`*Action on Failure*
- `sendmail=`*sendmail path name and options*
- `stop_on_abort` *Stop processing queue on filter abort*

If the job processing terminates abnormally, the following sequence of events occurs:

1. The job is marked as having an error during processing.
2. The LPD server will attempt to kill all filters and other associated process by using a sequence of `kill -INT`, `kill -QUIT`, and finally `kill -KILL` operations.
3. If there is a `mail_operator_on_error` value, the specified operator will be mailed an error indication. The `sendmail` option specifies the pathname of the *sendmail* program and the options needed to have it read mail addresses from its standard input. For example, `sendmail=/usr/sbin/sendmail -oi -t` is a commonly used set of options.
4. The `mail_from` value specifies the user name used for mail origination. If not specified, the default is to use the print spool queue or printer name.
5. If there is a `send_failure_action` specified, then it is decoded and the corresponding action taken. If the value is `remove`, `hold`, `abort`, or `retry`, then the job is removed, held, aborted, or retried. If the value is `|/program`, the program is executed and the number of attempts are written to the filter STDIN. The exit status of the filter will be used to determine the consequent actions. That is, JSUCC (0) will be success, and the standard success action will be taken; JFAIL will cause retry, JREMOVE will cause the job to be removed, JHOLD will cause the job to be held, JABORT or other status will abort processing.
6. If the status is ABORT and the `stop_on_abort` flag is set, then further processing of jobs is terminated. The job is not removed from the queue.
7. If the error status indicates removal, and the `save_on_error` flag is clear then the job is removed from the spool queue.
8. If the error status indicates that no further operations should be performed on the queue, then the `lpd` server will stop processing jobs.
9. If the error code indicated that the job should be retried, and the `send_try` value is 0 or the number of attempts is less than the `send_try` value, then the job is retried. Between each attempt to transfer a job to a remote site. This pause will double after each attempt, reaching a maximum of `max_connect_interval` seconds. If `max_connect_interval` is 0, there is no limit on the interval value.

12.7 LPD Spool Queue Processing

Options used:

- `lpd_force_poll=`*Force LPD to periodically poll print queues*
- `lpd_poll_time#`*Time between polls*
- `max_servers_active#`*Maximum number of active servers*

When the `lpd` server starts, it will fork a set of subserver processes, each which will handle an individual queue.

If a system has a large number of queues, then this forking operation may result in the `lpd` server exhausting the process resources. To control this, the `max_servers_active` value restricts the number of active children to the specified value. If this value is 0, then 50% of the maximum system processes value will be used.

Due to the limits on the number of processes, there may be times when a job is placed in a queue, but the `lpd` server is unable to start handling the job. When all of the children of the main `lpd` server have exited, the server starts a timer. After `lpd_poll_time` seconds, it will scan the queues, looking for jobs to process, and starts a process to service them. If it does not find any jobs it remains idle.

The `lpd_force_poll` flag causes the server to periodically poll the queues. This is useful when there is a high possibility that jobs could fail to be printed due to high loads on the server.

13 Filters

This section gives an overview of how LPRng uses filter programs, and gives a detailed discussion of how the printcap options and filters interact.

13.1 What are filters?

Print filters are one of the most powerful tools in BSD-style printer systems.

In general UNIX terms, a *filter* is a program that takes its input file(s), does something with it, and sends the result to its standard output. Most UNIX utilities are designed as filters. (But since you are a system manager, you should already know that :))

In the context of a BSD-style print spooler (and also LPRng), the term *filter* refers to a program that processes file while it is being transferred to a printer.

The filter is executed with STDIN reading from the file to be printed STDOUT to the printer device or a temporary file. STDERR (file handle 2) is redirected to a log file, and file handle 3 to an accounting file.

A filter can be as simple as a `LF` to `CR/LF` translator, or it can incorporate a complete accounting system, automatic file type translations, or even redirect the job to another printing system.

As part of the LPRng project, the following filters are supported. The are many others available for use, but are currently not supported.

- 13.8 (lpf) a very simple CR/LF or passthrough filter.
- 13.9 (ifhp) HP PCL and PJL printer filters
-

The supported filters and other facilities are available from *ftp://ftp.astart.com/pub/LPRng*

13.2 What are print formats?

Options used:

- `if`, `cf`, `df`, `gf`, `nf`, `of`, `rf`, `tf`, `vf`, X`f`, *Filter programs*

LPRng has inherited a set of so-called '**print formats**' from its BSD ancestor. The format was used to specify the type of file that was being printed. The `lpd` server used the print format to select the filter for processing the file. The de**f**ault format is `f`.

The user can specify the format (i.e., the file type) by giving the appropriate option to `lpr`:

- `-b` or `-l`: Binary (literal) file. No processing should be done. The `l` format is recorded as the file format.
- `-c`: cifplot(1) output.
- `-d`: TeX DVI file.

- -g: Output from the plot(3X) routines.
- -n or -t: (di)troff output.
- -p: Text file that should be pre-processed by the pr command, and then by the standard text filter.
- -v: Benson Varian raster image.

Alternatively, one can also use -Fx, where x is the format specifier. (E.g., -Fc instead of -c.) This last form also allows you to use other (non-standard) format specifiers.

The filter for format X is the value for the Xf printcap option, with some minor exceptions. The following Xf options have a pre-defined meaning.

- if The f format filter, i.e. - for the default f format. All print jobs are passed through this one, unless another format is selected.
- cf Cifplot data filter (for -c format).
- df Filter for DVI files (-d).
- gf Graph data filter (-g).
- nf Ditroff data filter (-n).
- of This filter is used for processing the (optional) banner at the start and/or end of the print job, and also for the interjob separators. See 13.2 (of) filter for details.
- rf Filter for Fortran style files (-r).
- tf Troff filter (-t).
- vf (Versatek) raster image filter (-v).

13.3 OF Filter

The of filter is used to process banners and job separators. The of filter is responsible for performing appropriate processing of this information and sending to the printer for action.

While the various file filters are invoked on a once per print file basis, the of filter is invoked on a once per print job basis.

This filter is the first one to be started, and should perform whatever specialized device initialization is needed. It should also do whatever accounting procedure is desired for start of job accounting.

The of filter will be given any banner printing or job separation information for a job. As part of its operation, it can detect a specific string, corresponding to a banner print request, and generate a banner. (See the 12 (Job Processing Steps and Printcap Options) for details.)

During operation, the lpd server will send the special **stop** sequence of \031\001 to the of filter. The filter must then suspend itself using a kill -STOP operation. The lpd server will detect that the of filter has suspended itself and then will perform other printing operations.

After the other printing operations have been completed, the of will then be sent a kill -CONT signal.

This sequence will continue until all information has been printed, and then the of filter's STDIN will be closed. The filter will then perform whatever cleanup operations are needed, update accounting or other information, and exit.

13.4 LPR -p format

Options used:

- `pr`=*pr program for p format*

The `-p` format is requires filtering the the input files by the `pr` utility and then passing the result through the `if` filter.

This is widely regarded as a kludge and may not be supported on your print spooler.

13.5 LPR binary (-l) format

The binary (or literal) format is `-l`. The `if` filter is used to process the file, and is invoked with the `-c` (cancel processing?) flag.

The filter will not modify the file when sending it to the printer, but may apply various setups to the printer.

13.6 Filter Command Line Flags

Options used:

- `bk_filter_options`=*Backwards Compatible Filter options*
- `bk_of_filter_options`=*Backwards Compatible OF Filter options*
- `bkf` *Backwards Compatible Filters*
- `filter_ld_path`=*Filter LD_ LIBRARY_ PATH environment*
- `filter_options`=*Filter options*
- `filter_path`=*Filter PATH environment*
- `of_filter_options`=*OF Filter options*
- `pass_env`=*Environment variables to copy to Filter environment*
- `pl#`*line count for page*
- `pw#`*column count for page*
- `px#`*pixel width for page*
- `py#`*pixel length for page*

A filter (or program) specification in the LPRng printcap database usually has the form:

```
:option=| [flags] /path [arguments]
:option=[flags] /path [arguments]
```

The first case is used where the option value can be a string or filter, and the second where a program is always expected. The following procedure is used to run a filter program.

The sequence of operations to run a filter is as follows:

1. The program must be specified with an absolute path name.

Option	DefaultValue
`filter_options`	$C $F $H $J $L $P $Q $R $Z $a $c $d $e $f $h $i $j $k $l $n pr $s $w $x $y $-a
`of_filter_options`	(same as `filter_options`)
`bk_filter_options`	$P $w $l $x $y $F $c $L $i $J $C $0n $0h $-a
`bk_of_filter_options`	$w $l $x $y

2. By default, the program is run as the user if invoked from a client program such as `lpr`, `lpc`, etc. If invoked from `lpd`, it is run as the `server_user` user (default `daemon`) configuration entry.

3. The *flags* control how the program is to be run. The following flags are supported:

   - **ROOT** This opens a horrible security loophole, as it will run the program as ROOT. To enable this option, you must set various compilation flags, and perform other arcane operations. This is deliberately done to make administrators read the warnings and admonitions. The alternative to ROOT is to have a setuid ROOT executable. Under NO circumstances should you run a shell script setuid ROOT, with general execute permissions on it.
   - **-$** This very odd looking flag is used to suppress the addition of additional command line arguments specified by the value of `filter_options` to the program command line.

4. If the **-$** flag is not specified, the arguments determined by the value of the `bkf` (Berkeley LPD filter compatible flag) flag are added to the filter command line. If `bkf` is false the `filter_options` are added for OF filters and `of_filter_options` are added for non-OF filters; if it is true, then the `bk_filter_options` and `bk_of_filter_options` are added for OF and non-OF filters respectively.

5. By default, for programs that are not being invoked as print job file filters, the `filter_options` arguments are added. For print job filters, if the `bkf` flag is set, then the `bk_filter_options` and `bk_of_filter_options` entries are used. The default `bk` filter options are the same as originally used with the BSD LPR filters. For the `of` filter, either the `of_filter_options` or `bk_of_filter_options` arguments will be added.

6. The program arguments will then be scanned and interpreted. Arguments of the form `$`*letter* will be translated into values from the print job control file and/or printcap entry. The letters have the following meaning:

7. If there is no value for the specified argument, then the argument is removed from the list. If there is a value, the actual form of the substitution is controlled by additional flags as follows.

   Each entry in quotes is treated as a single value, as in /bin/sh. The `$'x` does not quote the value. Combinations of the various flags are allowed. For example, `$-x` would simply substitute the value for `x`, and then pass the whitespace separated components as individual arguments. This last form is useful for adding in additional flags on the command line.

8. The command line is parsed, metacharacters are ruthlessly stripped from all arguments and pathnames and replaced by `_` (underscores), and an argument list suitable for the `execve` system call is formed.

9. A sanitized environment is set up for the program execution, with the following environment variables.

10. If the filter is to be run by a client program such as `lpr`, then the environment variables specified by the `pass_env` configuration or printcap option will be extracted from the environment, have any metacharacters removed, and then placed in the environment variable list. Commonly, the `PGPPASS`, `PGPPASSFD`, and `PGPPATH` are specified.

11. The program is started, with STDIN, STDOUT, and STDERR attached to the appropriate files or file descriptors. If none is specified, then they are attached to `/dev/null`.

Letter	TranslatedValue
a	printcap `af` (accounting file name)
b	job size (in K bytes)
c	binary file (`l` format for print file)
d	printcap `cd` or `sd` entry
e	print job data file name (currently being processed)
f	print job original name when spooled for printing (N info from control file)
h	print job originating host (H info from control file)
i	indent request (I info from control file)
j	job number in spool queue
k	print job control file name
l	printcap `pl` (page length)
m	printcap `co`
n	user name (L info from control file)
p	remote printer (when processing for bounce queue)
r	remote host (when processing for bounce queue)
s	printcap `sf` (status file)
t	time in common UNIX format
w	printcap `pw` (page width)
x	printcap `px` (page x dimension)
y	printcap `py` (page y dimension)
F	print file format
P	printer name
S	printcap `cm` (comment field)
Capital letter	Corresponding line from control file
{key}	printcap value for `key`

Form	TranslatedValue
`$x`	`'-x`*value*`'`
`$-x`	`'`*value*`'`
`$0x`	`-x '`*value*`'`
`$'x`	`-x` *value*

`USER`	User name (client only)
`LOGNAME`	L control file info
`HOME`	Home directory (client only)
`LOGDIR`	Home directory (client only)
`PATH`	`filter_path` configuration information
`LD_LIBRARY_PATH`	`filter_ld_path` configuration information
`SHELL`	`/bin/sh`
`IFS`	`" \t"`
`TZ`	Time zone
`SPOOL_DIR`	`sd` printcap info
`CONTROL_DIR`	`cd` printcap info
`PRINTCAP_ENTRY`	printcap info
`CONTROL`	control file

13.7 LPRng Supported Filters

There already exists a large library of ready-to-use filters. Some of them have LPRng-specific versions, which can be found at the 1.8 (LPRng ftp mirror sites).

13.7.1 Filter Distribution Conventions

By convention, most filters are either totally standalone (very rare), or require a set of support files. There are two types of support files: per print queue configuration information and global support information.

Since a print filter will execute with the current directory set to the spool queue directory, most filters expect that per print queue configuration information should be kept in the spool directory. Most *vintage* filters insist on having these files *hidden* with names such as `.setup`. This can make it difficult for administrators to determine where the configuration files are.

It is strongly recommended that filters and information be placed in commonly accessible directories such as `/usr/local/lib/filters`, and the executables in subdirectories. This allows the LPRng administrator to set the privileges on these directories such that only the `lpd` process can access them.

Most of the LPRng supported filters can either be used as a `if` or `of` filter. The filter will examine the format type passed by the `-F`*`X`* command line argument, and if it is `o` it will perform as an `of` filter.

Alternatively, the filter will check the filename in the pathname by which is was invoked. If the name has the substring `of` in the filename, then it assumes it is to act as an `of` filter. This allows symbolic links to be made to a common filter executable, each of which corresponds to the filter name by which it is to be invoked.

When a filter is invoked, it is passed a large number of options, many of which are totally ignored in filter operation. However, for many purposes it is necessary to provide options to the filters to tailor their operation to the particular spool queue needs.

By convention, all LPRng supported filters use the

```
-Tkey=value[,key=value]
```

convention for specifying filter configuration option values.

13.8 lpf

Source code: 1.8 (LPRng Distribution)

This filter is distributed as part of the LPRng source code, and has a very limited functionality. By default, it only translates `\n` to `\r\n` sequences, and detects the OF Filter Stop sequence when invoked as an OF filter.

- Options:
  `-Tcrlf` - suppress `\n` to `\r\n` translation

13.9 IFHP Filter

Source code: 1.8 (LPRng Distribution, ifhp-version.tgz)

This filter supports a wide variety of *smart* printers, or to be more specific, printers which support PostScript, PCL or PJL languages.

13.9.1 Printer Capabilities

As explained in 3 (Setting Up Your Printer), you can have a parallel (unidirectional), serial (bidirectional), or network (bidirectional) connection. When using a bidirectional connection, you can sometime obtain or gratuitously receive error and/or status information from the printer.

Some printers will spontaneously generate error messages when printing a job on a bidirectional interface. Usually, though, it it necessary to force the printer to provide status in a reasonable format.

Some printers have the capability of printing either PCL or PostScript; some require special setup commands and some will *autosense* which type of job is being printed.

If you are printing text, and not using a Page Description Language like PostScript or PCL, then you may want to download a font to the printer. This is especially the case when you are trying to print text files in a non-English font.

Some printers will provide a *hardware* page counter value when requested; however, the means of requesting differ from model to model.

Sometimes you want to generate a special banner for a particular printer, and need to put in some dynamic information. While this can be done by the `lpd` server using the `bp` program specification, it turns out that non-LPRng systems which want to use the `ifhp` want to have the same facilities. Thus, you need to have some way to get the same effect as the `bp` option, but at the filter level.

Having done `lpd` banner generation and printing, why not have the filter run an accounting script as well?

At this point, I suspect that the reader is beginning to suspect that making a general purpose filter to support all of these possibilities is difficult. That is incorrect. It is **extremely** difficult.

However, the `ifhp` filter greatly simplifies this, as it uses a simple database together with some printer configuration options. For details, see the `ifhp` documentation for details on using the filter. For the terminally impatient, the following is a quick cookbook:

```
# network connection to jet direct box,
#   no banners, HP compatible
lp
  :lp=ipaddr%9100
  :/usr/local/lib/filters/ifhp
  :sh:sf
#
# banner added
#
lp
  :bp=/usr/local/lib/filters/pclbanner
  :of=/usr/local/lib/filters/ifhp
  :if=/usr/local/lib/filters/ifhp
  :sf:sb
#
# for a parallel port printer or when you want VERY fast
#  throughput, no pagecounts, error messages, etc.  The
#
lp
  :ifhp=status@
  :/usr/local/lib/filters/ifhp -Tstatus@
  :sh:sf
```

13.10 Using your own filters

If you already have a working setup, with its own specific filter programs, you might want to keep them. Or, you might want to write a set of your own.

See the source code in the 1.8 (LPRng Distribution, FILTERS_LPRng-<version>.tgz) files for examples.

14 Permissions and Authentication

The contents of the `/etc/lpd.perm` file is used to specify the restrictions on the use of the LPRng software, printers, and other facilities. The model used for permission granting is similar to packet filters. An incoming request is tested against a list of rules, and the first match found determines the action to be taken. The action is either `ACCEPT` or the request is granted, or `REJECT` and the request is denied. You can also establish a default action.

The following is a sample `lpd.perms` file.

```
# allow root on server to control jobs
ACCEPT SERVICE=C SERVER REMOTEUSER=root
REJECT SERVICE=C
#
# allow same user on originating host to remove a job
ACCEPT SERVICE=M SAMEHOST SAMEUSER
# allow root on server to remove a job
ACCEPT SERVICE=M SERVER REMOTEUSER=root
REJECT SERVICE=M
# all other operations allowed
DEFAULT ACCEPT
```

A rule will ACCEPT or REJECT a request if all of the patterns specified in the rule match. If there is a match failure, the next rule in sequence will be applied. If all of the rules are exhausted, then the last specified default authorization will be used.

The sense of a pattern match can be inverted using the NOT keyword. For example, the rules with `ACCEPT NOT USER=john,bill` succeeds only if USER is defined and the USER value is not `john` or `bill`.

The following patterns and matching are applied.

Most of the patterns can be lists of alternative values to match, and can even contain wild cards. The full details of the rules and keywords are detailed in the `lpd.conf(5)` man page.

14.1 Information for matching

In order to do matching, the `lpd` server obtains and sets up the following information:

1. If the request is coming over a network connection, then the IP address (REMOTEIP) port (PORT) of the source of the connection and FQDN of the remote host (REMOTEHOST) are obtained and the indicated values are set. To be specific, the IP address of the remote host is obtained using `getpeername()`. The `gethostbyaddr()` is used to look up the host's fully qualified domain name, which is then assigned to the REMOTEHOST value. The REMOTEIP value is the *set* or *list* of IP addresses that could be used by this host. This is possible in the IPV6 environment.

2. If the request contains the name of the user, then REMOTEUSER is assigned the name.

Keyword	Match
`DEFAULT`	default result
`SERVICE`	lpC Status and User, lpR, lprM, lpQ request
`USER`	user name in print job
`REMOTEUSER`	user making request
`HOST`	host name in print job
`REMOTEHOST`	host making request
`IP`	IP address and mask of host in print job
`REMOTEIP`	IP address and mask of host making request
`PORT`	TCP/IP port of host making request
`SAMEUSER`	USER and REMOTEUSER same
`SAMEHOST`	HOST and REMOTEHOST same
`SERVER`	request originates on lpd server
`FORWARD`	destination of job is not host
`GROUP`	USER is in the specified group
`LPC`	LPC command requested
`REMOTEGROUP`	REMOTEUSER is in the specified group
`CONTROLLINE`	match a line in control file
`AUTH`	authentication type
`AUTHUSER`	authenticated user
`AUTHFROM`	authenticated forwarder
`AUTHJOB`	authenticated job in queue

3. If the request contains the name of the printer, then PRINTER is assigned the name.
4. If a print job is being printed, then the USER, HOST, IP, and PRINTER are set to the user name, host, and printer information in the control file for the print job. To be specific, the HOST entry in the control file is used by `gethostbyname()` to get the fully qualified domain name of the host. The IP value is assigned a *set* or *list* of IP addresses that could be used by this host.
5. If one of the optional authentication methods is being used, (see 16 (Authentication and Encryption)), then AUTH is true and AUTHTYPE is set to the type of authentication used. AUTHUSER to the authenticated originating user of the request and AUTHFROM is when the originating program is a server. The AUTHSAMEUSER will be true when the remote client authentication information matches the authentication information used to create the job on the server. The AUTHJOB will be true when checking for job permissions and the job has been authenticated.

14.2 Permission Checks

When a connection is made to the `lpd` server, the originating site's IP address and hostname are determined, and a check with `SERVICE=X` is made. The REMOTEHOST, REMOTEIP, and PORT will be defined for the purposes of this check.

If the result is to accept the connection, then the request is then read from the connection, and the SERVICE, REMOTEUSER and PRINTER will be defined. A further check is performed to determine if the service request would be accepted.

When performing a service activity and a particular job is to be acted on, the USER, HOST, and other control file information will be available, and a further check can be performed.

If a rule is specified and the particular value is not defined, then a rule will fail to match.

Key	Request
C	LPC Control Request
M	LPRM Removal Request
P	Job Printing
Q	LPQ Status Request
R	LPR Job Transfer
S	LPC Status Request
X	Connection Request

14.3 Match Procedure

```
key=pattern                         substring match
key=pattern1,pattern2,pattern3,...  glob and exact
key=IP1/mask1,IP2/mask2,...         IP address
```

Each of the indicated values is matched against a list of patterns. The following types of matches are used:

1. substring match. The indicated entry is present as a substring in the pattern.
2. GLOB matches. The pattern is interpreted as a GLOB style pattern, where * matches 0 or more characters, and ? matches a single character, and `[L-H]` specifies a range of characters from `L` to `H`, in ASCII order.
3. IP address match. The address must be specified in the standard `nn.nn.nn.nn` format. The mask must be either an integer number corresponding to the number of significant bits, or in the standard `nn.nn.nn.nn` format. Addresses are compared by doing

   ```
   ( IPaddr XOR IP ) AND mask
   ```

   If the result is 0, then a match results. Note that there may be one or more addresses being checked for; this can occur when a host may have multiple IP addresses assigned to it.
4. integer range match. The pattern has the form `low-high`, where low and high are integer numbers. The match succeeds if the value is in the specified range.
5. Same IP Address Match. This compares two lists of IP addresses; a match is found when there is one or more common addresses.

14.3.1 DEFAULT

```
DEFAULT ACCEPT
DEFAULT REJECT
```

The DEFAULT rule specifies the default if no rule matches. Normally, there is one DEFAULT entry in a permissions file.

14.3.2 SERVICE

Match type: substring

The SERVICE key is based on the type of request.

Each of the above codes corresponds either directly to the user command, or a set of subcommands.

All the LPC subcommands are SERVICE=C; status commands such as `lpc status, printcap, active,` or `lpd` are SERVICE=S commands as well.

14.3.3 USER

Match type: GLOB

The USER information is taken from the `P` (person or logname) information in the print job control file.

14.3.4 REMOTEUSER

Match type: GLOB

The REMOTEUSER information is taken from the user information sent with a service request.

Note that one of the flaws of 6.6 (RFC1179) is that an LPQ (print status) request does not provide a REMOTEUSER name.

14.3.5 HOST

Match type: GLOB

The HOST information is taken from the `H` (host) information in the print job control file.

14.3.6 REMOTEHOST

Match type: GLOB

The REMOTEHOST information is obtained by doing a reverse IP name lookup on the remote host address. If there is no FQDN available, then the IP address in text form will be used.

14.3.7 PORT

Match type: integer range

The PORT value is obtained from the originating port of the TCP/IP connection. The match succeeds if it is in the specified range.

14.3.8 IP

Match type: IPaddr

The IP information is obtained by doing a DNS lookup on the H (host) information in the control file. If there is no host information, the IP address is undefined. Note that for a single host name there may be multiple IP addresses; address matches are performed against the list of addresses and succeeds if there is one or more individual address matches.

14.3.9 REMOTEIP

Match type: IPaddr

The REMOTEIP information is the IP address of the host making the service request. Note that the REMOTEIP value is obtained by using the `gethostbyaddr` lookup to obtain the DNS information for the remote host. This information may include multiple IP addresses; address matches are performed against the list of addresses and succeeds if there is one or more individual address matches.

14.3.10 LPC

Match type: GLOB

If you are doing an LPC command, this matches the command. This allows the following permissions line to be used:

```
#allow remoteuser admin on server to use LPC topq and hold
ACCEPT LPC=topq,hold SERVER REMOTEUSER=x
```

14.3.11 SAMEUSER

Match type: exact string match

Both the REMOTEUSER and USER information must be present and identical.

14.3.12 SAMEHOST

Match type: Same IP Address

The REMOTEHOST and HOST address lists are checked; if there is a common value the match succeeds.

14.3.13 SERVER

Match type: Same IP Address

One of the REMOTEHOST addresses must be the same as one of the addresses of the `lpd` server host, or must be one of the addresses found by looking up the `localhost` name using `gethostbyname()`.

14.3.14 FORWARD

Match type: Address Match

The list of REMOTEHOST and HOST addresses must not have a common entry. This is identical to NOT SAMEHOST. This is usually the case when a remote `lpd` server is forwarding jobs to the `lpd` server.

14.3.15 GROUP

Match type: modified GLOB

If the pattern does not start with a `@` character, then the USER information must be present and the USER must be present in one of the groups in `/etc/group` or whatever permissions mechanism is used to determine group ownership which matches the GLOB pattern.

If the pattern starts with a `@` character, then the USER information must be present and the user must be in the specified `netgroup`. This match will be performed only if the `netgroup` mechanism is supported on the system and the specified netgroup exists. No wildcard match will be done for netgroups.

14.3.16 REMOTEGROUP

The same rules as for GROUP, but using the REMOTEUSER value.

14.3.17 CONTROLLINE

Match type: GLOB

A `CONTROLLINE` pattern has the form

```
X=pattern1,pattern2,...
```

X is a single upper case letter. The corresponding line must be present in a control file, and the pattern is applied to the line contents.

This pattern can be used to select only files with specific control file information for printing.

14.3.18 AUTH

Match type: GLOB

The AUTH value can be NONE, indicating that no authentication was done. If authentication was done, then AUTH=USER checks to see if there was user information, and AUTH=FWD checks to see if there was forwarding system identification.

14.3.19 AUTHUSER

Match type: GLOB

If AUTH=USER check succeeds, the AUTHUSER rule will check to see if the user identification matches the pattern.

14.3.20 FWDUSER

Match type: GLOB

If AUTH=FWD check succeeds, the FWDUSER rule will check to see if the forwarding system identification matches the pattern.

14.3.21 IFIP

Match type: IPmatch, but for IPV6 as well as IPV4

There is a subtle problem with names and IP addresses which are obtained for 'multi-homed hosts', i.e. - those with multiple ethernet interfaces, and for IPV6 (IP Version 6), in which a host can have multiple addresses, and for the normal host which can have both a short name and a fully qualified domain name.

The IFIP (interface IP) field can be used to check the IP address of the origination of the request, as reported by the information returned by the accept() system call. Note that this information may be IPV4 or IPV6 information, depending on the origination of the system. This information is used by gethostbyaddr() to obtain the originating host fully qualified domain name (FQDN) and set of IP addresses. Note that this FQDN will be for the originating interface, and may not be the cannonical host name. Some systems which use the Domain Name Server (DNS) system may add the cannonical system name as an alias.

14.4 Permission File Location

Options used:

- `perms_path=`*directory path list*

The `perms_path=` configuration variable specifies the location of the default permissions file. The default value is:

```
perms_path=/etc/lpd.perms:/usr/etc/lpd.perms
```

The `lpd.perms` file can be obtained by running a program, in a similar manner to the `/etc/printcap` file. See 13 (Filters) for details on how the program would be invoked. For example, assume the configuration information specified:

```
perms_path=|/usr/local/libexec/get_perms
```

Then the `get_perms` program would be invoked with STDIN attached to `/dev/null` and the complete set of permission information would be read from its STDOUT.

14.5 Example Permission File

```
# allow root on server to control jobs
ACCEPT SERVICE=C SERVER REMOTEUSER=root
REJECT SERVICE=C
#
# allow same user on originating host to remove a job
ACCEPT SERVICE=M SAMEHOST SAMEUSER
# allow root on server to remove a job
ACCEPT SERVICE=M SERVER REMOTEUSER=root
REJECT SERVICE=M
# all other operations allowed
DEFAULT ACCEPT
```

In the above sample, we first specify that lp`C` commands from user `root` on the lpd server will be accepted. This is traditionally the way that most lpc commands operate.

Next, we reject any other lpc requests.

We accept lpr`M` requests from the host and user that submitted the job, as well as from root on the server, and reject any others.

Finally, all other types of commands (lpq, lpr) are allowed by default.

14.6 Complex Permission Checking

One of the more useful types of permission checking is to restrict access to your printers from users outside your networks. The IP pattern can specify a list of IP addresses and netmasks to apply to them.

For example `IP=10.3.4.0/24` would match all hosts with the IP addresses `IP=10.3.4.0` to `IP=10.3.4.255`.

Similarly, the HOST pattern can specify a set of hostnames or patterns to match against based on the GLOB notation.

For example `REMOTEHOST=*.astart.com` would match all hosts with a DNS entry which ended with `astart.com`.

The NOT keyword reverses the match sense. For example `REJECT NOT REMOTEHOST=*.astart.com,*.murpy.com` would reject all requests from hosts which did not have a DNS entry ending in `astart.com` or `murphy.com`.

14.7 More Examples

The following is a more complex lpd.perms file.

```
# All operations allowed except those specifically forbidden
DEFAULT ACCEPT
#Reject connections which do not originate from hosts with an
# address on 130.191.0.0 or from localhost,
# or name is not assigned to Engineering pc's
REJECT SERVICE=X NOT IFIP=130.191.0.0/16,127.0.0.1/32
REJECT SERVICE=X NOT REMOTEHOST=engpc*
#Do not allow anybody but root or papowell on
#astart1.astart.com or the server to use control
#facilities.
ACCEPT SERVICE=C SERVER REMOTEUSER=root
ACCEPT SERVICE=C REMOTEHOST=astart1.astart.com REMOTEUSER=papowell
#Allow root on talker.astart.com to control printer hpjet
ACCEPT SERVICE=C HOST=talker.astart.com PRINTER=hpjet REMOTEUSER=root
#Reject all others
REJECT SERVICE=C
#Do not allow forwarded jobs or requests
REJECT SERVICE=R,C,M FORWARD
# allow same user on originating host to remove a job
ACCEPT SERVICE=M SAMEHOST SAMEUSER
# allow root on server to remove a job
ACCEPT SERVICE=M SERVER REMOTEUSER=root
```

15 Accounting

The LPRng method for doing accounting is based on experiences in a Academic environment, where avoiding printing accounting procedures has long been practiced. While the LPRng procedures are not bombproof, they do provide a wide range of facilities, with various degrees of trust built into them.

15.1 Printer Accounting Reality Check

The following was written by Patrick Powell <papowell@astart.com> in response to the expressions of frustration that are periodically vented in the 1.2 (lprng@lprng.org) mailing list. While this addresses the use of a particular set of printer filters, i.e. - the 13.9 (ifhp) set, the comments are appropriate to other issues.

In Academic institutions, avoiding printing accounting has been regarded as a challenge, an ongoing game of fat cat and poor starving mouse, between the Administration and the downtrodden, poor, overcharged student. The following is a lighthearted ramble down the dark lane of printing accounting.

We will disregard the fact that if most students put as much effort into their studies as in finding ways to avoid accounting procedures then they would be Rhodes Scholar material, but I digress...

The accounting procedures put into the LPRng and the hpif filters may appear to be extraordinarily complex, but believe me, they are not. Firstly, we make the assumption that the printer has some sort of non-volatile page counter mechanism that is reliable and impervious to power on/off cycles. Without this mechanism the enterprising student ummm... user will simply turn off the printer. Software that prescans jobs for line counts and pages is notoriously unreliable, given even the most modest efforts of users to hide these procedures. The cost of running a PostScript simulator simply to do accounting has its flaws; without ensuring that the simulator has all of the interesting security loopholes closed, such as opening files, etc., it can become a trap door to hell for the system administrator.

Secondly, we must make the assumption that the student... uhhh... user will not be able to tinker with the page counter mechanism, i.e.- they will not be able to roll back the odometer on the printer, FOR THE DURATION OF A SINGLE JOB. I will digress and point out that a student actually did this for a challenge; it only took him a couple of weeks of study and a fully equipped microcontroller lab, and two (2) laser printers which he ruined in the experiment. HP was not amused when we sent them back under warranty, claiming that this our 'normal lab usage.'

Lastly, you should not mind a small amount of pilferage, or a few pages here and there being charged to the wrong account.

How Does It Work?

The `ifhp` filter records the page counter value at the start and end of each part of a print job. Each record has the form:

```
start -ppagecounter -Ff -kjob -uuser -hhost -R...
end  -ppages -qpagecounter -Ff -kjob -uuser -hhost -R...
```

When we use the OF filter and/or banners, we will see the individual jobs bracketed by the OF filter records:

```
start -p100 -Fo -kcfA100taco -uuser -hhost -R...
start -p101 -Ff -kcfA100taco -uuser -hhost -R...
end  -p1 -q102 -Ff -kcfA100taco -uuser -hhost -R...
start -p102 -Ff -kcfA100taco -uuser -hhost -R...
end  -p3 -q105 -Ff -kcfA100taco -uuser -hhost -R...
end  -p5 -q105 -Fo -kcfA100taco -uuser -hhost -R...
```

It should be clear from the above that all we need to do is to add up the values for the -Fo (OF) filter lines and we are done.

Unfortunately, this is too simplistic. If for some reason the job is killed or terminates due to error conditions, the OF filter may not get to finish its work. Thus, we may see the following:

```
start -p100 -Fo -kcfA100taco -uuser -hhost -R...
start -p101 -Ff -kcfA100taco -uuser -hhost -R...
start -p110 -Fo -kcfA101taco -uuser -hhost -R...
```

This is a clear indication that the user's job has been terminated. In this case we need to use the differences between pagecounters of the start records to do accounting.

There is a caveat to all of this; that is the problem of the last dead job in the list. If the last line in the accounting file is:

```
start -p110 -Fo -kcfA101taco -uuser -hhost -R...
```

is the last job finished or did it abort?

Who Used Up 2000 Pages of Paper Today?

Now we move on to the problem of real time accounting. Due to limited budgets, etc., many institutions would like to strictly enforce limits on paper use by students. As jobs are printed their accounts should be docked for the amount of paper use. One way to do this is to have an external accounting procedure update a shared database. The `ifhp` filter has provision for a shell script to be called at the end of print job; this is done by both the OF and IF filter. Thus, we can blithely assume that there is a central database carefully getting updates from the LPRng software, probably from dozens of different printers, and updating the accounting information.

The first question to be asked is simple: is this worth it? Perhaps doing accounting as a batch job once an hour/four times a day/once a day is cheaper than building an running such a database. If it costs $5K/year for the database software, you might just consider ignoring the 10,000 pages that get lost in the shuffle and use a simple set of awk/sed/perl scripts to update a database once an hour.

BAD JOBS - Who Do We Bill?

We inevitably run into an interesting question: what happens if a job does not complete correctly?

If you use the completion of the OF filter as a success status, I have to point out that many students... ummm... users soon find ways to send jobs to the printer that will cause it to lock up after their output has been printed. These jobs require power cycling of the printer and restarting the filter; a bit extreme, perhaps, but it has happened.

I suggest that you simply adopt a 'bill to last user of record' attitude, using the pagecount information as follows:

```
start OF -- starting point for THIS job
start IF --  nice information, but not useful
start IF --
end   OF -- ending point for this job - can record infomation
start OF --
if no end OF for previous job,  then treat as end OF and
  update accounting.
```

Now somebody is sure to complain that they got charged for a bunch of pages that they did not use. This is inevitable; always carry a can of oil for the squeaky wheels. I might make the observation that once is accident, twice is coincidence, but three times is malice; be wary of the constant complainer and check out not only him or her but also their co-workers.

How Do We Update the Database?

I suggest that database update be done as follows:

You maintain a 'last page reported' counter for each printer in the database. When a successful job reports in, check to see that
pagecount + joblength == newpagecount;

If this is not the case, then you have had a some unsuccessful jobs. In this case I strongly recommend that you have a means to request the accounting reporting program to go back through the accounting file and find the last report for the page counter value and try to backtrack through the accounting files. The accounting file is one of the first things to be attacked by students... Ummm... users. It should NOT be kept on an NFS exported or mounted file system. It should be carefully pruned and copied, perhaps on an hourly basis.

Now some administrators have fallen in love with network based printers; do not believe ANYTHING that comes over a network connection without some form of authentication; PGP has some very nice Public Key mechansims for handling this. This is a major weakness in using a database for keeping track of accounting - a weak authentication mechanism may lead to denial of service attacks by students flooding the database with bogus print usage reports; suddenly NOBODY can print and the administrator is driven to turning off accounting.

Good luck. I am never surprised when I encounter yet another wrinkle in this area.

Patrick ("You call me a Bean Counter? Guido, break this kid's fingers
with an adding machine!") Powell

15.2 How HP Printers Implement Page Counters

The following is from *http://www.hp.com/cposupport/printers/support_ doc/bpl02119.html*

HP LaserJet Printer Family - Page Count

Description Of The Page Count Feature On HP LaserJet 4 Family Printers

All HP LaserJet 4/5/6 family printers have a page count feature built into the firmware. However, this feature works differently depending on which HP LaserJet printer is being used. The following is a description of how the page count feature works for each printer within the HP LaserJet 4/5/6 printer families.

```
HP LaserJet 4/4M printers
HP LaserJet 4 Plus/4M Plus printers
HP LaserJet 4P/4MP printers
HP LaserJet 4Si/4Si MX printers
HP LaserJet 4ML printers
HP LaserJet 5P/5MP printers
HP LaserJet 6P/6MP printers
```

All of the above printers use the same method for keeping track of the number of copies. There are really two different page count values: Primary and Secondary values. Every time a page is printed, whether it is an internal job (such as a self-test) or a standard print job, the Secondary page count increases by one. This value is stored in standard RAM. Once the Secondary page count value reaches 10, the Primary page count will increase by 10. The Primary page count value is stored in a type of memory called NVRAM (Non-Volatile RAM). This is important, since NVRAM is not cleared when the printer is powered off. Standard RAM, on the other hand, is cleared when the printer is turned off or reset. Thus, the Primary page count only increases in increments of 10.

Example

You have a brand new HP LaserJet 6P printer and you print a self-test page. When you look on the test page for the Page Count value, you will see that it says 1. Next, you decide to print a two page letter and, after that, another self-test. The page count value now says 4. Internally, the printers Secondary page count (stored in RAM) has the value of 4 while the Primary page count (stored in NVRAM) still has the value of 0. Now, you turn the printer off, then back on, and print another self-test. The page count value again says 1 since the previous value of 4, stored in RAM, was cleared when the printer was powered off. Finally, print a ten page document and then turn the printer off. Upon turning the printer back on and printing out another self test, you see that the page count value is 11. Internally, the Secondary page count value is back at 1 while the Primary page count value (stored in NVRAM) is 10. Added together, you end up with the resulting value seen on the self-test page.

HP LaserJet 4L/5L/6L Printers

The reason that the page count method for the HP LaserJet 4L/5L/6L printers differ from that of the other printers is that the HP LaserJet 4L/5L/6L printers do not have any NVRAM available. Thus, no way exists for the printer to retain a page count value once the printer is powered off. The HP LaserJet 4L/5L/6L printers have only a single page count value that increases in increments of one until the printer is powered off. At that point, the page count value is reset and begins from 0 once again.

15.3 Accounting Printcap Options

The accounting facilities are controlled and enabled by the following entries in the printcap file. The default value is indicated.

Tag	Default Value	Purpose
af	NULL	accounting file name
as	"jobstart $H $n $P $k $b $t"	accounting info for job start
ae	"jobend $H $n $P $k $b $t"	accounting info for job end
accounting_server	NULL	
achk	FALSE	
la	TRUE	do accounting for 'local' printer
ar	FALSE	do accounting for 'remote' transfers

15.4 Accounting File

The most common method of accounting is to record the start and end times of a job and its size to the accounting file. A typical entry for the printcap defaults are shown below.

```
jobstart -H'taco.astart.com' -n'root' -P'ps' -k'cfA938taco.astart.com' \
-b'1093' -t'Nov  5 19:39:59'
start -p'12942' -k'cfA938taco.astart.com' -n'root' -h'taco.astart.com' -P'ps' \
-c'0' -F'o' -t'Sun Nov  5 19:39:25 1995'
start -p'12944' -k'cfA938taco.astart.com' -n'root' -h'taco.astart.com' -P'ps' \
-c'0' -F'f' -t'Sun Nov  5 19:39:27 1995'
end -p'12944' -k'cfA938taco.astart.com' -n'root' -h'taco.astart.com' -P'ps' \
-b'3' -c'0' -F'f' -t'Sun Nov  5 19:39:58 1995'
end -p'12942' -k'cfA938taco.astart.com' -n'root' -h'taco.astart.com' -P'ps' \
-b'2' -c'0' -F'o' -t'Sun Nov  5 19:39:59 1995'
jobend -H'taco.astart.com' -n'root' -P'ps' -k'cfA938taco.astart.com' \
-b'1093' -t'Nov  5 19:39:59'
```

The `jobstart` and `jobend` lines are added by the LPD server, as specified by the `as` and `ae` printcap options; the -b (byte count) indicates the numbers of bytes in the job.

The `start` and `end` lines are produced by the filters; the of filter has an -Fo, and the if filter a -Ff entry. The filters in the LPRng distribution produce the indicated output format by default. The -p value is the current value of a page counter device (if any), and the -b value indicates the total number of pages used.

It should be clear that a simple AWK or Perl script will be able to process an accounting file and update accounting information for accounting purposes; the usual problems with truncation, time stamps, etc., are left as an exercise for the system administrator.

Note that the accounting file must exist; LPRng will not create it (and also will not create the log file). This prevents accidentally growing log and accounting files.

15.5 Remote Server Accounting

To accommodate even more aggressive and centralized accounting, a method to make a connection to a print server and send information to the server has been provided as well. If `achk` option is set, it is assumed that the `af` entry specifies a connection to server on a remote host. The `lpd` server will send the `as` string to the server, and then wait for a single line of text from the remote server. If the first word on the return line is `accept` or `hold`, the job will be either accepted for printing or held. Any other value will cause the job to be deleted.

At the end of the job the `ae` string will be sent to the server. No response is expected. Example:

```
:af=accounting.site.com%2300,tcp
```

```
:achk
:as=starting
:ae=ending
```

The port that the connection originates from will be in the range set by the configuration or printcap `18.1 (originate_port)` option.

15.6 Using Filters For Accounting

Some sites have expressed interest in using a central accounting mechanism to check that users have permissions. This can be done by using the an alternative form of the as (accounting start) and ae (accounting end) printcap tags. If the as and ae are filter specifications, then a filter is invoked. If the as (accounting start) filter returns a non-zero exit status, then its value is used to handle the job as indicated by the 12.6 (Abnormal Termination) codes for filters. At the end of the job the :ae: filter will be invoked in a similar manner, but its exit status is ignored.

When using an accounting filter, the STDIN is attached (read/write) to the accounting file or remote host specified by the `af` printcap option, STDOUT to the output device, and STDERR to the log file. The filter program would be invoked with the default filter options.

For example, here is a sample entry to check and update accounting

```
printer
:as=|/usr/local/lib/filters/accounting.pl start
:ae=|/usr/local/lib/filters/accounting.pl end
```

15.7 Accounting Utility accounting.pl

In order to provide a framework for doing using the outlined accounting methods, the LPRng distribution `UTILS` directory has a `accounting.pl` script. This script does the following.

1. It is assumed that the accounting filter is invoked with the following printcap entry. The start and end is used by the filter to determine at which point in the accounting process it is invoked.

   ```
   printer
   :as=|/usr/local/lib/filters/accounting.pl start
   :ae=|/usr/local/lib/filters/accounting.pl end
   ```

2. It maintains the accounting file as a set of entries in the following format:

   ```
   START [job identification]
   start -pnn ...
   ...
   end -pnn+pagecount ...
   END -ppagecount [job identification]
   ```

3. Each time the filter is invoked with the `start` tags, it will add a `START` record to the end of the accounting file.

4. When it is invoked with the `end` option, it will update the accounting file and add an `END` entry.

5. It will handle aborted jobs by looking for jobs with have a `START` entry and a following `start` line and assuming that they progressed to the point of starting print operations, i.e. - the printer page counter was accessed and reported. It will then look for the next `START` entry with a following `start` line, and assume that the pages between the two points were used by the aborted job.

Administrators can use this script as a starting point for more advanced accounting. For example, rather than just recording the information, at the job start the script can query either a local database or a remote server to see if the user has permissions to access the printer. At the end of the job or when an `END` line is written to the accounting file, the local database or remote accounting server can be updated.

16 Authentication and Encryption

One of the major problems in a print spooler system is providing privacy and authentication services for users. One method is to construct a specific set of protocols which will be used for providing the privacy or authentication; another is to provide a simple interface to a set of tools that will do the authentication and/or encryption.

LPRng provides native support for the LPR extensions used by MIT and the Kerberos 4 implementation. In addition, it provides Kerberos 5 based authentication.

LPRng also supports the use of the PGP (Pretty Good Privacy) program and can sign and optionally encrypt command and reponses between servers and clients.

Finally, LPRng provide a general purpose interface allowing users to insert their own authentication methods, either at the program level or at the code level.

16.1 Authentication

A careful study of the authentication problem shows that it should be done during reception of commands and/or jobs from a remote user and/or spooler. At this time the following must be done:

1. The received command must be checked for consistency, and the remote user and host must be determined.
2. The remote user and host must be authenticated.
3. The command and/or spooling operation must be carried out.
4. The results must be returned to the remote system.

16.2 Identifiers

When a user logs into a system, they are assigned a user name and a corresponding UserID. This user name is used by the LPRng software when transferring jobs to identify the user.

When we look into the problem of authentication, we will possibly have a more global user identification to deal with, the authentication identifier (AuthID). One way to deal with this problem is to give LPRng intimate knowledge of the UserID and AuthID relationship. While this is possible, it may be difficult to deal with in a simple and extensible manner. An alternate solution is to provide a mapping service, where the authentication procedure provides a map between the UserID and AuthID.

16.3 RFC1179 Protocol Extensions

The RFC1179 protocol specifies that a LPD server command sent on a connection has the form:

```
\nnn[additional fields]\n
```

`\nnn` is a one octet (byte) value with the following meaning:

```
REQ_START   1    start printer
REQ_RECV    2    transfer a printer job
REQ_DSHORT  3    print short form of queue status
REQ_DLONG   4    print long form of queue status
REQ_REMOVE  5    remove jobs
```

The LPRng system extends the protocol with the following additional types:

```
REQ_CONTROL 6    do control operation
REQ_BLOCK   7    transfer a block format print job
REQ_SECURE  8    do operation with authentication
REQ_VERBOSE 9    verbose status information
REQ_LPSTAT 10    lpstat simulation
```

The REQ_CONTROL allows a remote user to send LPC commands to the server. The REQ_BLOCK provides an alternate method to transfer a job. Rather than transferring the control and data files individually, this format transfers one file. The REQ_AUTH provides a mechanism for providing an authentication mechanism and is described in this document.

16.4 Client Operations for Client To lpd Server Authentication

Options used:

- `auth`=*client to server authentication type*
- `auth_client_filter`=*client to server transfer program*
- `auth_forward`=*server to server authentication type*
- `auth_forward_filter`=*server to server transfer program*
- `auth_forward_id`=*Server identification*
- `auth_receive_filter`=*server to server transfer program*
- `auth_server_id`=*server identification*

This section describes the general purpose interface used for client to server authentication.

The LPRng client will generate a set of commands and place them in a file. The file is then encrypted and/or signed by the appropriate authentication method, and is transferred to the server. The server will then decrypt and/or check the signature, perform the requested actions, and in turn generate a file of status information. This file is encrypted and/or signed by the server, and sent to the client, where it is in turn decrypted and/or checked for correct signature. These activities are controlled by the following printcap or configuration options.

1. The `auth` option specifies the authentication type to be used for client to server transfers. For example, `auth=pgp` would specify PGP authentication, `auth=kerberos5` would specify Kerberos 5 authentication, `auth=kerberos4` would specify Kerberos 4 authentication, and `auth=user` would specify using user provided filters for authentication.

2. For client to server operations, the server id is the value of the `auth_server_id` option.

3. The `auth_forward`> option specifies the authentication type to be used for server to server transfers.

4. For server to server operations, the id of the originating server is specified by `auth_server_id`. and the remote server by `auth_forward_id`.
5. When doing client to server transfers, the originating user is determined by using the current UID of the program as the search value for `getpwuid()`.

The LPRng client will open a connection to the server and send a command with the following format:

```
\008printer C userid auth\n         - for commands
\008printer C userid auth jobsize\n - for print jobs
```

Note that \008 is a one byte code indicating an authenticated transfer. Printer is the spool queue name, C in the character 'C' indicating a client request, userid is the login id of the user, auth is the the value of the `auth` option, and jobsize is the size of the job file to be printed.

On reception of this command, the server will send a one byte success code. If an error is indicated by a non-zero response, additional error status may follow the non-zero success code byte. At the end of this information the connection will be terminated. The values used by LPRng are:

```
ACK_SUCCESS 0   success
ACK_STOP_Q  1   failed; no spooling to the remote queue
ACK_RETRY   2   failed; retry later
ACK_FAIL    3   failed; job rejected, no retry
```

If the success code is zero, client will use the `auth_client_filter` to encrypt and/or sign a data file to be transferred to the server. The authentication program will have the following IO assignments:

```
FD  Options Purpose
0  R/W     connection to remote host
1  W       output for returned status
2  W       errors
```

Command line arguments:

```
program -C -Pprinter -nuser -Aauth -Rauth_server_id -Ttempfile
```

The `tempfile` will contain either a command line as would be transferred using the standard RFC1179 protocol, or a print job in block format. See 18 (RFC1179 Protocol) for details. The client authenticator program will open and transfer the contents of `tempfile` to the server authenticator, using FD 0 and a format compatible with the underlying authentication mechanism.

If the transfer fails the client authenticator will log error information on FD 2 and then exit with error code JFAIL.

The server will send the client authentication program any error or logging information over the FD 0 connection, in a form appropriate to the authentication operation. The client authenticator will write this information to FD 1. If data transfer or authentication fails, the authenticator will write an error message to FD 2 and exit with error code JFAIL.

If no error has occured the client authenticator will then exit with error code JSUCC.

16.5 Server Operations for Client To lpd Server Authentication

Options used:

- auth_receive_filter=*Server (lpd) authentication program*
- auth_server_id=*Server identification*

When an authentication command arrives at the server, it has the following form:

```
\008printer C userid auth\n          - for commands
\008printer C userid auth jobsize\n - for print jobs
```

The server will attempt to find the printcap for the specified printer. For some operations this printer will be a dummy entry; this will simply cause the following operations to use the default information in the `lpd` configuration.

If a print job is being performed and the spool queue does not exist, then the job will be rejected. A non-zero error code will be written to the connection and the operation will terminate.

The `auth` value is used to set the AUTHTYPE permission checking value. If the AUTHTYPE is not built in, and the `auth` value does not match the printcap or configuration `auth` option value then authentication will fail. An error message will be logged to the server log file, and a non-zero error code and message will be written to the connection to the remote client program.

Many authentication programs require that the users provide some form of key or identification. The `auth_server_id` option is used for this purpose.

The server will start the server authenticator program and provide the following open file descriptors for it. The program will run as the same UID as the `lpd` server. If this is a print job transfer, the current directory will be the spool directory of the print queue.

```
FD  Options Purpose
0  R/W     socket connection to remote host (R/W)
1  W       pipe or file descriptor,  for information for server
2  W       error log
3  R       pipe or file descriptor,  for responses to client
```

Command line arguments:

```
program -S -PPRINTER -nUSER -aAUTHTYPE -Rauth_server_id -Ttempfile
```

The `PRINTER`, `USER`, and `AUTHTYPE`, are obtained from the original command.

The authentication filter will read the file transferred by the client authenticator, decrypt it, and place the decrypted values in the tempfile. It will then write a `from_id` string to FD 1, which will be read by the LPD server and used as the identification of the originating end of the connection.

If the originating program is an LPRng client, then the `from_id` value will be the user identification for the authentication protocol; if the originating program is an LPRng server, this value will be the server identificatio for the authentication protocol.

After writing this value, the transfer program will close FD 1. At this point the LPD server will use the contents of the tempfile to perform the various requested actions.

If the transfer step or authentication fails, then the server authenticator will write an error message to FD 2 and exit with error code JFAIL.

The `lpd` server will record the authentication information returned by the server in the AUTHUSER permissions key.

The `lpd` server will perform the usual permissions checks, with the addition of the indicated permission keys and associated values. During this process, any error messages or logging information normally returned to client programs will be written to the authentication program FD 3.

The lpd server will carry out either the commands or print job specified in the temporary file. During this process, any error messages or logging information normally returned to client programs will be written to the authentication program FD 3.

At the end of the operations, the FD 3 file descriptor will be closed and the lpd server will wait for the authentication process to exit.

The server authentication process will read input from FD 3 until the end of input, and then transfer the received information to the client side authenticator. It may use the tempfile to hold the information during the reading and transfer process.

If the transfer of the logging information fails, then the authenticator process will exit with error code JFAIL, otherwise it will exit with error code JSUCC.

16.6 lpd Server to Server Authentication

Options used:

- auth_forward=*Server to server authentication type*
- auth_forward_id=*Destination server authentication id*

The Server to Server authentication procedure is used by one server to forward jobs or commands to another server. It should be noted that this forwarding operation puts an implicit trust in the security of the client to server to server chain. The lpd server will perform an authenticated transfer to another server when it either needs to transfer a job to a remote printer or when it needs to propagate a lpq, lprm, or lprc operation.

The procedure used to by the server to send commands and/or jobs is identical to that used by a client, with the minor modification that the server is identified as the originating endpoint of the connection, and the client authentication information is transferred in the file.

When propagating a command, the server uses the authentication information provided for the remote user by the client to server authentication program. When propagating or forwarding a job, the server will use the authentication information stored in the job control or hold file. This information will be represented as AUTHUSER in the following discussion.

The auth_forward option value specifies the type of authentication to be used to forward authentication, and the sending server uses the auth_server_id as its identification, and the auth_forward_id as the identification of the remote server. If there is no user authentication information, then a normal, non-authenticated transfer will be done.

The auth_forward_filter will be used for the forwarding operation.

The sending server takes the part of the client, and will transfer a job acting similar to a client. The initial information transfer from the sending server will have the format:

```
\008printer F server_user authtype \n          - for commands
\008printer F server_user authtype controlfilename\n - for print jobs
```

The sending server will invoke its authenticator with the arguments:

```
auth_forward_filter -F -Pprinter -nserver_user -aauthtype \
-Rremote_user -Ttempfile
```

The tempfile containing the job or command information to be sent will have the form:

```
user_authentication_info\n
\n
<normal file contents>
```

That is, the user authentication information is place in the tempfile.

The tempfile will be transferred to the remote server in the same fashion as for a user job. Any error or logging information returned will either be written to the lpd log file or to the previous lpd process in the transfer chain.

On the destination server the same operations for receiving an authentication request from a client is performed. The AUTHUSER, AUTHFROM, and AUTHTYPE values will be the derived from the authentication request as for the client. The AUTHSAMEUSER will compare the remote client authentication information and the authentication information used to create the job.

When the remote server receives the authentication request, it will carry out the same actions as for a client to server transfer, modified as follows:

1. The lpd server will remove the first line of the transferred file, which contains the user authentication information, and set AUTHUSER to this value.
2. Authentication is performed using the indicated values.
3. If authentication succeeds, then the command line or print job control file is processed in the normal manner. This might now add more permissions values to tags, but the authentication information will not be changed.

16.7 Permission Checking

The following patterns and values can be used to check that a particular type of authentication has been used, and what the authenticated user information is.

- AUTH - authentication is used
- AUTHTYPE=globmatch
  This matches the type of authentication request. Built in values include kerberos4, kerberos5, and pgp.
- AUTHUSER=globmatch
  The originating auth_user_id value.
- AUTHFROM=globmatch
  When a command received from a server (i.e.- forwarded by a server), the value is the the forwarding servers authentication information, otherwise it is NULL.
- AUTHSAMEUSER
  The originating auth_user_id value is compared to the value used to create the job. If they are identical, the match succeeds.

For example, to reject non-authenticated operations, the following line could be put in the permissions file.

```
REJECT NOT AUTH
```

To reject server forwarded authentication as well, we use the following. Note that the ? forces a value to be present.

```
REJECT AUTH AUTHFROM=?*
```

If a remote server has id information FFEDBEEFDEAF, then the following will accept only forwarded jobs from this server. Note that AUTHFROM will only match on authenticated transfers; FWDUSER will only match on forwarded transfers.

```
ACCEPT AUTH AUTHFROM=FFEDBEEFDEAF
REJECT AUTH
REJECT NOT AUTH
```

To allow only authenticated users to remove jobs you can use:

```
ACCEPT AUTH SERVICE=R,M,L,P AUTHSAMEUSER
REJECT AUTH
REJECT NOT AUTH
```

16.8 Using PGP for Authentication

PGP is a well known encryption and authentication program. For more details see the web site *http://www.pgp.net* or the ftp site *ftp://ftp.pgp.net*.

LPRng has greatly simplified the use of PGP for authentication by building in support as follows.

The LPD server usually runs as user `daemon`, and opens files as `daemon`. The system administrator should establish a home directory for daemon, and use the PGP key generation facility to create a public and private key for the `daemon` user. By default, the PGP program puts the public and secret key rings in the `$HOME/.pgp/` directory, and sets them to be readable only by the user. You should log in temporarily as `daemon`, run the `pgp -kg` command, and then disable logins for `daemon`.

The user id chosen for the LPD server should be easily used to identify the server. For example, `lpr@hostname`, where hostname is the fully qualified domain name of the server is userful.

The next step is to place the passphrase in a file that is only readable by `daemon`, say `~daemon/.pgp/serverkey`, with owned by `daemon`, and with `600` permissions (read/write only by `daemon`). This is extremely important, as if any other users can read this file then security will be severely compromised.

The next step is to distribute the `lpr@hostname` public key to all users of the LPRng server, and to place the public keys of LPRng users in the `daemon` public key ring. This can be done using:

```
pgp -kxa userid destfile keyfile

Example:
> pgp -kxa lpr@astart /tmp/lprkey ~daemon/.pgp/pubring.pgp
Key for user ID: lpr@astart
512-bit key, key ID BB261B89, created 1999/01/01

Transport armor file: /tmp/lprkey.asc
Key extracted to file '/tmp/lprkey.asc'.
```

User can add the `lpr@astart` key to their public key rings using:

```
pgp -ka /tmp/lprkey.asc
```

Finally, the administrator will need to add users public keys to the `daemon` users public key ring. This can most easily be done by copying all the keys (in ASCII text form) to a single file (`/tmp/keyfile`)and using:

```
pgp -ka /tmp/keyfile ~daemon/.pgp/pubring.pgp
```

16.8.1 PGP Configuration

Options used:

- pgp_path=*path to PGP program*
- pgp_server_key=*path to server passphrase file*
- pgp_passphrase=*user passphrase file in PGPPATH or $HOME/.pgp*

The pgp_path option is the path to the PGP program.

The pgp_server_key is the path to the file containing the server passphrase. This file will be read by lpd to get the passphrase to unlock the server's keyring.

The LPRng client software will check to see if the pgp_passphrase value (default clientkey) file in $PGPPATH, and if not present, then in $HOME/.pgp. By default, the pass_env option is pass_env=PGPPASS,PGPPATH,PGPPASSFD and will pass the values of the PGPPATH, PGPPASS, and PGPPASSFD environment variables. See below for a method to use these.

Example printcap entry:

```
pr:
    :lp=pr@wayoff
    :auth=pgp
    :auth_server_id=lpr@wayoff.com
    :pgp_path=/usr/local/bin/pgp
    :pgp_passphrase=.mypass
```

One problem with using PGP is the need to have users input their passphrases. If the user is daring, then the pass phrase can be put in the file: ~/.pgp/clientkey . This file will be read by the LPRng client program the contents passed to PGP as the passphrase. This file MUST have 0400 permissions (read only by user) and MUST owned by the user.

A more subtle solution is to use the PGPPASSFD environment variable facility. This causes PGP to read the passphrase from a file descriptor. If the user puts his passphrase in a file, say $(HOME)/.pgp/.hidden, then the following shell script can be used.

```
#!/bin/sh
#  /usr/local/bin/pgplpr script - passphrase in $(HOME)/.pgp/.hidden
#
PGPASSFD=3 3<$(HOME)/.pgp/.hidden lpr "$@"
```

16.8.2 Permissions

If you wish to enforce the use of authentication, then you should modify the lpd.perms file. Here are some examples.

```
# force authentication
REJECT NO AUTH
REJECT NO AUTHTYPE=pgp
```

16.8.3 Client Configuration

One problem with using PGP is the need to have users input their passphrases. If the user is daring, then the pass phrase can be put in the file: ~/.pgp/clientkey . This file will be read by the LPRng client program the contents passed to PGP as the passphrase. This file MUST have 0400 permissions (read only by user) and MUST owned by the user.

A more subtle solution is to use the `PGPPASSFD` environment variable facility. This causes PGP to read the passphrase from a file descriptor. If the user puts his passphrase in a file, say `$(HOME)/.pgp/.hidden`, then the following shell script can be used.

```
#!/bin/sh
#  /usr/local/bin/pgplpr script - passphrase in $(HOME)/.pgp/.hidden
#
PGPASSFD=3 3<$(HOME)/.pgp/.hidden lpr "$@"
```

By using the -V (verbose) flag, users can see the results of the PGP interaction.

16.9 Using Kerberos 5 for Authentication

LPRng Kerberos 5 authentication is based on the Kerberos5-1.0.5 release as of March 28, 1999. This was obtained from MIT:

1. ftp to ATHENA-DIST.MIT.EDU (18.159.0.42), login anonymous, password your_email_address
2. Change into the directory '/pub/kerberos/
3. Get the README files and look at the details of using FTP to get the distribution. Note that there are also patches available which you might want to use.

Note that the distribution has only the most superficial documentation. There are no man pages for any of the support libraries, etc. etc.

16.9.1 Kerberos Installation Procedure

1. Get the Kerberos 5 distribution.
2. Compile and install the distribution.
3. Create the /etc/krb5.conf, /usr/local/var/krb5kdc/kdc.conf files using templates from the src/conf-files subdirectory. See the Installation notes and the System Administrators Guide.
4. Don't forget to create the /usr/local/var/krb5kdc/kdc.acl file; I did and it took me HOURS to figure out what was wrong...
5. Start up the KDC and KADMIN servers - you might want to put the following in your rc.local or equivalent file:

   ```
   if [ -f /etc/krb5.conf -a -f /usr/local/var/krb5kdc/kdc.conf  ]; then
       echo -n ' krb5kdc ';    /usr/local/sbin/krb5kdc;
       echo -n ' kadmind ';    /usr/local/sbin/kadmind;
   fi
   ```

6. use kadmin (or kadmin.local) to create principals for your users.
7. Now you need to create principals for the lprng servers. I have been using `lpr/hostname.REALM` as a template- i.e. `lpr/astart1.astart.com@ASTART.COM` for an example. Do this for all the servers. You should use fully qualified domain names for the principals.
8. Now you need to extract the keytab for each of the servers:

```
kadmin ...
ktadd -k file_for_host  lpr/hostname.REALM
```

The 'file_for_host' contains the keytab information, which is the equivalent information for the server.

9. Copy the 'file_for_host' to the server (you might want to encrypt or use a secure transfer for this). You need to put this in `/etc/lpd.keytab`. Make sure that this file is readable only by user `daemon`, as it will try to read the file to get its server key.

```
#> ls -l /etc/lpd.keytab
-rw-------  1 daemon  wheel  128 Jan 16 11:06 /etc/lpd.keytab
```

10. Modify (uncomment) the following entries in `/etc/lpd.conf`:

```
auth=kerberos5
kerberos_keytab=/etc/lpd.keytab
kerberos_service=lpr
# optional - explicit prinipal name for server, used by clients
#kerberos_server_principal=lpr/hostname@REALM
# optional - forwarding to another server with authentication
# kerberos_forward_principal=lpr/hostname@REALM
```

The `kerberos_keytab` entry is the location of the keytab file; kerberos_service is the service that will be used to generate a server principal name. This is the `lpr` used in the above key generation operations.

kerberos_life and kerberos_renew determine the lifetime and renewability of Kerberos tickets. The lifetime defaults to 10 hours, and the ticket will be refreshed when it expires if necessary.

11. You might like to check out the authentication using the sclient and sserver test programs. These link in the kerberos authentication and allow you to test it without all of LPD being involved.

```
cd LPRng/src; make sserver sclient
usage: sserver [-D] [-p port] [-s service] [-S keytab] file
   -D turns debugging on
   1. opens TCP   port 'port' (default 1234)
   2. waits for a connection
   3. when a connection comes in,  uses 'service' to get the principal
       name of the server,  and looks up the key in keytab file.
   4. Goes through the kerberos authentication.
   5. Copies the input from remote server to 'file'
   6. exits.
 usage: sclient [-D] [-p port] [-s service] host file
   -D turns debugging on
   1. opens a connection to port on host (i.e. - host%port)
   2. does the authentication.  You must have done kinit to get
       for your ticket to be valid.
   3. sends the file to remote host.
```

To test this, start up sserver on one host/window, then run sclient. The error messages are pretty straight forward, and when in doubt, look at the source code which has more than sufficient information.

16.9.2 Testing Transfers

Restart the server, and then try getting information using LPQ.

You can turn on tracing at LPQ to see if authentication is being used and is working:

```
lpq -Dnetwork,database
```

If the lpq works, then try send a job and see if the transfer is successful.

16.9.3 Explicit Server Principal Name

If you are using printers in different domains, then you can put the explicit principal name of the server in the printcap file, using the server_principal entry. For example:

```
lp_offsite
    :lp=printer@erehwon.org
    :auth=kerberos5
    :auth_server_id=lpr/erehwon.org@BLUESKY.ORG
```

16.10 Using Kerberos 4 for Authentication

LPRng has built-in support for the Project Athena extensions to the RFC1179 protocol. These provide an extremely simple authentication protocol using an initial credential exchange. After the initial exchange the usual RFC1179 protocol is used.

To enable Kerberos 4 support, you must modify the `LPRng/src/Makefile` and recompile the LPRng code. You should be aware that this is not a supported extension, and is provided as a courtesy to MIT and Project Athena.

17 Status Monitoring and Logging

Options used:

- `stalled_time#`*time after which to report a stalled active job*

The most commonly used tool for LPRng status is LPQ. However, the LPC command can be used, and you can also get real time logging of status to a remote host.

17.1 LPQ status reporting

The LPQ status display produced by LPRng has three formats.

17.2 LPQ Short Format (lpq -s)

This is one line per spool queue:

```
% lpq -sa
t1@astart110  (printing disabled) 1 job
t2@astart110  (routed/bounce to t1@astart110.astart.com) 0 jobs
t3@astart110  (forwarding to t3a@astart110.astart.com)
t3a@astart110  (forwarding to t2@astart110.astart.com)
t4@astart110  (subservers t5, t6)  0 jobs
t5@astart110  (serving t4) 0 jobs
t6@astart110  (serving t4) 0 jobs
```

Note that the name of the printer/host is first, followed by optional status information, followed by the number of jobs. Only printcap entries with spool queues have a jobs word in the last position. The `-a` option forces status for all queues or the queues in the `all` printcap entry to be returned.

The `stalled_time` (default 120 seconds) printcap option can be used to set a time after which active jobs will be reported as stalled.

17.3 LPQ Long Format (lpq default, lpq -l, lpq -L)

This is the default status display. It is a nicely formatted, extremely verbose format that is suitable for humble human interpretation. For example:

```
% lpq -a
Printer: t1@astart110  'Test Printer 1' (printing disabled)
 Queue: 1 printable job
 Server: no server active
 Status: finished operations at 09:44:00
 Rank   Owner/ID                   Class Job  Files               Size Time
1       papowell@astart110+202228663    A 10663 /tmp/hi              3 20:22:29
Printer: t2@astart110  'Test Printer 2' (routed/bounce to t1@astart110.astart.com)
 Queue: no printable jobs in queue
 Status: finished operations at 16:30:08
Printer: t3@astart110  (forwarding to t3a@astart110.astart.com)
Printer: t3a@astart110  (forwarding to t2@astart110.astart.com)
Printer: t4@astart110  (subservers t5, t6)
 Queue: no printable jobs in queue
 Status: finished operations at 09:44:06
Server Printer: t5@astart110  (serving t4)
 Queue: no printable jobs in queue
 Status: finished operations at 09:44:06
Server Printer: t6@astart110  (serving t4)
 Queue: no printable jobs in queue
 Status: finished operations at 09:10:00
```

The `lpq -l` (longer information) option causes more of the status information to be printed. You can use increasing numbers of `lpq -l` options (`lpq -ll` also works) to get more status. Use `lpq -L` for the maximum amount of status information.

17.4 LPQ Verbose Format (lpq -v)

This uses an extension to the RFC1179 protocol, and is supported only by LPRng. The amount of information displayed is the brutal, and in effect does a total database dump of the LPD. This has been developed in order to provide diagnostic and status information for databases that need to keep track of job progress through a spool queue.

```
% lpq -v
Printer: t1@astart110
 Comment: Test Printer 1
 Printing: no
 Spooling: yes
 Queue: 1 printable job
 Server: no server active
 Status: accounting at end 'papowell@astart110+094352860' at 09:44:00
```

```
 Status: printing 'papowell@astart110+094352860', closing device at 09:44:00
 Status: printing 'papowell@astart110+094352860', finished  at 09:44:00
 Status: subserver status 'JSUCC' for 'papowell@astart110+094352860' \
            on attempt 1 at 09:44:00
 Status: finished operations at 09:44:00
 Job: papowell@astart110+202228663 status= 1
 Job: papowell@astart110+202228663 CONTROL=
 - Hastart110.astart.com
 - Ppapowell
 - J/tmp/hi
 - CA
 - Lpapowell
 - Apapowell@astart110+202228663
 - Qt1
 - fdfA010663astart110.astart.com
 - N/tmp/hi
 - UdfA010663astart110.astart.com
 Job: papowell@astart110+202228663 HOLDFILE=
 - active 0
 - done 0
 - hold 0
 - move 0
....
```

17.5 lpc status

The LPC status command is used to show the status of the queues currently being managed by the LPRng server. Note that this form of the command is supported only by LPRng, and is not backwards compatible with BSD LPR implementations.

```
%lpc status all
 Printer           Printing Spooling Jobs  Server   Slave Redirect Status/Debug
lw4@astart4         enabled  enabled    0    none    none
lw5@astart4         enabled  enabled    0    none    none
```

The status display has a heading line and summary of the server status.

17.6 Remote Logger Operation

Several sites have wanted a way to provide central logging of job status and/or information. In order to do this, the following functionality is provided with LPRng.

17.6.1 Logger Network Communication

Options used:

- `logger_destination=`*logger information destination*
- `logger_pathname=`*pathname of temp file for log information*
- `logger_max_size=`*max size in K of temp file for log information*
- `logger_timeout=`*time between connection attempts*

The printcap/configuration variable `logger_destination` specifies a destination in the standard `host%port` notation used by LPRng. Host is the destination host, and can be a name or IP address. Port is the port on the destination host. A TCP/IP connection is made to the indicated port.

Log information is save in a temporary file specified by `logger_path`, and up to `logger_max_size` K bytes of data will be saved.

If a connection cannot be made to the `logger_destination`, then every `logger_timeout` seconds a new connection attempt will be made. If `logger_timeout` is 0, then a connection attempt will be made every time new data arrives to be logged.

17.6.2 Logger Messages

Log messages consist of a single line terminated with a newline (`\n`) character.

Each log message reports a system event or status change of the LPD server. When the connection is first established, a complete dump of the status of the LPD server is sent. After this, only status update messages are sent. The remote monitor can force a status dump by simply closing and reopening the connection.

17.6.3 Message Format

Each message is encoded as a URI escaped string. That is, non-alphanumeric characters are encoded as the 3 character sequence `%xx`, where `xx` is the hexadecimal value of the character. The message has the format `key=value`, where `key` indicates the message type. For example:

```
dump=host=astart4.astart.com%0aprinter=t1%0aprocess=1613%0aupdate_time=1999-03-2
  3-20:32:17.148%0avalue=queue=holdall 0%25250aprinting_aborted=0x0%25250aprinting
  _disabled=0x0%25250aspooling_disabled=0x0%25250a%250a%0a
```

The following keys are used:

1. dump
   A status dump of the current contents of a print queue.

Each message has a set of headers and a value. For example, the decoded dump message from the previous section would be:

```
host=astart4.astart.com
printer=t1
process=1613
update_time=1999-03-23-20:32:17.148
value=queue=holdall 0%250aprinting_aborted=0x0%250aprinting_disabled=0x0%250aspo
  oling_disabled=0x0%250a%0a
```

Each line consists of a key and a value. The `host` key indicates the host name, `printer` is the print queue, `process` is the process which generated the report or action, `update_time` is the time at which the report was generated, and `value` is the value of the report.

The decoded `value` of the above report is:

```
queue='holdall 0%0aprinting_aborted=0x0%0aprinting_disabled=0x0%0aspooling_dis
  abled=0x0%0a
```

The `queue` key provides the current value of the queue control file.

17.6.4 Dump Messages

Dump messages are generated at the start of operations, and consist of a list of queue status messages.

17.6.5 LPD Messages

These are used to indicate LPD startup or change in operation.

```
Decode: lpd=host=astart4.astart.com%0aprocess=1672%0aupdate_time=1999-03-23-20:5
1:10.507%0avalue=Starting%0a
host=astart4.astart.com
process=1672
update_time=1999-03-23-20:51:10.507
value=Starting
lpd: 'Starting'
```

17.6.6 Job Status Messages - UPDATE

Update messages are used to report changes in the queue contents, such as job arrival.

```
Decode: update=host=astart4.astart.com%0aidentifier=papowell@astart4+676%0anumbe
 ...
host=astart4.astart.com
identifier=papowell@astart4+676
number=676
printer=t1
process=1677
update_time=1999-03-23-20:51:17.197
value=bnrname=papowell%0acf_esc_image=Apapowell@astart4+676%250aCA%250aD1999-03-
  ...
```

This update message reports the arrival of a new job at the queue. The `value` field reports the control file contents:

```
cf_esc_image=Apapowell@astart4+676%0aCA%0aD1999-03-23-20:51:17.151%0aHastart4.as
   tart.com%0aJ/tmp/hi%0aLpapowell%0aPpapowell%0aQt1%0aN/tmp/hi%0afdfA676astart4.as
   tart.com%0aUdfA676astart4.astart.com%0a
class=A
date=1999-03-23-20:51:17.151
file_hostname=astart4.astart.com
fromhost=astart4.astart.com
held=0x0
hf_name=/var/tmp/LPD/t1/hfA676
hold_class=0x0
hold_time=0x0
identifier=papowell@astart4+676
job_time=0x36f86f45
jobname=/tmp/hi
logname=papowell
number=676
priority=A
queuename=t1
size=3
transfername=cfA676astart4.astart.com
```

```
update_time=1999-03-23-20:51:17.187
```

The `update_time` field in the section above is the time that the job information was last updated. The `cf_esc_image` value is the URL escaped control file information.

17.6.7 Printer Status Messages - PRSTATUS

These messages report printing or other activity related to a job.

```
Decode: prstatus=host=astart4.astart.com%0aidentifier=papowell@astart4+676%0anum
ber=676%0aprinter=t1%0aprocess=1692%0aupdate_time=1999-03-23-21:02:04.855%0avalu
e=finished 'papowell@astart4+676'%252c status 'JSUCC'%0a

host=astart4.astart.com
identifier=papowell@astart4+676
number=676
printer=t1
process=1692
update_time=1999-03-23-21:02:04.855
value=finished 'papowell@astart4+676'%2c status 'JSUCC'
PRSTATUS: 'finished 'papowell@astart4+676', status 'JSUCC''
```

17.7 LPR -mhost%port and user logging support

The `lpr -m` option is used to request that `lpd` send mail to the user when a job has completed. LPRng extends this to allow mail addresses of the form `host[%port][/(TCP|UPD)]` to request that logging information be sent to the user as well.

The administrator should be aware that this is a possible security loophole, and that the `allow_user_logging` flag must be enabled to allow this operation.

18 RFC 1179 - Line Printer Daemon Protocol

RFC1179 can be obtained from the LPRng distribution, in the LPRng_DOC/rfc1179 directory, or from one of many sites which mirror the RFCs.

This RFC is an *informational* RFC, which means that the information in it is meant as a guide to users, and not as a fixed standard. In addition, the RFC tried to document the behavior of the BSD LPD print server, and left out many details dealing with error recover, error messages, extensions to the protocol, etc.

In this section, I will try to explain what RFC1179 specifies as a protocol, and many of the problems encountered in trying to use it.

18.1 Ports and Connections

Options used:

- `lpd_port=`*Port for LPD to accept connection*
- `originate_port=`*Ports to originate connections on*
- `reuse_addr` *Set SO_REUSEADDR flag on connection*

- `retry_econnrefused` *Retry on connect ECONNREFUSED error*
- `retry_nolink` *Retry on device open or connection ffailure*
- `socket_linger#` *Linger time for sockets*

RFC1179 requires that the `lpd` server listen for TCP/IP connections on port 515. This port is registered with the Internet Naming Authority, and the `/etc/services` file or TCP/IP services database usually has an entry:

```
printer     515/tcp     spooler     # line printer spooler
```

RFC1179 explicitly states that all connections to port 515 must originate from ports 721-731. The reason for this restriction is due to the UNIX concept of *reserved* and *privileged* ports. By convention, ports in the range 1-1023 can only **bound** by processes whose Effective User ID (EUID) is 0 (root). This, ordinary users could not originate a connection from the reserved or privileged port range.

In a UNIX environment, this means that the user programs `lpr`, `lprm`, `lpq`, and `lpc` would have to be SETUID root.

As experience has shown, for security purposes, the fewer programs that need to have privileged status, the better. LPRng uses the `lpd_port=printer` configuration option to set the actual port to be use. By default, this is port 515, but can be set to other values.

The restriction of originating ports to 721-731 causes another set of problems. Part of the TCP/IP protocol is concerned with avoiding communications problems resulting from the arrival of old or *stale* packets. When a connection between `sourcehost, sourceport` and `desthost, destport` is made, a set of sequence numbers is established and used for sending and acknowledgement of data. When the connection terminates, the TCP/IP protocol restricts the establishment of a new connection between `sourcehost, sourceport` and `desthost, destport` for a period long enough for all *stale* packets to be removed from the system. This is approximately 10 minutes long.

In order to simplify assignments of ports, timing out connections, and other matters, many TCP/IP packages do keep track of explicit connections *originating* from a port, but simply prevent the port from being reused for either origination or reception of a connection. They do, however, keep track of the active connections **to** a port, and perform timeouts on these. This is usually much simpler to implement, as it can be done with a list attached to the port.

This implementation method creates some problems when a large number of connections must be originated from a relatively small number of port numbers. Observe what happens when host 1 tries to send a large number of jobs to a server 2. The following connections are established and terminated:
`host 1, port 721` and `host 2, port 515`
`host 1, port 722` and `host 2, port 515`
`host 1, port 723` and `host 2, port 515`
`host 1, port 724` and `host 2, port 515`
`host 1, port 725` and `host 2, port 515`
`host 1, port 726` and `host 2, port 515`
`host 1, port 727` and `host 2, port 515`
`host 1, port 728` and `host 2, port 515`
`host 1, port 729` and `host 2, port 515`
`host 1, port 730` and `host 2, port 515`
`host 1, port 731` and `host 2, port 515`

Now according to the RFC1179 rules and the TCP/IP protocol, we will have to wait until one of these connections terminates before we can make another. On the originating system, if the TCP/IP implementation does timeouts on the originating port, we will have to wait for the timeout to elapse before we can make a new connection. Unfortunately, there is no way to find out what the status of the port is, so we will have to try them each in turn until we get a successful connection.

The LPRng code has tried to provide several methods to deal with these problems. Firstly, the `originate_port=512 1023` option specifies the range of ports used to originate connections when the software is running either as ROOT or SETUID root. By strict RFC1179 rules, this should be `originate_port=721 731`, but it turns out that most BSD LPD based implementations only check for a *reserved* originating port. By using 512 ports we get a greatly reduced rate of errors due to lack of ports due to pending timeouts.

However, on some systems which are acting as servers for a large number of printers even increasing this port range is insufficient, and steps need to be taken use the originating port numbers more efficiently. The Berkeley TCP/IP implementation `getsockopt()` and `setsockopt()` allows the user to manipulate some of the underlying timeouts and options of the TCP/IP network. When a TCP/IP connection is established, the `setsockopt()` facility can be used to set the `SO_REUSEADDR` flag on the connection. This flag effectively sets the timeout value on the ports and connections to 0, allowing immediate reuse of the ports. When done on an originating end of a connection, this will allow the originating port number to be reused immediately.

It would appear that by setting `SO_REUSEADDR` on the originating end that we have solved our problems. However, unless the destination end of the connection sets its `SO_REUSEADDR` flag on the connection, it will still do a timeout. Thus when we try to make a connection from a port that was active within a short period of time to the same host, then it will reject the connection until the timeout is over.

The `reuse_addr` flag (default off) forces the LPRng software to set the `SO_REUSEADDR` flag on originating connections. As indicated, this will allow ports to be reused immediately for outgoing connections, rather than waiting for a timeout.

While the `reuse_addr` flag usually allows us to reuse ports, there is still the problem of dealing with connections failing due to the remote site rejecting the connection due to a pending timeout from a previous connection. A careful study of the original BSD TCP/IP network code and of some others indicates that when a connection fails due to a pending timeout, an ECONNREFUSED error code is returned to a `connect()` system call. If this happens and we suspect that the remote site is rejecting the connection due to a timeout problem, then we should retry making the connection but from a new port, and continue retrying until all possible ports are used.

The `retry_econnrefused` (default on) flag is used to specify that we retry connections in this manner. When this is set, a `connection refused` error causes the connection to be retried using a new port. This will be repeated until all available ports have been tried.

When printing a job and the `lpd` server connection to a remote site or device open fails, the `retry_nolink` (default on) will cause the attempt to be retried indefinately. The combination of `retry_econnrefused` and `retry_nolink` will provide robust connection attempts to remote systems.

While the above problems cause difficulties when making connections, there are also problems when terminating connections. After closing a socket, the TCP/IP software will try to flush any pending data to the destination. Unfortunately, on some systems it will only do this while the process is active. This has caused problems on systems which terminate a process it has received an abnormal (signal caused) termination.

The `setsockopt()` SO_LINGER option allows the user to specify that when a socket is closed normally, that the process should block until pending data is flushed or for the `socket_linger` period. If `socket_linger` is 0, then no SO_LINGER operation is done.

In summary, if you experience problems with connection failures due to port exhaustion, first try setting the `reuse_port` flag, and you should see a reduction. Check to ensure that the `retry_econnrefused` and `retry_nolink` flags are set, and the error code in the log and status files. If the failures continue, then the problem is caused by the remote end having timeout limitations and there is little you can do except to set a very long `connect_retry` interval, say `connect_retry=120` (2 minutes).

NNN	RFC1179	Operation	program
1	yes	start print	`lpc`
2	yes	transfer a printer job	`lpr`
3	yes	print short form of queue status	`lpq`
4	yes	print long form of queue status	`lpq`
5	yes	remove jobs	`lprm`
6	LPRng	do control operation	`lpc`
7	LPRng	transfer a block format print job	`lpr`
8	LPRng	secure command transfer	`lpc`
9	LPRng	verbose status information	`lpq`

18.2 Protocol Requests and Replies

Options used:

- `remote_support=`*Remote operations supported*

After a connection has been established, a request can be sent to the `lpd` server. The request consists of a single octet indicating the request type, followed by the printer (or print queue) name, followed by a set of options for the request, followed by a LF (line feed) character.

```
\NNNprinter[ options]\n
  NNN    Operation
```

After the request has been sent, then a reply will be returned. In general the reply has the following form:

```
\000\n    Success
\NNN\n    Failure (NNN is error code)
text\n    Text or status information
```

As can be seen, this protocol is extremely simple, but there are a set of problems due to the loosely written language of RFC1179.

1. Firstly, while RFC1179 sets limits on the lengths of commands, it does not strictly set limits on the characters set used in the commands. This can result in problems when trying to print status information, headers on banners, and other details.
2. The original RFC1179 protocol did not provide any way to do remote control of queues or LPD servers. This has been added to the protocol. As a side effect, if you try to use `lpc` to control a non-LPRng printer, it will not work.
3. You can specify that a network printer is non-LPRng by using the `remote_support=RQVMC` option and specify the operations supported by the printer. The letters R, Q, V, M, and C stand for `lpr`, `lpq`, `lpq -v` (verbose), verbose `lpq`, `lprm`, and `lpc` operations respectively, and indicate that these are supported. If `remote_support` does not allow a particular operation, then the LPRng software will not send a corresponding request to the printer. For example, `remote_support=R` would restrict operations to spooling jobs only, and the LPRng software would not query the printer for status.

18.3 Job Transfer

Options used:

Command	Purpose
\001\n	abort
\002nnnn cfname	control file transfer
\003nnnn dfname	data file transfer

Status	Purpose
\000	Accepted, proceed
\nnn	Rejected with error code

- `longnumber` *Long job number (6 digits)*
- `send_data_first` *Send data files first*
- `use_shorthost` *Use short hostname*

A job transfer operation starts with a job transfer request, followed by several file transfer operations. At the end of the file transfers, the connection should be closed.

A file transfer request has the form:

The abort operation is used to terminate job transfer and indicate that the job should not be processed for printing. The connection will be closed and the partly transferred job will be discarded.

The control file and data file transfer commands have a length (in bytes) of the file and the name of the file to be transferred. When the command is received, the server will reply with a status line:

The reply is only a single octet. Some defective implementations of RFC1179 send a LF after the octet, which makes life very difficult. LPRng makes an effort to detect these non-conforming RFC1179 systems and will accept jobs from them. However, it will not send jobs to them.

If LPRng sends a reject code, as an extension to RFC1179 it also sends an error message. Note that the values for error codes are not defined, nor are their causes. LPRng uses the following values for error codes, which appear to be compatible with many, but not all, of the BSD LPD based systems:

When the sender gets the reply indicating success, it sends the `nnnn` bytes of the control or data file, followed by a `\000` octet. The receiver will then reply as above; a single `\000` octet indicating success.

The above procedure is carried out until all data files and the control file of a job are transferred.

RFC1179 is silent on the following issues:

1. When sending a job, do you send the control file first, followed by the data file(s), or the data files first?
2. When sending multiple jobs, can you send them on a single connection, or do you have to establish a new connection for each job?

Code	Error
\000	Accepted, proceed
\001	Queue not accepting jobs
\002	Queue temporarily full, retry later
\003	Bad job format, do not retry

Command	Purpose
\003nnnn dfname	data file transfer

LPRng will *accept* jobs whether they are sent control or data files first. By default, it sends the control file first, followed by the data file. If the destination system requires that the data files be sent first, the `send_data_first` printcap option can be used to force data files to be sent first.

RFC1179 states that:

> The name of the control file ... should start with ASCII "cfA", followed by a three digit job number, followed by the host name which has constructed the control file.

The *should* in this wording indicates that this is simply a guideline, and that other formats are possible. Some of the major problems with this format are as follows:

1. The restriction to 3 digits means that at most 1000 jobs can be in a queue. Strangely, some systems generate far more than 1000 jobs a day, and need to archive them on a regular basis. The `longnumber` option will allow LPRng to use a 6 digit job number for files in the print queue.
2. The host name format is not specified. Some implementations consider that this is the short host name, while others think it is the fully qualified domain name (FQDN). LPRng, by default, will use the FQDN host name. However, the `use_shorthost` option will force it to use short host names in control and data files.
3. The `cfA` control file name was modified to allow the job priority to be used as the A letter of the control file. By default, this is A (lowest, i.e. `cfA`) and but can range to Z (highest, i.e. `cfZ`). All known spoolers except LPRng seem to ignore the actual value of the letter.

18.4 Data File Transfer

As discussed, a data file is transferred using the command below.

From RFC1179:

> The data file may contain any 8 bit values at all. The total number of bytes in the stream may be sent as the first operand, otherwise the field should be cleared to 0. The name of the data file should start with ASCII "dfA". This should be followed by a three digit job number. The job number should be followed by the host name which has constructed the data file. Interpretation of the contents of the data file is determined by the contents of the corresponding control file.

There are several surprises in RFC1179.

1. Apparently a job should only consist of a single data file. This is a severe limitation, and in fact the BSD LPR and other print spoolers process jobs with multiple data files. By convention, these data files have names of the form `dfA`, `dfB`, ... `dfZ`, `dfa`, `dfz`.
2. The RFC does not specify that the control file and data file job numbers must be identical. Most implementations follow this convention, which simplifies life tremendously.
3. The RFC does not specify that the control file and data file job host names must be identical. Most implementations follow this convention, which simplifies life tremendously.

4. A zero length data file does not cause a data transfer to take place. LPRng modifies this action to be slightly different. When a zero length data file transfer is indicated, all of the input until the connection is closed is used as the contents of the data file. When 'piping' into the `lpr` program, this can be very useful as it eliminates the need to create temporary files on the senders host. The `lpr -k` option for details. Note that some print spoolers do not use this interpretation, and this option should be used carefully.

18.5 Control File Contents

The control file consists of a set of lines which either provide printing information or specify data files to be printed. The information lines start with upper case letters or digits, while the data files lines start with lower case letters. Here is a sample control file:

```
Hastart4.astart.com
J(stdin)
CA
Lpapowell
Apapowell@astart4+955
Ppapowell
fdfA955astart4.astart.com
N(stdin)
UdfA955astart4.astart.com
```

The following are the letters and their meanings in the control file.

The `A` (Identifier) line was introduced to record a unique system wide job identifier for LPRng submitted jobs. This is basically formed from the user name, job number, and host at the time of submission. For example: `papowell@astart4+955` is job number 995 submitted by papowell from host astart4.

The `C` (Class) line is set by the `lpr -C class` option, and the value can be used to control printing. For example, the `lpc class zone` command would restrict job printing to only jobs with class `zone`.

The `H` (hostname), `P` (username), and `J` (jobname) fields are used to identify the host and user which sent the job, and to provide information to be displayed by `lpq` when reporting job status.

The `L` (print banner page) field is one that has caused many problems for users. RFC1179 indicates that its presence causes the banner page to be printed, and its absense suppresses banner pages. The `lpr -h` option suppresses putting this line into the control file. Usually the `L` field is a duplicate of the `P` field.

The `M` (mail information) field supplies a mail address for LPRng to send mail to when a job is completed. See 17.7 (LPR -m and user logging) for more details.

The `N` (file name) field is usually provided to identify the file name corresponding to the data file. This can be used to print names on page separators, etc. LPRng largely ignores this line.

The `I` (indent) and `W` (width) fields are supposed to specify a page indent and width for printing. These fields are passed to filters if they are present.

The `Q` (queue name) field is an LPRng extension, and contains the name of the print queue the job was originally sent to. See 11.4 (qq printcap option) for details.

The `R` (accounting info) field was added by LPRng to allow a specified account to be billed for job printing. The `lpr -Rname` option can be used to specify the accounting name.

The `S` (symbolic link) and `U` (unlink after printing) lines were used by the original BSD LPD print system to control how it passed files to the print server. LPRng ignores these lines. In fact, it will remove `S` lines and force the `U` lines to refer only to job data files. This closes a nasty security loophole on non-LPRng print spoolers.

X	RFC1179	Meaning
A	LPRng	Identifier for job
C	RFC1179	Class for banner page
H	RFC1179	Host name
I	RFC1179	Indent Printing
J	RFC1179	Job name for banner page
L	RFC1179	Print banner page
M	RFC1179	Mail When Printed
N	RFC1179	Name of source file
P	RFC1179	User identification
Q	LPRng	Queue name
R	LPRng	Accounting info
S	RFC1179	Symbolic link data
T	RFC1179	Title for pr
U	RFC1179	Unlink data file
W	RFC1179	Width of output
Z	LPRng	Filter options
1	RFC1179	troff R font
2	RFC1179	troff I font
3	RFC1179	troff B font
4	RFC1179	troff S font
c	RFC1179	Plot CIF file
d	RFC1179	Print DVI file
f	RFC1179	Print formatted file
g	RFC1179	Plot file
k	RFC1179	Reserved for use by Kerberized LPR clients and servers.
l	RFC1179	Print file leaving control characters
n	RFC1179	Print ditroff output file
o	RFC1179	Print Postscript output file
p	RFC1179	Print file with 'pr' format
t	RFC1179	Print troff output file
v	RFC1179	Print raster file
z	RFC1179	Reserved for future use with the Palladium print system.

The `T` (pr job title) is used with the `lpr -p` operation to supply a banner to the `pr` program.

The `Z` (filter options) value is specified with `lpr -Zoption` and is passed to the data file filters during the printing operation. See 13 (Filters) for details on how the this is used during the printing process.

All of the lower case letters are reserved for format specifications for data files. In the control file, these are followed by the name of the data file to which they correspond. While in principle different data files in the control file can have different formats, this has not been implemented in any known spooling system. See 13 (Filters) for details on how the data file formats are used during the printing process.

18.6 LPQ Requests

The RFC1179 protocol specifies that `lpq` print status requests can be sent to the `lpd` server. The lpq requests have the format:

```
\003printer [id]* \n    short
\004printer [id]* \n    long
\009printer [id]* \n    LPRng extension- verbose
```

The `lpd` print server will then return queue status and close the data connection.

RFC1179 does not state in any manner what the format of the queue status should be. Thus, implementors have been free to augment or change the status as they like. Even the BSD LPR status format has been changed from different versions.

See 17 (Status Monitoring and Logging) for information on the formats returned.

The `id` values are used to select the jobs to be displayed. LPRng displays any job whose ID, hostname, or user name information from the control file `A`, `H`, or `P` fields match any of the id values.

Note that since there is no identification of the information requestor, then restriction of information is almost impossible.

18.7 LPRM Requests

The RFC1179 protocol specifies that `lprm` job removal requests can be sent to the `lpd` server. The lpq requests have the format:

```
\005printer user [id]* \n
```

The `lpd` print server will search the specified print queue and remove any job whose ID, hostname, or user name information from the control file `A`, `H`, or `P` fields match any of the id values and for which the user has permission to perform a removal operation. See the 14 (/etc/lpd.perms) file for details on permissions.

Most RFC1179 compatible spoolers use the user information in the request as the name of the user which spooled the job. However, in a network environment this is extremely easy to fabricate, and is at best a weak type of authentication.

18.8 LPC Requests

LPRng has extended the RFC1179 protocol to allow queue and printer control commands to be sent to the LPD server. The format of these commands are:

```
\006printer user key [options]
```

Command	Operation
`active [printer[@host]]`	check to see if server accepting connections
`abort (printer[@host] | all)`	terminate server process printing job
`disable (printer[@host] | all)`	disable queueing
`debug (printer[@host] | all) debugparms`	set debug level for printer
`enable (printer[@host] | all)`	enable queueing
`hold (printer[@host] | all) (name[@host] | job | all)*`	hold job
`holdall (printer[@host] | all)`	hold all jobs on
`kill (printer[@host] | all)`	stop and restart server
`lpd [printer[@host]]`	get LPD PID for server
`lpq (printer[@host] | all) (name[@host] | job | all)*`	invoke LPQ
`lprm (printer[@host] | all) (name[@host]|host|job| all)*`	invoke LPRM
`move printer (user|jobid)* target`	move jobs to new queue
`noholdall (printer[@host] | all)`	hold all jobs off
`printcap (printer[@host] | all)`	report printcap values
`quit`	exit LPC
`redirect (printer[@host] | all) (printer@host | off )*`	redirect jobs
`release (printer[@host] | all) (name[@host] | job | all)*`	release job
`reread [printer[@host]]`	LPD reread database information
`start (printer[@host] | all)`	start printing
`status (printer[@host] | all)`	status of printers
`stop (printer[@host] | all)`	stop printing
`topq (printer[@host] | all) (name[@host] | job | all)*`	reorder job
`defaultq`	default queue for LPD server
`local (printer | all)`	client printcap and configuration information
`server (printer | all)`	server printcap and configuration information

The following commands are supported.

Many of these commands support extremely specialized operations for print queue management, However, the following are the most commonly used and are supported by the BSD LPD print spooling system as well:

- `start, stop, enable, disable`
  Start and stop will start and stop printing for a specified queue. Enable and disable enable and disable sending and/or accepting jobs for the queue.
- `abort, kill`
  Abort will cause the process doing the actual job printing to be terminated. Kill does an abort, and then restarts the printing process. These commands are used to restart a queue printing after some disaster.
- `topq` Places selected jobs at the top of the print queue.
- `status`
  Shows a status display of the print spools on the server.

The following commands are extensions to the basic set provided by the BSD LPD system.

- `lpq, lprm`
  Invokes the lpq or lprm program from lpc. Useful when in the interactive mode.
- `hold, holdall, release`
  The hold command will cause the selected jobs to be held until released. The holdall jobs sets all jobs submitted to the queue to be held until released. The release command releases jobs for printing. If a job has had an error and is in the error state, the release command will cause it to be reprinted.
- `move, redirect`
  The move command will move selected jobs to the specified spool queue. The redirect command sends all jobs submitted to the queue to be sent to the specified queue.
- `active, lpd, reread`
  The active command will connect to the server for the printer. This is used to check to see if non-LPRng print servers are active. The lpd command will connect to the server and get the process id (PID) of the `lpd` server. The reread command causes a SIGHUP signal to be sent to the lpd process, causing it to reread the `/etc/lpd.conf`, `/etc/printcap`, or `/etc/lpd.perms` files. This is usually done when some important configuration information has been modified and the administrator wants to have the server use the new information.
- `debug`
  This is a desperation facility for developers that allows dynamic enabling of debug information generation. Not normally used in general operation.
- `local, server`
  These commands will print out the configuration information in the local `/etc/lpd.conf` file, as well as the printcap information for the specified printers; `local` prints what the LPRng clients (`lpr, lpq, ...`) would use while `server` prints what the LPRng server (`lpd`) would use if running on this host. This is an extremely useful diagnostic tool for administrators. Not normally used in general operation.

18.9 Block Job Transfer

Options used:

- `send_block_format` *Transfer job as a block*

In normal job transfer operations, the sender and receiver have a handshake interaction in order to transfer a print job. Each file is sent individually. The `send_block_format` option forces a Block Job Transfer operation. This causes the sender to transfer a single file containing all the job printing information, including control file and data files.

The transfer command line has the form:

```
\006printer user@host size\n
```

The receiver will return any acknowledgement of a single 0 octet, and then the size bytes of the job will be transferred by the sender. At the end of the transfer a single 0 octet is added, and the receiver will indicate success by returning a single 0 octet. Any other value returned by the receiver indicates an error condition.

The file transferred by the sender is simply the command lines that it would have normally sent for job transfer, followed by the control or data file values.

18.10 Authenticated Transfer

RFC1179 does not provide any authentication or encryption mechanism for the transfer of jobs or commands to the `lpd` print server. The Authenticated Transfer operation was added to allow an encrypted or authenticated transfer of print jobs or commands.

Since there are various restrictions on the incorporation of authentication facilities into programs, LPRng supports authentication by providing a simple interface to encryption programs.

The idea is that when authentication is required when sending a job, LPRng will generate a block transfer job as described for the 18.9 (Block Transfer operation,) and then invoke a set of programs to encryt and transfer the file, and encrypt and transfer the returned status.

Similarly, when sending a command, the command information will be placed in a file and the encrypted file will be transferred.

This technique means that the programs and support to do encryption are external to LPRng, and can use any type of method that they choose to implement the secure and/or authenticated transfer.

See 16 (Authentication and Encryption) for details on the authentication interface.

19 Defective RFC1179 Implementations

Most printer (or print server box) manufacturers totally ignore the details of the RFC1179 protocol and simply accept the data files for printing, disregarding the control file **until they need to print a banner or provide status information**.

At this point, you suddenly discover all sorts of little details that cause horrible problems. For example, the use of non-ASCII characters (i.e. - values are 128-255) in the J (job) line of a control file has been known to crash one network interface card in such a manner that a power-up is needed to restart the printer.

Also, if you send one particular RFC1179 compatible print spooler a control file with a character whose value is 255 (i.e. 0xFF), the job will never get printed, and there is a mysterious diagnostic on the console:

```
unexpected end of input
```

This is due to the fact that the 0xFF eight bit value is getting sign extended to a 16 bit value 0xFFFF, which just turns out to be -1, or the error indication from a read.

19.1 OS2 Print Spoolers

For various reasons, some versions of the OS/2 `lpd` print spooler have decided to make the control file and data file names have different formats.

In addition, the OS/2 spooler does not follow RFC1179 correctly, and truncates the data and job file protocol exchange.

19.2 Serious Security Loophole

There is the subtle and nasty problem with some print filters that are not *meta-char-escape* proof. For example, suppose that a user decided to spool a job as follows:

```
lpr '-J; rm -rf /*;' /tmp/a
```

This would create a job file with the line:

```
J `rm /etc/passwd; echo Job;`
```

The job line ends up getting passed to a print filter:

```
pr:sd=/...
  :if=/usr/local/hack
    ... invoked as:
    /usr/local/hack '-J; rm -rf /*;'

/usr/local/hack is:

#!/bin/sh
while [ -n "$1" ] ; do
        case "$1" in
        -J  )  shift; args="$args -M$1";;
        esac;
        shift;
done;
# reformat the command line
eval /usr/local/realfilter $args
```

The observant reader will notice that the above line gets expanded to:

```
eval /usr/local/realfilter -MJ; rm -rf /*;
```

20 Acknowledgements

I'd like to thank the proof-readers from the LPRng mailing list, in particular: Gordon Haverland (`haverlan@agric.gov.ab.ca`), Lars Anderson, Bertrand Decouty, Horst Fickenscher, Philip Griffith, Gordon Haverland, John Perkins, Richard S. Shuford, James H. Young and the ones I forgot.

Finally, Patrick would like to thank all of the LPRng users who so relentlessly tried the incredible number of permutations and combiNATIONS Of printers and software, and made requests for *just one more feature*.

21 Index To All The Configuration and Printcap Options

`12.2 (ab)`	always print banner, ignore lpr -h option
`12.1 (achk)`	query accounting server when connected
`12.5 (ae)`	accounting at end (see also af, la, ar, as)
`12.1 (af)`	name of accounting file (see also la, ar)
`10.7 (ah)`	automatically hold all jobs
`11.14.1 (allow_duplicate_args)`	Allow duplicate command line arguments (legacy requirement)
`11.2 (allow_getenv)`	Allow use of LPD_CONF
`17.7 (allow_user_logging)`	allow users to request logging info using lpr -mhost%port
`12.1 (ar)`	enable remote transfer accounting (if af is set)
`12.1 (as)`	accounting at start (see also af, la, ar)
`16.4 (use_auth)`	authentication type to use
`16.4 (auth_client_filter)`	client to server authentication transfer program
`16.6 (auth_forward)`	server to server authentication method
`16.5 (auth_forward_filter)`	server to server authentication transfer program
`16.5 (auth_receive_filter)`	server receive authentication program
`16.4 (auth_server_id)`	server id for authentication
`12.2 (be)`	Banner at End Generation Program
`11.9 (bk)`	Berkeley LPD job file format
`13.6 (bk_filter_options)`	Berkeley LPD filter options
`13.6 (bk_of_filter_options)`	Berkeley LPD OF filter options
`13.6 (bkf)`	backwards-compatible filters: use simple paramters
`12.2 (bl)`	short banner line sent to banner printer
`12.2 (bp)`	Banner Generation Program (see bs, be)
`8.10 (bq)`	Use filters on bounce queue jobs
`8.10 (bq_format)`	Format of bounce queue output
`8.8 (br)`	Serial port bit rate (see ty)
`12.2 (bs)`	Banner at Start Generation Program
`10 (cd)`	control directory
`11.5 (check_for_nonprintable)`	LPR checks for nonprintable file
`8.14.2 (check_idle)`	program used to check for idle printer
`11.14.2 (class_in_status)`	Show job class name in `lpq` status information
`8.2 (cm)`	comment identifying printer (LPQ)
`11 (config_file)`	configuration file
`12.1 (connect_grace)`	connection control for remote printers
`12.1 (connect_interval)`	connection control for remote printers
`12.1 (connect_timeout)`	connection control for remote printers
`12.1 (connect_try)`	connection control for remote printers
`11.9 (control_filter)`	control file filter
`11.12 (db)`	debug options for queue
`11.3.1 (default_format)`	default job format
`11.3.2 (default_permission)`	default permission for files
`7.1 (default_printer)`	default printer
`7.1 (default_printer_when_unknown)`	used by LPD when printer name not in printcap

11.3.3 (default_priority) default job priority
11.3.4 (default_remote_host) default remote host
11.3.5 (default_tmp_dir) default directory for temp files
8.13 (destinations) printers that a route filter may return and we should query
12.1 (ff) string to send for a form feed
13.6 (filter_ld_path) filter LD_LIBRARY_PATH value
13.6 (filter_options) filter options
13.6 (filter_path) filter PATH environment variable
12.1 (fo) send form feed when device is opened
11.14.6 (force_fqdn_hostname) force FQDN HOST value in control file
7.1 (force_localhost) force clients to send all requests to localhost
11.4 (force_queuename) force use of this queuename if none provided
12.5 (fq) send form feed when device is closed
11.12 (full_time) use extended time format
4.7 (group) Effective Group ID (EGID) for SUID ROOT programs
12.2 (hl) Header (banner) last, at end of job
11.14.6 (ignore_requested_user_priority) Ignore requested user priority
13.2 (if) default (f, l) filter program
11.13 (ipv6) using IPV6 conventions
16.9.1 (kerberos_keytab) kerberos keytab file location
16.9.1 (kerberos_life) kerberos key lifetime
16.9.1 (kerberos_renew) kerberos key renewal time
16.9.1 (kerberos_forward_principal) kerberos remote principle name for forwarding
16.9.1 (kerberos_server_principal) kerberos remote server principle name
16.9.1 (kerberos_service) kerberos default service
12.1 (la) enable local printer accounting (if af is set)
12.1 (ld) leader string sent on printer open
10.4 (lf) error log file for spool queue
12.1 (lk) lock the IO device
11.13 (lockfile) lpd lock file
17.6.1 (logger_destination) destination for logging information
17.6.1 (logger_timeout) intervals between connection attempts
17.6.1 (logger_pathname) temp file for log information
17.6.1 (logger_max_size) max size in Kbytes of temp file for log information
18.3 (longnumber) use long job number when a job is submitted
8.1 (lp) printer device name or specification
8.10 (lpd_bounce) force lpd to filter job before forwarding
12.7 (lpd_force_poll) force lpd to poll idle printers
12.7 (lpd_poll_time) interval between lpd printer polls
18.1 (lpd_port) lpd listening port
8.15 (lpd_printcap_path) lpd printcap path
8.12 (lpr_bounce) lpr does filtering as in bounce queue
11.14.7 (lpr_bsd) lpr does filtering as in bounce queue
12.6 (mail_from) mail user from user name
12.6 (mail_operator_on_error) mail to this operator on error
12.6 (max_connect_interval) maximum time between connection attempts
10.4 (max_log_file_size) maximum size (in K) of spool queue log file
12.7 (max_servers_active) maximum number of lpd queue servers that can be active
10.4 (max_status_line) maximum length of status line
10.4 (max_status_size) maximum size (in K) of status file

11.10 (mc) maximum copies allowed
10.4 (min_log_file_size) minimum size (in K) of spool queue log file
10.4 (min_status_size) minimum size to reduce status file to
11.11 (minfree) minimum amount of free space needed
11.5 (ml) minimum number of printable characters for printable check
11.12 (ms_time_resolution) millisecond time resolution
8.2 (mx) maximum job size (1Kb blocks, 0 = unlimited)
12.1 (nb) use nonblocking device open
12.1 (network_connect_grace) pause between transferring jobs to remote printer
12.2 (of) banner output filter
13.6 (of_filter_options) OF filter options
18.1 (originate_port) originate connections from these ports
13.6 (pass_env) clients pass these environment variables to filters
14.4 (perms_path) lpd.perms files
13.6 (pl) page length (in lines)
13.4 (pr) pr program for p format
8.15 (printcap_path) /etc/printcap files
10.4 (ps) printer status file name
13.6 (pw) page width (in characters)
13.6 (px) page width in pixels (horizontal)
13.6 (py) page length in pixels (vertical)
11.4 (qq) put queue name in control file
18.2 (remote_support) operations allowed to remote host
11.13 (report_server_as) server name for status reports
18.1 (retry_econnrefused) Retry on connect ECONNREFUSED errors
18.1 (retry_nolink) Retry device open or connect failures
11.14.4 (return_short_status) return short lpq status when request arrives from specified host
18.1 (reuse_addr) set SO_REUSEADDR on outgoing ports
11.14.3 (reverse_lpq_format) reverse lpq format when request arrives from specified host
11.6 (rg) clients allow only users in this group access to printer
8.1 (rm) remote machine (hostname) (with rp)
8.13 (router) routing filter, returns destinations
8.1 (rp) remote printer name (with rm)
12.1 (rw) open printer for reading and writing
11.8 (safe_chars) additional safe characters in control file lines
12.6 (save_on_error) save job when an error
12.5 (save_when_done) save job when done
12.2 (sb) short banner (one line only)
10 (sd) spool directory pathname
18.9 (send_block_format) send block of data, rather than individual files
18.3 (send_data_first) send data files first in job transfer
12.6 (send_failure_action) failure action to take after send_try attempts failed
12.3 (send_job_rw_timeout) print job read/write timeout
12.3 (send_query_rw_timeout) status query operation read/write timeout
12.6 (send_try) maximum number of times to try sending job
12.6 (sendmail) sendmail program
12.1 (server_tmp_dir) server temporary file directory
12.3 (sf) suppress form feeds separating data files in job
12.2 (sh) suppress header (banner) pages

11.14.4 (short_status_length)	short lpq status length in lines
18.1 (socket_linger)	set the SO_LINGER socket option
11.13 (spool_dir_perms)	spool directory permissions
11.13 (spool_file_perms)	spool file permissions
11.13 (spread_jobs)	amount to spread jobs to avoid collisions
8.14.1 (ss)	name of queue that server serves (with sv)
17 (stalled_time)	time after which to report active job stalled
12.6 (stop_on_abort)	stop processing queue on filter abort
8.8 (stty)	stty commands to set output line characteristictics
8.14.1 (sv)	names of servers for queue (with ss)
11.12 (syslog_device)	name of syslog device
12.5 (tr)	trailer string to send before closing printer
8.11 (translate_format)	translate data format in control file
11.12 (use_date)	force date in control file
10.8 (use_identifier)	force identifier in control file
11.12 (use_info_cache)	read and cache information
11.4 (use_queuename)	put queue name in control file (alias for qq)
18.3 (use_shorthost)	Use short hostname for lpr control and data file names
4.7 (user)	Effective User ID (EUID) for SUID ROOT programs