Linux Training

Paul Cobbaut

Linux Training

Paul Cobbaut

lt-0.970.177

Published Wed 10 Dec 2008 10:10:33 PM CET

Abstract

This book is meant to be used in an instructor-led training. For self-study, the idea is to read this book next to a working Linux computer so you can immediately do every subject, even every command.

This book is aimed towards novice Linux system administrators (and might be interesting and useful for home users that want to know a bit more about their Linux system). However, this book is not meant as an introduction to Linux desktop applications like text editors, browsers, mail clients, multimedia or office applications.

More information and free .pdf available at **http://www.linux-training.be** .

Feel free to contact the authors:

- Paul Cobbaut: paul.cobbaut@gmail.com, http://www.linkedin.com/in/cobbaut

Contributors to the Linux Training project are:

- Serge van Ginderachter: serge@ginsys.be, docbook xml and pdf build scripts; svn hosting
- Hendrik De Vloed: hendrik.devloed@ugent.be, buildheader.pl script

We'd also like to thank our reviewers:

- Wouter Verhelst: wouter@grep.be, http://grep.be
- Geert Goossens: goossens.geert@gmail.com, http://www.linkedin.com/in/geertgoossens
- Elie De Brauwer: elie@de-brauwer.be, http://www.de-brauwer.be
- Christophe Vandeplas: christophe@vandeplas.com, http://christophe.vandeplas.com

Table of Contents

1. Introduction to Unix and Linux 1
1.1. Unix History 1
1.1.1. AT&T Bell Labs 1
1.1.2. The Unix Wars 1
1.1.3. University of California, Berkeley 2
1.1.4. GNU's not Unix 2
1.1.5. Linux 3
1.2. Licensing 3
1.2.1. Proprietary 3
1.2.2. BSD 3
1.2.3. GNU General Public License (GPL) 3
1.2.4. Others... 4
1.3. Current Distributions 4
1.3.1. What is a distribution ? 4
1.3.2. Linux Distributions 4
1.3.3. BSD Distributions 7
1.3.4. Major Vendors of Unix 8
1.3.5. Solaris 8
1.4. Certification 9
1.4.1. LPI: Linux Professional Institute 9
1.4.2. Red Hat Certified Engineer 9
1.4.3. MySQL 9
1.4.4. Novell CLP/CLE 10
1.4.5. Sun Solaris 10
1.4.6. Other certifications 10
1.5. Where to find help ? 10
1.5.1. Manual Pages 10
1.5.2. Red Hat Manuals online 11
1.5.3. Searching the internet with Google 11
1.5.4. Wikipedia 12
1.5.5. The Linux Documentation Project 12
1.5.6. This book 12
1.6. Discovering the classroom 12
2. First Steps 13
2.1. Working with directories 13
2.1.1. pwd 13
2.1.2. cd 13
2.1.3. ls 15
2.1.4. mkdir 17
2.1.5. rmdir 17
2.2. Practice: Working with directories 18
2.3. Solution: Working with directories 18
2.4. Working with files 20
2.4.1. file 20
2.4.2. touch 21
2.4.3. rm 22

2.4.4. cp 22
2.4.5. mv 24
2.4.6. rename 24
2.5. Practice: Working with files 25
2.6. Solution: Working with files 25
2.7. File contents 27
2.7.1. head 27
2.7.2. tail 27
2.7.3. cat 28
2.7.4. tac 29
2.7.5. more and less 29
2.7.6. strings 30
2.8. Practice: File contents 30
2.9. Solution: File contents 30
3. The Linux File system Tree 32
3.1. About files on Linux 32
3.1.1. case sensitive 32
3.1.2. everything is a file 32
3.1.3. root directory 32
3.1.4. man hier (FileSystem Hierarchy) 32
3.1.5. Filesystem Hierarchy Standard 33
3.2. Filesystem Hierarchy 33
3.2.1. /bin binaries 33
3.2.2. /boot static files to boot the system 33
3.2.3. /dev device files 34
3.2.4. /etc Configuration Files 34
3.2.5. /home sweet home 36
3.2.6. /initrd 37
3.2.7. /lib shared libraries 37
3.2.8. /media for Removable Media 38
3.2.9. /mnt standard mount point 38
3.2.10. /opt Optional software 38
3.2.11. /proc conversation with the kernel 38
3.2.12. /root the superuser's home 45
3.2.13. /sbin system binaries 45
3.2.14. /srv served by your system 45
3.2.15. /sys Linux 2.6 hot plugging 45
3.2.16. /tmp for temporary files 46
3.2.17. /usr Unix System Resources 46
3.2.18. /var variable data 46
3.2.19. Practice: file system tree 47
3.2.20. Solutions: file system tree 48
4. Introduction to the shell 51
4.1. about shells 51
4.1.1. several shells 51
4.1.2. type 51
4.1.3. which 51
4.1.4. alias 52

4.1.5. echo 53
4.1.6. shell expansion 53
4.1.7. internal and external commands 54
4.1.8. displaying shell expansion 54
4.2. Practice: about shells 54
4.3. Solution: about shells 55
4.4. control operators 56
4.4.1. ; semicolon 56
4.4.2. & ampersand 57
4.4.3. && double ampersand 57
4.4.4. || double vertical bar 57
4.4.5. Combining && and || 58
4.4.6. # pound sign 58
4.4.7. \ escaping special characters 58
4.4.8. end of line backslash 59
4.5. Practice: control operators 59
4.6. Solution: control operators 59
4.7. shell variables 60
4.7.1. $ dollar sign 60
4.7.2. common variables 61
4.7.3. $? dollar question mark 61
4.7.4. unbound variables 61
4.7.5. creating and setting variables 62
4.7.6. set 62
4.7.7. unset 62
4.7.8. env 62
4.7.9. exporting variables 63
4.7.10. delineate variables 63
4.7.11. quotes and variables 63
4.8. Practice: shell variables 64
4.9. Solution: shell variables 64
4.10. white space and quoting 66
4.10.1. white space removal 66
4.10.2. single quotes 66
4.10.3. double quotes 66
4.10.4. echo and quotes 67
4.10.5. shell embedding 67
4.10.6. backticks and single quotes 68
4.11. Practice: white space and quoting 68
4.12. Solution: white space and quoting 69
4.13. file globbing 70
4.13.1. * asterisk 70
4.13.2. ? question mark 70
4.13.3. [] square brackets 70
4.13.4. a-z and 0-9 ranges 71
4.13.5. $LANG and square brackets 71
4.14. Bash shell options 72
4.15. shell history 72

4.15.1. history variables 72
4.15.2. repeating commands in bash 73
4.15.3. repeating commands in ksh 74
4.16. Practice: discover the shell 74
4.17. Solution: discover the shell 75
5. Introduction to vi 78
5.1. About vi 78
5.2. Introduction to using vi 78
5.2.1. command mode and insert mode 78
5.2.2. Start typing (a A i I o O) 78
5.2.3. Replace and delete a character (r x) 78
5.2.4. Undo and repeat (u .) 79
5.2.5. Cut, copy and paste a line (dd yy p P) 79
5.2.6. Cut, copy and paste lines (3dd 2yy) 79
5.2.7. Start and end of a line (0 or ^ and $) 79
5.2.8. Join two lines (J) 79
5.2.9. Words (w b) 79
5.2.10. Save (or not) and exit (:w :q :q!) 79
5.2.11. Searching (/ ?) 79
5.2.12. Replace all (:1,$ s/foo/bar/g) 80
5.2.13. Reading files (:r :r !cmd) 80
5.2.14. Setting options 80
5.3. Practice 80
5.4. Solutions to the Practice 81
6. Introduction to Users 82
6.1. Users 82
6.1.1. User management 82
6.1.2. /etc/passwd 82
6.1.3. root 82
6.1.4. useradd 83
6.1.5. default useradd options 83
6.1.6. userdel 83
6.1.7. usermod 83
6.2. Identify yourself 84
6.2.1. whoami 84
6.2.2. who 84
6.2.3. who am i 84
6.2.4. w 84
6.2.5. id 84
6.3. Passwords 85
6.3.1. passwd 85
6.3.2. /etc/shadow 85
6.3.3. password encryption 86
6.3.4. password defaults 87
6.3.5. disabling a password 88
6.3.6. editing local files 89
6.4. About Home Directories 89
6.4.1. creating home directories 89

6.4.2. /etc/skel/ 89
6.4.3. deleting home directories 90
6.5. User Shell 90
6.5.1. login shell 90
6.5.2. chsh 90
6.6. Switch users with su 91
6.7. Run a program as another user 91
6.7.1. About sudo 91
6.7.2. setuid on sudo 92
6.7.3. visudo 92
6.7.4. sudo su 93
6.8. Practice Users 93
6.9. Solutions to the practice 94
7. Introduction to Groups 97
7.1. About Groups 97
7.2. groupadd 97
7.3. /etc/group 97
7.4. usermod 97
7.5. groupmod 98
7.6. groupdel 98
7.7. groups 98
7.8. gpasswd 98
7.9. vigr 99
7.10. Practice: Groups 99
7.11. Solution: Groups 100
8. Standard File Permissions 101
8.1. file ownership 101
8.1.1. user owner and group owner 101
8.1.2. chgrp 101
8.1.3. chown 101
8.2. special files 102
8.3. permissions 102
8.4. setting permissions (chmod) 103
8.5. setting octal permissions 104
8.6. umask 105
8.7. Practice: File Permissions 105
8.8. Solutions: File Permissions 106
8.9. Sticky bit on directory 108
8.10. SetGID bit on directory 108
8.11. Practice: sticky and setGID on directory 109
8.12. Solution: sticky and setGID on directory 109
9. File Links 111
9.1. about inodes 111
9.2. about directories 111
9.3. hard links 112
9.4. symbolic links 112
9.5. removing links 113
9.6. Practice: Links 113

9.7. Solutions: Links 113
10. Introduction to Scripting 115
10.1. About scripting 115
10.2. Hello World 115
10.3. Variables 115
10.4. Shell 116
10.5. for loop 117
10.6. while loop 117
10.7. until loop 117
10.8. parameters 117
10.9. test [ ] 118
10.10. if if, then then, or else 119
10.11. let 120
10.12. runtime input 120
10.13. sourcing a config file 120
10.14. case 121
10.15. shopt 122
10.16. Practice : scripts 122
10.17. Solutions 123
11. Process Management 127
11.1. About processes 127
11.2. Process ID 127
11.3. fork 127
11.4. exec 128
11.5. ps 128
11.6. top 129
11.7. priority and nice values 129
11.8. signals (kill) 130
11.9. pgrep 131
11.10. pkill 131
11.11. jobs 131
11.12. fg 132
11.13. bg 132
11.14. Zombiecreator ;-) 133
11.15. Practice 133
11.16. Solutions to the Practice 134
12. More about Bash 136
12.1. bash shell environment 136
12.2. path 136
12.3. Shell I/O redirection 137
12.3.1. output redirection 137
12.3.2. noclobber 138
12.3.3. append 138
12.3.4. error redirection 138
12.3.5. input redirection 139
12.3.6. here document 139
12.4. Confusing I/O redirection 139
12.5. swapping stdout and stderr 140

12.6. Practice: more bash ... 140

13. Pipes and Filters ... 141

13.1. pipes ... 141
13.2. tee ... 142
13.3. grep ... 142
13.4. cut ... 143
13.5. tr ... 144
13.6. wc ... 145
13.7. sort ... 145
13.8. uniq ... 146
13.9. find ... 147
13.10. locate ... 147
13.11. diff ... 148
13.12. comm ... 148
13.13. compress ... 149
13.14. od ... 149
13.15. other tools and filters ... 150
13.16. Practice tools and filters ... 150
13.17. Solutions: tools and filters ... 151

14. Disk Management ... 153

14.1. Hard disk devices ... 153
14.1.1. Terminology ... 153
14.1.2. IDE or SCSI ... 153
14.1.3. Device Naming ... 154
14.1.4. Erasing a hard disk ... 154
14.1.5. fdisk ... 155
14.1.6. hdparm ... 156
14.1.7. dmesg ... 156
14.1.8. /proc/scsi/scsi ... 156
14.1.9. scsi_info ... 157
14.1.10. lsscsi ... 157
14.1.11. Practice hard disk devices ... 157
14.2. Partitions ... 157
14.2.1. About Partitions ... 157
14.2.2. Partition naming ... 158
14.2.3. fdisk -l ... 158
14.2.4. other tools ... 158
14.2.5. Partitioning new disks ... 159
14.2.6. Master Boot Record ... 160
14.2.7. partprobe ... 161
14.2.8. logical drives ... 161
14.2.9. Practice Partitions ... 161
14.3. File Systems ... 161
14.3.1. About file systems ... 161
14.3.2. Common file systems ... 162
14.3.3. Putting a file system on a partition ... 163
14.3.4. Tuning a file system ... 164
14.3.5. Checking a file system ... 164

14.3.6. Practice File Systems ... 165
14.4. Mounting ... 165
14.4.1. Mounting local disks ... 166
14.4.2. Displaying mounted file systems ... 166
14.4.3. Permanent mounts ... 167
14.4.4. Disk Usage (du) ... 167
14.4.5. Disk Free (df) ... 167
14.4.6. Practice Mounting File Systems ... 168
14.5. UUID and filesystems ... 168
14.5.1. About Unique Objects ... 168
14.5.2. Using UUID in /etc/fstab ... 169
14.5.3. Practice UUID ... 169
14.6. RAID ... 170
14.6.1. Hardware or software ... 170
14.6.2. RAID levels ... 170
14.6.3. Building a software RAID array ... 171
14.6.4. /proc/mdstat ... 174
14.6.5. Removing a software RAID ... 174
14.6.6. Practice RAID ... 174
15. Logical Volume Management ... 176
15.1. Introduction to lvm ... 176
15.1.1. Problems with standard partitions ... 176
15.1.2. Solution with lvm ... 176
15.1.3. About lvm ... 176
15.2. LVM Terminology ... 177
15.2.1. Physical Volume (pv) ... 177
15.2.2. Volume Group (vg) ... 177
15.2.3. Logical Volume (lv) ... 177
15.3. Verifying existing Physical Volumes ... 177
15.3.1. lvmdiskscan ... 177
15.3.2. pvs ... 178
15.3.3. pvscan ... 178
15.3.4. pvdisplay ... 178
15.4. Verifying existing Volume Groups ... 179
15.4.1. vgs ... 179
15.4.2. vgscan ... 179
15.4.3. vgdisplay ... 179
15.5. Verifying existing Logical Volumes ... 180
15.5.1. lvs ... 180
15.5.2. lvscan ... 180
15.5.3. lvdisplay ... 180
15.6. Manage Physical Volumes ... 181
15.6.1. pvcreate ... 181
15.6.2. pvremove ... 181
15.6.3. pvresize ... 182
15.6.4. pvchange ... 182
15.6.5. pvmove ... 182
15.7. Manage Volume Groups ... 183

15.7.1. vgcreate 183
15.7.2. vgextend 183
15.7.3. vgremove 183
15.7.4. vgreduce 183
15.7.5. vgchange 184
15.7.6. vgmerge 184
15.8. Manage Logical Volumes 184
15.8.1. lvcreate 184
15.8.2. lvremove 185
15.8.3. lvextend 186
15.8.4. lvrename 186
15.9. Example: Using lvm 186
15.10. Example: Extend a Logical Volume 188
15.11. Example: Resize a Physical Volume 189
15.12. Example: Mirror a Logical Volume 191
15.13. Example: Snapshot a Logical Volume 192
15.14. Practice LVM 193
16. Booting Linux 194
16.1. Booting the system 194
16.2. GRUB 194
16.2.1. GRand Unified Bootloader 194
16.2.2. menu.lst 194
16.2.3. Stanza commands 195
16.2.4. Chainloading 195
16.2.5. Installing grub 196
16.3. Lilo 196
16.3.1. Linux Loader 196
16.3.2. lilo.conf 196
16.4. Booting 196
16.5. Daemons 197
16.6. Init 197
16.6.1. /etc/inittab 197
16.6.2. Runlevel 197
16.6.3. sysinit 198
16.6.4. rc scripts 198
16.6.5. Power and Ctrl-Alt-Del 199
16.6.6. getty 199
16.7. Starting and stopping daemons 200
16.8. Display the runlevel 201
16.9. Changing the runlevel 201
16.10. more info 201
16.11. Practice 201
16.12. Solutions 202
17. Linux Kernel 204
17.1. about the Linux kernel 204
17.1.1. kernel versions 204
17.1.2. uname -r 204
17.2. Linux kernel source 204

17.2.1. ftp.kernel.org 204
17.2.2. /usr/src 205
17.2.3. downloading the kernel source 206
17.3. kernel boot files 208
17.3.1. vmlinuz 208
17.3.2. initrd 208
17.3.3. System.map 209
17.3.4. .config 209
17.4. Linux kernel modules 209
17.4.1. about kernel modules 209
17.4.2. /lib/modules 210
17.4.3. <module>.ko 210
17.4.4. lsmod 210
17.4.5. /proc/modules 210
17.4.6. insmod 211
17.4.7. modinfo 211
17.4.8. modprobe 211
17.4.9. /lib/modules/<kernel>/modules.dep 212
17.4.10. depmod 212
17.4.11. rmmod 212
17.4.12. modprobe -r 212
17.4.13. /etc/modprobe.conf 213
17.5. compiling a kernel 213
17.5.1. extraversion 213
17.5.2. make mrproper 213
17.5.3. .config 214
17.5.4. make menuconfig 214
17.5.5. make clean 214
17.5.6. make bzImage 214
17.5.7. make modules 215
17.5.8. make modules_install 215
17.5.9. /boot 215
17.5.10. mkinitrd 216
17.5.11. bootloader 216
17.6. compiling one module 216
17.6.1. hello.c 216
17.6.2. Makefile 216
17.6.3. make 217
17.6.4. hello.ko 217
18. Introduction to Networking 219
18.1. About TCP/IP 219
18.1.1. Overview of tcp/ip v4 219
18.1.2. Internet and routers 219
18.1.3. many protocols 219
18.1.4. Practice TCP/IP 220
18.2. Using TCP/IP 220
18.2.1. to GUI or not to GUI 220
18.2.2. /sbin/ifconfig 220

18.2.3. /etc/init.d/network(ing) 221
18.2.4. /etc/sysconfig 221
18.2.5. /sbin/ifup and /sbin/ifdown 223
18.2.6. /sbin/dhclient 223
18.2.7. /sbin/route 224
18.2.8. arp 224
18.2.9. ping 224
18.2.10. Red Hat network settings backup 225
18.2.11. Restarting the network 225
18.2.12. ethtool 225
18.2.13. Practice IP Configuration 226
18.3. multiple IP adresses 226
18.3.1. Binding multiple ip-addresses 226
18.3.2. Enabling extra ip-addresses 227
18.3.3. Practice multiple IP addresses 227
18.4. multihomed hosts 227
18.4.1. bonding 227
18.4.2. /proc/net/bond* 229
18.4.3. Practice multihomed hosts 229
18.5. Introduction to iptables 229
18.5.1. Introducing iptables 229
18.5.2. Practice iptables 231
18.6. xinetd and inetd 231
18.6.1. About the superdaemon 231
18.6.2. inetd or xinetd 231
18.6.3. The superdaemon xinetd 232
18.6.4. The superdaemon inetd 233
18.6.5. Practice 233
18.7. OpenSSH 234
18.7.1. Secure Shell 234
18.7.2. SSH Protocol versions 234
18.7.3. About Public and Private keys 235
18.7.4. Setting up passwordless ssh 235
18.7.5. X forwarding via SSH 237
18.7.6. Troubleshooting ssh 238
18.7.7. Practice SSH 238
18.8. Network File System 238
18.8.1. Network Attached Storage (NAS) 238
18.8.2. NFS: the Network File System 239
18.8.3. Practice NFS 241
19. Scheduling 242
19.1. About scheduling 242
19.2. at 242
19.3. cron 243
19.3.1. crontab 243
19.3.2. /etc/cron* 243
19.4. Practice Scheduling 244
20. Logging 245

20.1. About logging 245
20.1.1. /var/log 245
20.1.2. /var/log/messages 245
20.2. Login logging 245
20.2.1. /var/run/utmp (who) 246
20.2.2. /var/log/wtmp (last) 246
20.2.3. /var/log/lastlog (lastlog) 246
20.2.4. /var/log/btmp (lastb) 247
20.2.5. su and ssh logins 247
20.3. Syslogd daemon 248
20.3.1. About syslog 248
20.3.2. Facilities 249
20.3.3. Levels 249
20.3.4. Actions 249
20.3.5. Configuration 250
20.4. logger 250
20.5. Watching logs 251
20.6. Rotating logs 251
20.7. Practice Logging 251
21. Memory Management 252
21.1. About Memory 252
21.2. Swap space 252
21.2.1. About swap space 252
21.2.2. Creating a swap partition 252
21.2.3. Creating a swap file 253
21.2.4. Swap space in /etc/fstab 253
21.3. Practice Memory 253
22. Installing Linux 254
22.1. About 254
22.2. Installation by cdrom 254
22.3. Installation with rarp and tftp 254
22.4. About Red Hat Kickstart 255
22.5. Using Kickstart 255
23. Package Management 257
23.1. About repositories 257
23.2. Red Hat Package Manager 257
23.2.1. rpm 257
23.2.2. yum 259
23.2.3. up2date 259
23.3. Debian Package Management 259
23.3.1. About .deb 259
23.3.2. dpkg -l 260
23.3.3. dpkg 260
23.3.4. aptitude 260
23.4. Downloading software 260
23.5. Compiling software 261
23.6. Practice: Installing software 261
23.7. Solution: Installing software 261

24. Backup 263
24.1. About tape devices 263
24.1.1. SCSI tapes 263
24.1.2. IDE tapes 263
24.1.3. mt 264
24.2. Compression 264
24.3. tar 265
24.4. Backup Types 267
24.5. dump and restore 267
24.6. cpio 268
24.7. dd 268
24.7.1. About dd 268
24.7.2. Create a CDROM image 268
24.7.3. Create a floppy image 269
24.7.4. Copy the master boot record 269
24.7.5. Copy files 269
24.7.6. Image disks or partitions 269
24.7.7. Create files of a certain size 269
24.7.8. CDROM server example 269
24.8. split 270
24.9. Practice backup 270
25. Performance Monitoring 272
25.1. About Monitoring 272
25.2. top 272
25.3. free 272
25.4. watch 273
25.5. vmstat 273
25.6. iostat 274
25.7. mpstat 274
25.8. sadc and sar 275
25.9. ntop 275
25.10. iftop 275
26. User Quota's 277
26.1. About Disk Quotas 277
26.2. Practice Disk quotas 277
27. VNC 278
27.1. About VNC 278
27.2. VNC Server 278
27.3. VNC Client 278
27.4. Practive VNC 279
28. Cloning a Linux Server 280
28.1. About cloning 280
28.2. About offline cloning 280
28.3. Offline cloning example 280
29. Introduction to Samba 282
29.1. Verify installed version 282
29.2. Installing Samba 282
29.3. Documentation 282

29.4. smb.conf ... 283
29.5. testparm ... 284
29.6. Samba daemons ... 285
29.7. smbclient ... 286
29.8. smbtree ... 287
29.9. Samba Web Administration Tool (SWAT) ... 289
29.10. Practice ... 289

30. Simple Samba File Servers ... 291

30.1. Read Only File Server ... 291
30.2. Practice ... 293
30.3. Writable File Server ... 293
30.4. Forcing a User Owner ... 294
30.5. More about smbclient ... 295
30.6. NetBIOS name resolving ... 296
30.7. Practice ... 297

31. Samba Servers with authentication and restrictions ... 298

31.1. Authenticated User Access ... 298
31.2. Frequently used share settings ... 299
31.2.1. valid users ... 299
31.2.2. invalid users ... 300
31.2.3. create mask and inherit permissions ... 300
31.2.4. hosts allow ... 300
31.2.5. hosts deny ... 300
31.2.6. hide unreadable ... 301
31.2.7. read list ... 301
31.3. Practice ... 301

32. Samba Domain Member Server ... 303

32.1. smb.conf ... 303
32.2. Joining the Active Directory Domain ... 303
32.3. nsswitch.conf ... 304
32.4. winbind ... 304
32.5. Practice ... 305

33. Samba Domain Controller ... 306

33.1. About Domain Controllers ... 306
33.1.1. Samba 3 ... 306
33.1.2. Samba 4 ... 306
33.1.3. About password backends ... 306
33.2. smb.conf ... 306
33.3. Users and Groups ... 307
33.4. About Computer Accounts ... 308
33.5. Roaming Profiles ... 308
33.6. Groups in NTFS acls ... 309
33.7. logon scripts ... 310
33.8. Practice ... 310

34. Samba Print Servers ... 312

34.1. Simple CUPS Print Server ... 312
34.2. Simple BSD Print Server ... 313
34.3. Simple Unix SysV Print Server ... 314

34.4. Samba Prining tips ... 314
34.5. Practice ... 315
35. Introduction to Apache ... 316
35.1. about apache ... 316
35.2. is apache installed ? ... 316
35.3. is apache running ? ... 316
35.4. apache configuration ... 317
35.5. virtual hosts ... 317
35.6. aliases and redirects ... 318
35.7. securing directories with htpasswd and .htaccess ... 319
35.8. more on .htaccess ... 320
35.9. traffic ... 320
35.10. practice: apache ... 320
36. Routers ... 321
36.1. terminology ... 321
36.1.1. router or firewall ... 321
36.1.2. packet forwarding ... 321
36.1.3. packet filtering ... 321
36.1.4. stateful ... 321
36.1.5. NAT (network address translation) ... 321
36.2. packet forwarding ... 322
36.2.1. about packet forwarding ... 322
36.2.2. /proc/sys/net/ipv4/ip_forward ... 322
36.2.3. /etc/sysctl.conf ... 323
36.2.4. Practice: packet forwarding ... 323
36.2.5. Solution: packet forwarding ... 324
37. Iptables ... 327
37.1. about iptables ... 327
37.2. packet filtering ... 327
37.2.1. about packet filtering ... 327
37.2.2. filter table ... 327
37.2.3. Changing default policy rules ... 328
37.2.4. Allowing ssh over eth0 ... 328
37.2.5. Allowing access from a subnet ... 329
37.2.6. iptables save ... 329
37.2.7. scripting example ... 329
37.2.8. Allowing ICMP(ping) ... 330
37.2.9. Practice: packet filtering ... 330
37.2.10. Solution: packet filtering ... 331
37.3. network address translation ... 331
37.3.1. about NAT ... 331
37.3.2. SNAT (Source NAT) ... 332
37.3.3. SNAT example setup ... 332
37.3.4. IP masquerading ... 333
37.3.5. DNAT (Destination NAT) ... 333
38. Introduction to squid ... 334
38.1. about proxy servers ... 334
38.1.1. usage ... 334

38.1.2. open proxy servers 334
38.1.3. squid 334
38.2. squid proxy server 335
38.2.1. /etc/squid/squid.conf 335
38.2.2. /var/spool/squid 335
38.2.3. port 3128 or port 8080 335
38.2.4. /var/log/squid 336
38.2.5. access control 336
38.2.6. testing squid 336
38.2.7. name resolution 337
39. Introduction to bind 338
39.1. DNS History 338
39.2. DNS Structure 338
39.2.1. root 338
39.2.2. top level domains (TLD) 338
39.2.3. domains 339
39.2.4. fully qualified domain name 339
39.3. How DNS works 339
39.3.1. zone 339
39.3.2. master and slave 340
39.3.3. zone transfers 340
39.3.4. full or incremental zone transfers 341
39.3.5. DNS cache 341
39.3.6. caching only server 341
39.3.7. iterative or recursive query 341
39.4. old stuff....work in progress 342
39.5. bind 342
39.6. named 342
39.7. Caching only Name Server 342
39.8. Our first zone 343
39.9. Starting the name server 343
39.10. Practice DNS 344
A. Keyboard settings 345
A.1. About Keyboard Layout 345
A.2. X Keyboard Layout 345
A.3. Shell Keyboard Layout 345
Index 347

List of Tables

1.1. Early Unix Timeline 1
1.2. Eighties Unix Timeline 2
1.3. Current BSD Timeline 2
8.1. Unix special files 102
8.2. standard Unix file permissions 102
8.3. Octal permissions 104
14.1. IDE device naming 154
14.2. SCSI device naming 154
14.3. Partition naming 158
15.1. Disk Partitioning Example 176
15.2. LVM Example 176
36.1. Packet Forwarding Exercise 323
36.2. Packet Forwarding Solution 325

Chapter 1. Introduction to Unix and Linux

1.1. Unix History

1.1.1. AT&T Bell Labs

In 1969 **Dennis Ritchie** and **Ken Thompson** wrote **UNICS** (Uniplexed Information and Computing System) at Bell Labs. Together with **Douglas McIlroy** they are seen as the creators of Unix. The name Unix is a play on the Multics Operating System for large mainframe computers. Unics (later renamed to Unix) was written for mini computers like the DEC PDP-series. In 1973 they decided to write Unix in C (instead of assembler), to make it portable to other computers. Unix was made available to universities, companies and the US government, including the full source code. This meant that every C programmer could make changes. By 1978 about 600 computers were running Unix.

Table 1.1. Early Unix Timeline

1969-1977	1978-1980	1981	1982
UNIX Time Sharing System	BSD	4.1BSD	4.1BSD
			SunOS 1.0
	Unix		Unix System III

1.1.2. The Unix Wars

The unity and openness that existed in the Unix world until 1977 was long gone by the end of the eighties. Different vendors of distinct versions of Unix tried to set the standard. Sun and AT&T joined the **X/Open** group to unify Unix. Other vendors joined the Open Software Foundation or **OSF**. These struggles were not good for Unix, allowing for new operating system families like OS/2, Novell Netware and Microsoft Windows NT to take big chunks of server market share in the early nineties. The table below shows the evolution of a united Unix into several Unixes in the eighties.

Table 1.2. Eighties Unix Timeline

1983	1984	1985	1986	1987	1988	1989	1990	1991	1992
4.1BSD			4.3BSD					BSD Net/2	
			4.3BSD		NeXTSTEP				
SunOS1.0			SunOS3.2		SystemVr4			Solaris	
System V								UnixWare	
System V			AIX						
III + V	HP-UX								

1.1.3. University of California, Berkeley

Students of Berkeley were happy to join in the development of Bell Labs Unix, but were not so happy with the restrictive licensing. Unix was open source software, but it still required purchase of a license. So during the eighties, they rewrote all the Unix tools, until they had a complete Unix-like operating system. By 1991, the **BSD** (Berkeley Software Distribution) branch of Unix was completely separate from the Bell Labs Unix. **NetBSD**, **FreeBSD** and **OpenBSD** are three current Unix-like operating systems derived from the 1991 **BSD Net/2** codebase. Sun Solaris, Microsoft Windows NT and Apple Mac OS X all used source code from BSD. The table below shows operating systems still in use today that are in a way derived from the 1978-1981 BSD codebase.

Table 1.3. Current BSD Timeline

1991	1992	1993	1994	1995	1996	1997	1998	1999	2000-2008
BSD Net/2		FreeBSD							
		NetBSD							
		NetBSD			OpenBSD				
NeXTSTEP								Mac OS X	
Solaris									

1.1.4. GNU's not Unix

Largely because of unhappiness with the restrictive licensing on existing Unix implementations, **Richard Stallman** initiated the **GNU Project** in 1983. The GNU project aims to create free software. Development of the GNU operating system started, aiming to create a complete Unix-like branch, seperate from the two other (BSD and Bell Labs). Today the GNU compiler **gcc** and most other GNU utilities (like **bash**) are among the most popular on many Unix-like systems. The official kernel of this project is **GNU/Hurd**, but you can hardly call that kernel a finished product.

1.1.5. Linux

Where **GNU/Hurd** failed, the Linux kernel succeeded! In 1991 a Finnish student named **Linus Torvalds** started writing his own operating system for his intel 80386 computer. In January 1992, Linus decided to release Linux under the GNU GPL. Thanks to this, thousands of developers are now working on the Linux kernel. Linus Torvalds is in charge of the kernel developers.

Contrary to popular believe, they are not all volunteers. Today big companies like Red Hat, Novell, IBM, Intel, SGI, Oracle, Montavista, Google, HP, NetApp, Cisco, Fujitsu, Broadcom and others are actively paying developers to work on the Linux kernel. According to the Linux Foundation "over 3700 individual developers from over 200 different companies have contributed to the kernel between 2005 and april 2008". 1057 developers from 186 different companies contributed code to make kernel version 2.6.23 into 2.6.24.

1.2. Licensing

1.2.1. Proprietary

Some flavors of Unix, like HP-UX, IBM AIX and Sun Solaris 9 are delivered after purchase in binary form. You are not authorized to install or use these without paying a license to the owner. You are not authorized to distribute these copies to other people, and you are not authorized to look at or change the closed source code of the operating system. This software is usually protected by copyright, patents and extensive software licensing.

1.2.2. BSD

BSD style licenses are close to the public domain. They essentially state that you can copy the software, but you have to leave the copyright notice that refers to BSD. This license gives a lot of freedom, but offers few protection to someone copying and selling your work.

1.2.3. GNU General Public License (GPL)

More and more software is being released under the **GPL** (in 2006 Java was released under the GPL). The goal of the GPL is to guarantee that free software stays free. Everyone can work together on GPL software, knowing that the software will be freely available to everyone. The GPL can protect software, even in court.

Free as in **freedom of speech**, not to be confused with free as in not having to pay for your free beer. In other words, or even better, in other languages free software translates to **vrije software** (Dutch) or **Logiciel Libre** (French). Whereas the free from free beer translates to gratis.

Briefly explained, the GPL allows you to copy software, the GPL allows you to distribute (sell or give away) that software, and the GPL grants you the right to read and change the source code. But the person receiving or buying the software from you has the same rights. And also, should you decide to distribute modified versions of GPL software, then you are obligated to put the same license on the modifications (and provide the source code of your modifications).

1.2.4. Others...

There are many other licenses on software. You should read and understand them before using any software.

1.3. Current Distributions

1.3.1. What is a distribution ?

Unix comes in many flavors, usually called **distributions**. A distribution (or in short distro) is a collection of software packages, distributed on CD, online or pre-installed on computers. All the software in a distribution is tested and integrates nicely into a whole. Software is maintained (patched) by the distributor, and is managed by an **integrated package manager**. Many distro's have a central **repository of approved software**. Installing software from outside the distro can sometimes be cumbersome and may void your warranty on the system.

1.3.2. Linux Distributions

There are hundreds of Linux distributions, just take a look at the distrowatch.com website. For many years, Red Hat, Suse and Mandrake were considered the big three for end users. Red Hat is still the biggest commercial Linux vendor, but in 2008, the most popular Linux distribution for home users is **Ubuntu** from Canonical.

1.3.2.1. Linux distribution detection

Depending on the distribution used, there are distinct files that contain the distribution version.

The **/etc/redhat-release** file contains the Red Hat version on most of the Red Hat and Red Hat derived systems. Debian and Ubuntu systems contain **/etc/debian-version**. Note that Ubuntu was originally derived from Debian.

```
paul@RHELv4u4:~$ cat /etc/redhat-release
```

```
Red Hat Enterprise Linux AS release 4 (Nahant Update 4)

serge@venusia:~$ cat /etc/debian_version
lenny/sid
```

The **/etc/lsb-release** file can be found on distributions that follow the Linux Standard Base. Other variations to these files are **/etc/slackware-version**, **/etc/SuSE-release**, **/etc/gentoo-release** and **/etc/mandrake-release**.

```
serge@venusia:~$ cat /etc/lsb-release
DISTRIB_ID=Ubuntu
DISTRIB_RELEASE=8.04
DISTRIB_CODENAME=hardy
DISTRIB_DESCRIPTION="Ubuntu 8.04.1"
```

1.3.2.2. Red Hat

Redhat exists as a company since 1993. They distribute **Red Hat Enterprise Linux** (RHEL) to companies and manage the **Fedora** project. RHEL is probably the most popular Linux-based distro on servers. Fedora is a very popular and user friendly Linux-based distro, aimed towards home users. The company makes a profit of around one hundred million dollars a year, selling support contracts. Red Hat contributes a lot to the Linux kernel and other free software projects.

1.3.2.2.1. Red Hat Linux

Red Hat Linux was distributed from 1994 until 2003. It was one of the oldest common Linux distributions. Red Hat Linux was the first distro to use the **rpm** package format. Many other distro's are originally derived from Red Hat Linux. The company **Red Hat, Inc.** decided to split Red Hat Linux into **Fedora** and **Red Hat Enterprise Linux**.

1.3.2.2.2. Fedora

Fedora is sponsored by Red Hat, and is aimed toward home users. There is no official support from Red Hat. Every six to eight months, there is a new version of Fedora. Fedora usually has more recent versions of kernel and applications than RHEL. Fedora 9 was released May 2008.

1.3.2.2.3. Red Hat Enterprise Linux 4

Since 2005 Red Hat distributes four different RHEL4 variants. **RHEL AS** is for mission-critical computer systems. **RHEL ES** is for small to mid-range servers. **RHEL WS** is for technical power user desktops and critical design. **Red Hat Desktop** is for multiple deployments of single user desktops. Red Hat does not give an explanation for the meaning of AS, ES and WS, but it might be Advanced Server, Entry-level Server and Workstation.

1.3.2.2.4. Red Hat Enterprise Linux 5

Red Hat Enterprise Linux version 5 is available since March 2007. One of the notable new features is the inclusion of **Xen**. Xen is a free virtual machine application that allows NetBSD and Linux to serve as host for guest OS'ses. Beyond just virtualization, RHEL 5 also has better SELinux support, clustering, network storage and smartcard integration.

1.3.2.2.5. CentOS and Unbreakable Linux

Both **CentOS** and Oracle's **Unbreakable Linux** are directly derived from RHEL, but all references to Red Hat trademarks are removed. Companies are allowed to do this (GPL), and are hoping to make a profit selling support (without having the cost to maintain and develop their own distribution). Red Hat is not really worried about this, since they develop a lot on Linux, and thus can offer much better support. The Oracle offer however is still very recent, let's wait and see how many organizations will buy a complete solution from Oracle.

1.3.2.3. Ubuntu

Ubuntu is a rather new distribution, based on **Debian** and funded by South African developer and billionaire astronaut **Mark Shuttleworth**. Ubuntu is giving away free (as in beer and speech) CD's with **Ubuntu, Linux for Human Beings**. Many people consider Ubuntu to be the most user friendly Linux distribution. The company behind Ubuntu is **Canonical**, they aim to make a profit of selling support soon. Ubuntu is probably the most popular Unix-like distribution on personal desktops.

Image copied from **xkcd.com**.

I HAVE FOUND THE PERFECT PHRASE FOR CONDESCENDINGLY DISMISSING ANYTHING:
HAVE YOU SEEN THE NEW UBUNTU RELEASE?
NAH, I'M NOT REALLY INTO POKÉMON.

1.3.2.4. Novell Suse

A couple of years ago, **Novell** bought the German company **Suse**. They are seen as the main competitor to Red Hat with their SLES (Suse Linux Enterprise Server) and SLED (Suse Linux Enterprise Desktop) versions of Suse Linux. Similar to Fedora, Novell hosts the **OpenSUSE** project as a testbed for upcoming SLED and SLES releases.

Novell has signed a very controversial deal with Microsoft. Some high profile open source developers have left the company because of this agreement, and many people from the open source community are actively advocating to abandon Novell completely.

1.3.2.5. Debian

Debian is one of the most secure Linux distro's. It is known to be stable and reliable. The Debian people also have a strong focus towards freedom. You will not find patented technologies or non-free software in the standard Debian repositories. A lot of distributions (Ubuntu, Knoppix, ...) are derived from the Debian codebase. Debian has **aptitude**, which is considered the best package management system.

1.3.2.6. Mandriva

Mandriva is the unification of the Brazilian distro Conectiva with the French distro Mandrake. They are considered a user friendly distro, with support from the French government.

1.3.3. BSD Distributions

1.3.3.1. FreeBSD

FreeBSD is a complete operating system. The kernel and all of the utilities are held in the same source code tree. FreeBSD runs on many architectures and is considered to be reliable and robust. Millions of websites are running on FreeBSD, including some big like yahoo.com, apache.org, sony.co.jp, netcraft, php.net, freebsd.org and (until last year) ftp.cdrom.com. Apple's MacOSX contains the FreeBSD virtual file system, network stack and more.

1.3.3.2. NetBSD

NetBSD development started around the same time (1993) as FreeBSD. NetBSD aims for maximum portability and thus runs on many architectures. NetBSD is often used in embedded devices.

1.3.3.3. OpenBSD

Co-founder **Theo De Raadt** from NetBSD founded the **OpenBSD** project in 1994. OpenBSD aims for maximum security. The past ten years, only two vulnerabilities were found in the default install of OpenBSD. All source code is thoroughly checked. OpenBSD runs on sixteen different architectures and is commonly used for firewalls and IDS. The OpenBSD people also bring us **OpenSSH**.

1.3.4. Major Vendors of Unix

We should at least mention IBM's **AIX**, Sun's **Solaris** and Hewlett-Packards **HP-UX**, all are based on the original Unix from Bell Labs (Unix System V). Sun's **SunOS**, HP's **Tru64** (originally from DEC) and Apple's **MacOSX** are more derived from the BSD branch. But most Unixes today may contain source code and implementations from both original Unix-branches.

1.3.5. Solaris

1.3.5.1. Solaris 8 and Solaris 9

All **Sun Solaris** releases before Solaris 10 are proprietary binary only, just like IBM AIX and HP-UX.

1.3.5.2. Solaris 10

Solaris 10 is the officialy supported Sun distribution. It is a free (as in beer) download. Sun releases binary patches and updates. Sun would like a community built around the solaris kernel, similar to the Linux community. Sun released the Solaris kernel under the CDDL, a license similar to the GPL, hoping this will happen.

1.3.5.3. Nevada and Solaris Express

Nevada is the codename for the next release of Solaris (Solaris 11). It is currently under development by Sun and is based on the OpenSolaris code. Solaris Express Community Edition is an official free binary release including open source OpenSolaris and some closed source technologies, updated twice a month without any support from Sun. Solaris Express Developer Edition is the same, but with some support, thorough testing before release, and released twice a year.

1.3.5.4. OpenSolaris, Belenix and Nexenta

OpenSolaris is een open source development project (yes, it is only source code). Future versions of the Solaris operating system are based on this source code. The **Belenix** LiveCD is based on OpenSolaris. Another famous opensolaris based distro is **Nexenta**. Nexenta (www.gnusolaris.org) looks like **Ubuntu** and feels like **Debian**. The goal of this **GNU/Solaris** project is to have the best Linux desktop (Ubuntu) including the **aptitude** package manager running on a Sun Solaris kernel.

1.4. Certification

1.4.1. LPI: Linux Professional Institute

1.4.1.1. LPIC Level 1

This is the junior level certification. You need to pass exams 101 and 102 to achieve **LPIC 1 certification**. To pass level one, you will need Linux command line, user management, backup and restore, installation, networking and basic system administration skills.

1.4.1.2. LPIC Level 2

This is the advanced level certification. You need to be LPIC 1 certified and pass exams 201 and 202 to achieve **LPIC 2 certification**. To pass level two, you will need to be able to administer medium sized Linux networks, including Samba, mail, news, proxy, firewall, web and ftp servers.

1.4.1.3. LPIC Level 3

This is the senior level certification. It contains one core exam (301) which tests advanced skills mainly about ldap. To achieve this level you also need LPIC Level 2 and pass a specialty exam (302 or 303). Exam 302 mainly focuses on Samba, 303 is about advanced security. More info on http://www.lpi.org.

1.4.1.4. Ubuntu

When you are LPIC Level 1 certified, you can take a LPI Ubuntu exam (199) and become Ubuntu certified.

1.4.2. Red Hat Certified Engineer

The big difference with most certs is that there are no multiple choice questions for **RHCE**. Red Hat Certified Engineers have taken a live exam consisting of two parts. First they have to troubleshoot and maintain an existing but broken setup (scoring at least 80 percent), second they have to install and configure a machine (scoring at least 70 percent).

1.4.3. MySQL

There are two tracks for MySQL certification; Certified MySQL 5.0 Developer (CMDEV) and Certified MySQL 5.0 DBA (CMDBA). The **CMDEV** is focused at database application developers, the **CMDBA** is for database administrators. Both tracks require two exams each. The MySQL cluster DBA certification requires CMDBA certification and passing the CMCDBA exam.

1.4.4. Novell CLP/CLE

To become a **Novell Certified Linux Professional**, you have to take a live practicum. This is a VNC session to a set of real SLES servers. You have to perform several tasks and are free to choose your method (commandline or YaST or ...). No multiple choice involved.

1.4.5. Sun Solaris

Sun uses the classical formula of multiple choice exams for certification. Passing two exams for an operating system gets you the Solaris Certified Administrator for Solaris X title.

1.4.6. Other certifications

There are many other less known certs like EC council's Certified Ethical Hacker, CompTIA's Linux+ and Sair's Linux GNU.

1.5. Where to find help ?

1.5.1. Manual Pages

Most Unix tools and commands have pretty good man pages. Type **man** followed by a command (for which you want help) and start reading. Ah, and press **q** to quit the manpage.

```
paul@laika:~$ man whois
Reformatting whois(1), please wait...
paul@laika:~$
```

Manpages can be useful when you are switching a lot between different flavors of unix, to find those little differences in commands. Very often manpages also describe configuration files and daemons.

```
paul@laika:~$ man syslog.conf
Reformatting syslog.conf(5), please wait...
paul@laika:~$ man syslogd
Reformatting syslogd(8), please wait...
```

The **man -k** command (same as **apropos**) will show you a list of manpages containing your searchstring.

```
paul@laika:~$ man -k syslog
lm-syslog-setup (8)  - configure laptop mode to switch syslog.conf ...
logger (1)           - a shell command interface to the syslog(3) ...
syslog-facility (8)  - Setup and remove LOCALx facility for sysklogd
syslog.conf (5)      - syslogd(8) configuration file
syslogd (8)          - Linux system logging utilities.
syslogd-listfiles (8) - list system logfiles
paul@laika:~$
```

By now you will have noticed the numbers between the round brackets. **man man** will explain to you that these are section numbers. If you want to know more, RTFM (Read The Fantastic Manual). *Unfortunately, manual pages do not have the answer to everything...*

```
paul@laika:~$ man woman
No manual entry for woman
```

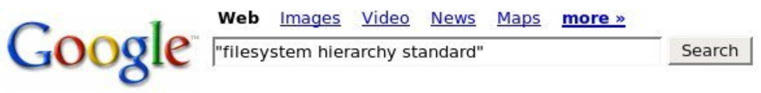
1.5.2. Red Hat Manuals online

Red Hat has a lot of info online at http://www.redhat.com/docs/manuals/ in both pdf and html format. *Unfortunately, the information there is not always up to date.*

1.5.3. Searching the internet with Google

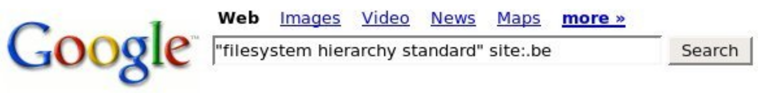
Google is a powerful tool to find help about Unix, or anything else. Here are some tricks.

Look for phrases instead of single words.



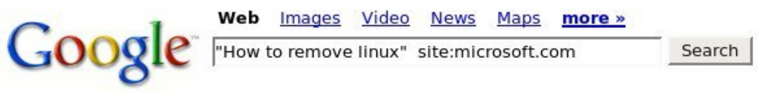
"filesystem hierarchy standard"

Search only pages from the .be TLD (or substitute .be for any other Top Level Domain). You can also use "country:be" to search only pages from Belgium (based on ip rather than TLD).



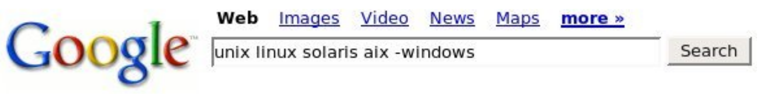
"filesystem hierarchy standard" site:.be

Search for pages inside one domain



"How to remove linux" site:microsoft.com

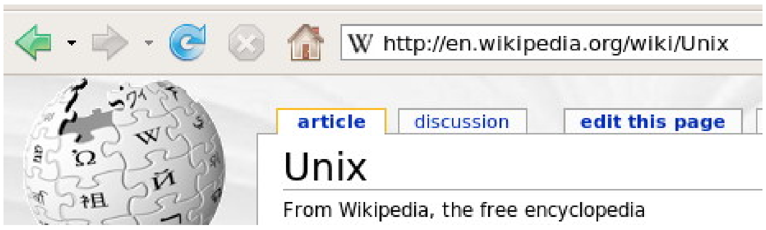
Search for pages **not** containing some words.



unix linux solaris aix -windows

1.5.4. Wikipedia

Wikipedia is a web-based, free-content encyclopedia. Its growth the past two years has been astonishing. You have a good chance of finding a clear explanation by typing your search term behind **http://en.wikipedia.org/wiki/** like this example shows.



http://en.wikipedia.org/wiki/Unix

Unix

From Wikipedia, the free encyclopedia

1.5.5. The Linux Documentation Project

On **www.tldp.org** you will find a lot of documentation, faqs, howtos and man pages about Linux and many other programs running on Linux.

1.5.6. This book

This book is available in .pdf format. Download it at http://www.linux-training.be .

1.6. Discovering the classroom

It is time now to take a look at what we have in this classroom. Students should be able to log on to one or more (virtual) Linux computers and test connectivity to each other and to the internet.

Chapter 2. First Steps

2.1. Working with directories

To explore the Linux filetree, you will need some tools. Here's a small overview of the most common commands to work with directories. These commands are available on any Linux system.

2.1.1. pwd

The **you are here** sign can be displayed with the **pwd** command (Print Working Directory). Go ahead, try it: open a commandline interface (like gnome-terminal, konsole, xterm or a tty) and type **pwd**. The tool displays your **current directory**.

```
paul@laika:~$ pwd
/home/paul
```

2.1.2. cd

You can change your current directory with the **cd** command (Change Directory).

```
paul@laika$ cd /etc
paul@laika$ pwd
/etc
paul@laika$ cd /bin
paul@laika$ pwd
/bin
paul@laika$ cd /home/paul/
paul@laika$ pwd
/home/paul
```

2.1.2.1. cd ~

You can pull off a trick with cd. Just typing **cd** without a target directory, will put you in your home directory. Typing **cd ~** has the same effect.

```
paul@laika$ cd /etc
paul@laika$ pwd
/etc
paul@laika$ cd
paul@laika$ pwd
/home/paul
paul@laika$ cd ~
paul@laika$ pwd
/home/paul
```

2.1.2.2. cd ..

To go to the **parent directory** (the one just above your current directory in the directory tree), type **cd ..** .

```
paul@laika$ pwd
/usr/share/games
paul@laika$ cd ..
paul@laika$ pwd
/usr/share
paul@laika$ cd ..
paul@laika$ cd ..
paul@laika$ pwd
/
```

*To stay in the current directory, type **cd .** ;-)*

2.1.2.3. cd -

Another useful shortcut with cd is to just type **cd -** to go to the previous directory.

```
paul@laika$ pwd
/home/paul
paul@laika$ cd /etc
paul@laika$ pwd
/etc
paul@laika$ cd -
/home/paul
paul@laika$ cd -
/etc
```

2.1.2.4. absolute and relative paths

You should be aware of **absolute and relative paths** in the filetree. When you type a path starting with a slash, then the root of the filetree is assumed. If you don't start your path with a slash, then the current directory is the assumed starting point.

The screenshot below first shows the current directory (/home/paul). From within this directory, you have to type **cd /home** instead of **cd home** to go to the /home directory.

```
paul@laika$ pwd
/home/paul
paul@laika$ cd home
bash: cd: home: No such file or directory
paul@laika$ cd /home
paul@laika$ pwd
/home
```

When inside /home, you have to type **cd paul** instead of **cd /paul** to enter the subdirectory paul of the current directory /home.

```
paul@laika$ pwd
/home
paul@laika$ cd /paul
bash: cd: /paul: No such file or directory
paul@laika$ cd paul
paul@laika$ pwd
/home/paul
```

In case your current directory is the root directory, then both **cd /home** and **cd home** will get you in the /home directory.

```
paul@laika$ cd /
paul@laika$ pwd
/
paul@laika$ cd home
paul@laika$ pwd
/home
paul@laika$ cd /
paul@laika$ pwd
/
paul@laika$ cd /home
paul@laika$ pwd
/home
```

This was the last screenshot with pwd statements. From now on, the current directory will often be displayed in the prompt. We will explain later in this book, how the shell variable $PS1 can be configured to do this.

2.1.3. ls

You can list the contents of a directory with **ls**.

```
paul@pasha:~$ ls
allfiles.txt  dmesg.txt  httpd.conf  stuff  summer.txt
paul@pasha:~$
```

2.1.3.1. ls -a

A frequently used option with ls is **-a** to show all files. All files means including the **hidden files**. When a filename on a Unix file system starts with a dot, it is considered a hidden file, and it doesn't show up in regular file listings.

```
paul@pasha:~$ ls
allfiles.txt  dmesg.txt  httpd.conf  stuff  summer.txt
```

```
paul@pasha:~$ ls -a
.   allfiles.txt   .bash_profile  dmesg.txt   .lesshst  stuff
..  .bash_history  .bashrc        httpd.conf  .ssh      summer.txt
paul@pasha:~$
```

2.1.3.2. ls -l

Many times you will be using options with ls to display the contents of the directory in different formats, or to display different parts of the directory. Just typing ls gives you a list of files in the directory. Typing **ls -l** (that is a letter L, not the number 1) gives you a long listing (more information on the contents).

```
paul@pasha:~$ ls -l
total 23992
-rw-r--r-- 1 paul paul 24506857 2006-03-30 22:53 allfiles.txt
-rw-r--r-- 1 paul paul    14744 2006-09-27 11:45 dmesg.txt
-rw-r--r-- 1 paul paul     8189 2006-03-31 14:01 httpd.conf
drwxr-xr-x 2 paul paul     4096 2007-01-08 12:22 stuff
-rw-r--r-- 1 paul paul        0 2006-03-30 22:45 summer.txt
```

2.1.3.3. ls -lh

Another frequently used ls option is **-h**. It shows the numbers (file sizes) in a more human readable format. Also shown below is some variation in the way you can give the options to ls. We will explain the details of the output later in this book.

```
paul@pasha:~$ ls -l -h
total 24M
-rw-r--r-- 1 paul paul  24M 2006-03-30 22:53 allfiles.txt
-rw-r--r-- 1 paul paul  15K 2006-09-27 11:45 dmesg.txt
-rw-r--r-- 1 paul paul 8.0K 2006-03-31 14:01 httpd.conf
drwxr-xr-x 2 paul paul 4.0K 2007-01-08 12:22 stuff
-rw-r--r-- 1 paul paul    0 2006-03-30 22:45 summer.txt
paul@pasha:~$ ls -lh
total 24M
-rw-r--r-- 1 paul paul  24M 2006-03-30 22:53 allfiles.txt
-rw-r--r-- 1 paul paul  15K 2006-09-27 11:45 dmesg.txt
-rw-r--r-- 1 paul paul 8.0K 2006-03-31 14:01 httpd.conf
drwxr-xr-x 2 paul paul 4.0K 2007-01-08 12:22 stuff
-rw-r--r-- 1 paul paul    0 2006-03-30 22:45 summer.txt
paul@pasha:~$ ls -hl
total 24M
-rw-r--r-- 1 paul paul  24M 2006-03-30 22:53 allfiles.txt
-rw-r--r-- 1 paul paul  15K 2006-09-27 11:45 dmesg.txt
-rw-r--r-- 1 paul paul 8.0K 2006-03-31 14:01 httpd.conf
drwxr-xr-x 2 paul paul 4.0K 2007-01-08 12:22 stuff
-rw-r--r-- 1 paul paul    0 2006-03-30 22:45 summer.txt
paul@pasha:~$ ls -h -l
total 24M
-rw-r--r-- 1 paul paul  24M 2006-03-30 22:53 allfiles.txt
-rw-r--r-- 1 paul paul  15K 2006-09-27 11:45 dmesg.txt
-rw-r--r-- 1 paul paul 8.0K 2006-03-31 14:01 httpd.conf
drwxr-xr-x 2 paul paul 4.0K 2007-01-08 12:22 stuff
-rw-r--r-- 1 paul paul    0 2006-03-30 22:45 summer.txt
```

2.1.4. mkdir

Walking around the Unix filetree is fun, but it is even more fun to create your own directories with **mkdir**. You have to give at least one parameter to **mkdir**, the name of the new directory to be created. Think before you type a leading / .

```
paul@laika:~$ mkdir MyDir
paul@laika:~$ cd MyDir
paul@laika:~/MyDir$ ls -al
total 8
drwxr-xr-x  2 paul paul 4096 2007-01-10 21:13 .
drwxr-xr-x 39 paul paul 4096 2007-01-10 21:13 ..
paul@laika:~/MyDir$ mkdir stuff
paul@laika:~/MyDir$ mkdir otherstuff
paul@laika:~/MyDir$ ls -l
total 8
drwxr-xr-x 2 paul paul 4096 2007-01-10 21:14 otherstuff
drwxr-xr-x 2 paul paul 4096 2007-01-10 21:14 stuff
paul@laika:~/MyDir$
```

2.1.4.1. mkdir -p

When given the option **-p**, then mkdir will create parent directories as needed.

```
paul@laika:~$ mkdir -p MyDir2/MySubdir2/ThreeDeep
paul@laika:~$ ls MyDir2
MySubdir2
paul@laika:~$ ls MyDir2/MySubdir2
ThreeDeep
paul@laika:~$ ls MyDir2/MySubdir2/ThreeDeep/
```

2.1.5. rmdir

When a directory is empty, you can use **rmdir** to remove the directory.

```
paul@laika:~/MyDir$ rmdir otherstuff
paul@laika:~/MyDir$ ls
stuff
paul@laika:~/MyDir$ cd ..
paul@laika:~$ rmdir MyDir
rmdir: MyDir/: Directory not empty
paul@laika:~$ rmdir MyDir/stuff
paul@laika:~$ rmdir MyDir
```

2.1.5.1. rmdir -p

And similar to the mkdir -p option, you can also use rmdir to recursively remove directories.

```
paul@laika:~$ mkdir -p dir/subdir/subdir2
```

```
paul@laika:~$ rmdir -p dir/subdir/subdir2
paul@laika:~$
```

2.2. Practice: Working with directories

1. Display your current directory.

2. Change to the /etc directory.

3. Now change to your home directory using only three key presses.

4. Change to the /boot/grub directory using only eleven key presses.

5. Go to the parent directory of the current directory.

6. Go to the root directory.

7. List the contents of the root directory.

8. List a long listing of the root directory.

9. Stay where you are, and list the contents of /etc.

10. Stay where you are, and list the contents of /bin and /sbin.

11. Stay where you are, and list the contents of ~.

12. List all the files (including hidden files) in your homedirectory.

13. List the files in /boot in a human readable format.

14. Create a directory testdir in your homedirectory.

15. Change to the /etc directory, stay here and create a directory newdir in your homedirectory.

16. Create in one command the directories ~/dir1/dir2/dir3 (dir3 is a subdirectory from dir2, and dir2 is a subdirectory from dir1).

17. Remove the directory testdir.

18. If time permits (or if you are waiting for other students to finish this practice), use and understand pushd and popd. Use the man page of bash to find information about pushd, popd and dirs.

2.3. Solution: Working with directories

1. Display your current directory.

```
pwd
```

2. Change to the /etc directory.

```
cd /etc
```

3. Now change to your home directory using only three key presses.

```
cd (and the enter key)
```

4. Change to the /boot/grub directory using only eleven key presses.

```
cd /boot/grub (use the tab key)
```

5. Go to the parent directory of the current directory.

```
cd .. (with space between cd and ..)
```

6. Go to the root directory.

```
cd /
```

7. List the contents of the root directory.

```
ls
```

8. List a long listing of the root directory.

```
ls -l
```

9. Stay where you are, and list the contents of /etc.

```
ls /etc
```

10. Stay where you are, and list the contents of /bin and /sbin.

```
ls /bin /sbin
```

11. Stay where you are, and list the contents of ~.

```
ls ~
```

12. List all the files (including hidden files) in your homedirectory.

```
ls -al ~
```

13. List the files in /boot in a human readable format.

```
ls -lh /boot
```

14. Create a directory testdir in your homedirectory.

```
mkdir ~/testdir
```

15. Change to the /etc directory, stay here and create a directory newdir in your homedirectory.

```
cd /etc ; mkdir ~/newdir
```

16. Create in one command the directories ~/dir1/dir2/dir3 (dir3 is a subdirectory from dir2, and dir2 is a subdirectory from dir1).

```
mkdir -p ~/dir1/dir2/dir3
```

17. Remove the directory testdir.

```
rmdir testdir
```

18. If time permits (or if you are waiting for other students to finish this practice), use and understand pushd and popd. Use the man page of bash to find information about pushd, popd and dirs.

```
man bash
```

The Bash shell has two built-in commands called **pushd** and **popd**. Both commands work with a common stack of previous directories. Pushd adds a directory to the stack and changes to a new current directory, popd removes a directory from the stack and sets the current directory.

```
paul@laika:/etc$ cd /bin
paul@laika:/bin$ pushd /lib
/lib /bin
paul@laika:/lib$ pushd /proc
/proc /lib /bin
paul@laika:/proc$
paul@laika:/proc$ popd
/lib /bin
paul@laika:/lib$
paul@laika:/lib$
paul@laika:/lib$ popd
/bin
paul@laika:/bin$
```

2.4. Working with files

2.4.1. file

The **file** utility determines the file type. Linux does not use extensions to determine the file type. Your editor does not care whether a file ends in .TXT or .DOC. As a system administrator, you should use the **file** command to determine the file type. First some examples on a typical Linux system.

```
paul@laika:~$ file pic33.png
pic33.png: PNG image data, 3840 x 1200, 8-bit/color RGBA, non-interlaced
paul@laika:~$ file /etc/passwd
/etc/passwd: ASCII text
paul@laika:~$ file HelloWorld.c
HelloWorld.c: ASCII C program text
```

Here's another example of the file utility. It shows the different type of binaries on different architectures.

```
# Solaris 9 on Intel
bash-2.05$ file /bin/date
/bin/date:      ELF 32-bit LSB executable 80386 Version 1, dynamically \
linked, stripped

# Ubuntu Linux on AMD64
paul@laika:~$ file /bin/date
/bin/date: ELF 64-bit LSB executable, AMD x86-64, version 1 (SYSV), for\
 GNU/Linux 2.6.0, dynamically linked (uses shared libs), for GNU/Linux \
2.6.0, stripped

# Debian Sarge on SPARC
paul@pasha:~$ file /bin/date
/bin/date: ELF 32-bit MSB executable, SPARC, version 1 (SYSV), for GNU/\
Linux 2.4.1, dynamically linked (uses shared libs), for GNU/Linux 2.4.1\
, stripped
```

The file command uses a magic file that contains patterns to recognize filetypes. The magic file is located in **/usr/share/file/magic**. Type **man 5 magic** for more information.

2.4.2. touch

One easy way to create a file is with **touch**. (We will see many other ways for creating files later in this book.)

```
paul@laika:~/test$ touch file1
paul@laika:~/test$ touch file2
paul@laika:~/test$ touch file555
paul@laika:~/test$ ls -l
total 0
-rw-r--r-- 1 paul paul 0 2007-01-10 21:40 file1
-rw-r--r-- 1 paul paul 0 2007-01-10 21:40 file2
-rw-r--r-- 1 paul paul 0 2007-01-10 21:40 file555
```

2.4.2.1. touch -t

Of course, touch can do more than just create files. Can you find out what by looking at the next screenshot ? If not, check the manual of touch.

```
paul@laika:~/test$ touch -t 200505050000 SinkoDeMayo
paul@laika:~/test$ touch -t 130207111630 BigBattle
paul@laika:~/test$ ls -l
total 0
-rw-r--r-- 1 paul paul 0 1302-07-11 16:30 BigBattle
-rw-r--r-- 1 paul paul 0 2005-05-05 00:00 SinkoDeMayo
```

2.4.3. rm

When you no longer need a file, use **rm** to remove it. Unlike some graphical user interfaces, the command line in general does not have a *waste bin* or *trashcan* to recover files. When you use rm to remove a file, the file is gone. So be careful before removing files!

```
paul@laika:~/test$ ls
BigBattle  SinkoDeMayo
paul@laika:~/test$ rm BigBattle
paul@laika:~/test$ ls
SinkoDeMayo
```

2.4.3.1. rm -i

To prevent yourself from accidentally removing a file, you can type **rm -i**.

```
paul@laika:~/Linux$ touch brel.txt
paul@laika:~/Linux$ rm -i brel.txt
rm: remove regular empty file `brel.txt'? y
paul@laika:~/Linux$
```

2.4.3.2. rm -rf

By default, rm will not remove non-empty directories. However rm accepts several options that will allow you to remove any directory. The **rm -rf** statement is famous because it will erase anything (providing that you have the permissions to do so). When you are logged on as root, be very careful with **rm -rf** (the **f** means **force** and the **r** means **recursive**), because being root implies that permissions don't apply to you, so you can literally erase your entire system by accident.

```
paul@laika:~$ ls test
SinkoDeMayo
paul@laika:~$ rm test
rm: cannot remove `test': Is a directory
paul@laika:~$ rm -rf test
paul@laika:~$ ls test
ls: test: No such file or directory
```

2.4.4. cp

To copy a file, use **cp** with a source and a target argument. If the target is a directory, then the sourcefiles are copied in that target directory.

```
paul@laika:~/test$ touch FileA
paul@laika:~/test$ ls
```

```
FileA
paul@laika:~/test$ cp FileA FileB
paul@laika:~/test$ ls
FileA  FileB
paul@laika:~/test$ mkdir MyDir
paul@laika:~/test$ ls
FileA  FileB  MyDir
paul@laika:~/test$ cp FileA MyDir/
paul@laika:~/test$ ls MyDir/
FileA
```

2.4.4.1. cp -r

To copy complete directories, use **cp -r** (the **-r** option forces **recursive** copying of all files in all subdirectories).

```
paul@laika:~/test$ ls
FileA  FileB  MyDir
paul@laika:~/test$ ls MyDir/
FileA
paul@laika:~/test$ cp -r MyDir MyDirB
paul@laika:~/test$ ls
FileA  FileB  MyDir  MyDirB
paul@laika:~/test$ ls MyDirB
FileA
```

2.4.4.2. cp multiple files to directory

You can also use cp to copy multiple file into a directory. In that case, the last argument (aka the target) must be a directory.

```
cp file1 file2 dir1/file3 dir1/file55 dir2
```

2.4.4.3. cp -i

To prevent cp from overwriting existing files, use the -i (for interactive) option.

```
paul@laika:~/test$ cp fire water
paul@laika:~/test$ cp -i fire water
cp: overwrite `water'? no
paul@laika:~/test$
```

2.4.4.4. cp -p

To preserve permissions and time stamps from source files, use **cp -p**.

```
paul@laika:~/perms$ cp file* cp
paul@laika:~/perms$ cp -p file* cpp
paul@laika:~/perms$ ll *
```

```
-rwx------ 1 paul paul    0 2008-08-25 13:26 file33
-rwxr-x--- 1 paul paul    0 2008-08-25 13:26 file42

cp:
total 0
-rwx------ 1 paul paul 0 2008-08-25 13:34 file33
-rwxr-x--- 1 paul paul 0 2008-08-25 13:34 file42

cpp:
total 0
-rwx------ 1 paul paul 0 2008-08-25 13:26 file33
-rwxr-x--- 1 paul paul 0 2008-08-25 13:26 file42
```

2.4.5. mv

Use **mv** to rename a file, or to move the file to another directory.

```
paul@laika:~/test$ touch file100
paul@laika:~/test$ ls
file100
paul@laika:~/test$ mv file100 ABC.txt
paul@laika:~/test$ ls
ABC.txt
paul@laika:~/test$
```

When you need to rename only one file, then **mv** is the prefered command to use.

2.4.6. rename

The **rename** command can also be used, but it has a more complex syntax to enable renaming of many files at once. Below two examples, the first switches all occurrences of txt in png for all filenames ending in .txt. The second example switches all occurrences of uppercase ABC in lowercase abc for all filenames ending in .png . The following syntax will work on debian and ubuntu (prior to Ubuntu 7.10).

```
paul@laika:~/test$ ls
123.txt  ABC.txt
paul@laika:~/test$ rename 's/txt/png/' *.txt
paul@laika:~/test$ ls
123.png  ABC.png
paul@laika:~/test$ rename 's/ABC/abc/' *.png
paul@laika:~/test$ ls
123.png  abc.png
paul@laika:~/test$
```

On Red Hat Enterprise Linux (and many other Linux distro's like Ubuntu 8.04), the syntax of rename is a bit different. The first example below renames all *.conf files, replacing any occurrence of conf with bak. The second example renames all(*) files, replacing one with ONE.

```
[paul@RHEL4a test]$ ls
```

```
one.conf  two.conf
[paul@RHEL4a test]$ rename conf bak *.conf
[paul@RHEL4a test]$ ls
one.bak  two.bak
[paul@RHEL4a test]$ rename one ONE *
[paul@RHEL4a test]$ ls
ONE.bak  two.bak
[paul@RHEL4a test]$
```

2.5. Practice: Working with files

1. List the files in the /bin directory

2. Display the type of file of /bin/cat, /etc/passwd and /usr/bin/passwd.

3a. Download wolf.jpg and book.pdf from http://www.linux-training.be (wget http://www.linux-training.be/studentfiles/wolf.jpg)

3b. Display the type of file of wolf.jpg and book.pdf

3c. Rename wolf.jpg to wolf.pdf (use mv).

3d. Display the type of file of wolf.pdf and book.pdf.

4. Create a directory ~/touched and enter it.

5. Create the files today.txt and yesterday.txt in touched.

6. Change the date on yesterday.txt to match yesterday's date.

7. Copy yesterday.txt to copy.yesterday.txt

8. Rename copy.yesterday.txt to kim

9. Create a directory called ~/testbackup and copy all files from ~/touched in it.

10. Use one command to remove the directory ~/testbackup and all files in it.

11. Create a directory ~/etcbackup and copy all *.conf files from /etc in it. Did you include all subdirectories of /etc ?

12. Use rename to rename all *.bak files to *.backup . (if you have more than one distro available, try it on all!)

2.6. Solution: Working with files

1. List the files in the /bin directory

```
ls /bin
```

2. Display the type of file of /bin/cat, /etc/passwd and /usr/bin/passwd.

```
file /bin/cat /etc/passwd /usr/bin/passwd
```

3a. Download wolf.jpg and book.pdf from http://www.linux-training.be (wget http://www.linux-training.be/studentfiles/wolf.jpg)

```
wget http://www.linux-training.be/studentfiles/wolf.jpg
```

```
wget http://www.linux-training.be/studentfiles/book.pdf
```

3b. Display the type of file of wolf.jpg and book.pdf

```
file wolf.jpg wolf.pdf
```

3c. Rename wolf.jpg to wolf.pdf (use mv).

```
mv wolf.jpg wolf.pdf
```

3d. Display the type of file of wolf.pdf and book.pdf.

```
file wolf.pdf book.pdf
```

4. Create a directory ~/touched and enter it.

```
mkdir ~/touched ; cd ~/touched
```

5. Create the files today.txt and yesterday.txt in touched.

```
touch today.txt yesterday.txt
```

6. Change the date on yesterday.txt to match yesterday's date.

```
touch -t 200810251405 yesterday (substitute 20081025 with yesterdays date)
```

7. Copy yesterday.txt to copy.yesterday.txt

```
cp yesterday.txt copy.yesterday.txt
```

8. Rename copy.yesterday.txt to kim

```
mv copy.yesterday.txt kim
```

9. Create a directory called ~/testbackup and copy all files from ~/touched in it.

```
mkdir ~/testbackup ; cp -r ~/touched ~/testbackup/
```

10. Use one command to remove the directory ~/testbackup and all files in it.

```
rm -rf ~/testbackup
```

11. Create a directory ~/etcbackup and copy all *.conf files from /etc in it. Did you include all subdirectories of /etc ?

```
cp -r /etc/*.conf ~/etcbackup
```

```
Only *.conf files that are directly in /etc/ are copied.
```

12. Use rename to rename all *.bak files to *.backup . (if you have more than one distro available, try it on all!)

```
On RHEL: touch 1.bak 2.bak ; rename bak backup *.bak
```

```
On Debian: touch 1.bak 2.bak ; rename 's/bak/backup/' *.bak
```

2.7. File contents

2.7.1. head

You can use **head** to display the first ten lines of a file.

```
paul@laika:~$ head /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/bin/sh
man:x:6:12:man:/var/cache/man:/bin/sh
lp:x:7:7:lp:/var/spool/lpd:/bin/sh
mail:x:8:8:mail:/var/mail:/bin/sh
news:x:9:9:news:/var/spool/news:/bin/sh
paul@laika:~$
```

The head command can also display the first n lines of a file.

```
paul@laika:~$ head -4 /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
```

Head can also display the first n bytes.

```
paul@laika:~$ head -c4 /etc/passwd
rootpaul@laika:~$
```

2.7.2. tail

Similar to head, the **tail** command will display the last ten lines of a file.

```
paul@laika:~$ tail /etc/services
vboxd           20012/udp
binkp           24554/tcp               # binkp fidonet protocol
asp             27374/tcp               # Address Search Protocol
asp             27374/udp
csync2          30865/tcp             # cluster synchronization tool
dircproxy       57000/tcp             # Detachable IRC Proxy
tfido           60177/tcp             # fidonet EMSI over telnet
fido            60179/tcp             # fidonet EMSI over TCP

# Local services
paul@laika:~$
```

The tail command has other useful options, we will use some of them during this course.

2.7.3. cat

The **cat** command is one of the most universal tools. All it does is copying standard input to standard output, but in combination with the shell, this can be very powerful and diverse. Some examples will give a glimpse of the possibilities. The first example is simple, you can use cat to display a file on the screen. If the file is longer than the screen, it will scroll by until the end.

```
paul@laika:~$ cat /etc/resolv.conf
nameserver 194.7.1.4
paul@laika:~$
```

2.7.3.1. concatenate

cat is short for **concatenate**. One of the basic uses of cat is to concatenate files into a bigger (or complete) file.

```
paul@laika:~$ echo one > part1
paul@laika:~$ echo two > part2
paul@laika:~$ echo three > part3
paul@laika:~$ cat part1 part2 part3
one
two
three
paul@laika:~$
```

2.7.3.2. create files

You can use cat to create files with one or more lines of text. Just type the command as is shown in the screenshot below. Then type one or more lines, finish each line with the enter key. After the last line, type and hold the Control (Ctrl) key and press d. The **Ctrl d** key combination will send an EOF (End of File) to the running process, this will end the cat command.

```
paul@laika:~/test$ cat > winter.txt
It is very cold today!
```

```
paul@laika:~/test$ cat winter.txt
It is very cold today!
paul@laika:~/test$
```

You can actually choose this end marker for cat with **<<** as is shown in this screenshot.

```
paul@laika:~/test$ cat > hot.txt <<stop
> It is hot today!
> Yes it is summer.
> stop
paul@laika:~/test$ cat hot.txt
It is hot today!
Yes it is summer.
paul@laika:~/test$
```

2.7.3.3. copy files

In the third example you will see that cat can be used to copy files. We will explain in detail what happens here in the bash shell chapter.

```
paul@laika:~/test$ cat winter.txt
It is very cold today!
paul@laika:~/test$ cat winter.txt > cold.txt
paul@laika:~/test$ cat cold.txt
It is very cold today!
paul@laika:~/test$
```

2.7.4. tac

Just one example will show you the purpose of **tac** (as the opposite of cat).

```
paul@laika:~/test$ cat count
one
two
three
four
paul@laika:~/test$ tac count
four
three
two
one
paul@laika:~/test$
```

2.7.5. more and less

The **more** command is useful for displaying files that take up more than one screen. More will allow you to see the contents of the file page by page. You can use the spacebar to see the next page, or q to quit more. Some people prefere the **less** command instead of more.

2.7.6. strings

With the **strings** command you can display readable ascii strings found in (binary) files. This example locates the ls binary, and then displays readable strings in the binary file (output is truncated).

```
paul@laika:~$ which ls
/bin/ls
paul@laika:~$ strings /bin/ls
/lib/ld-linux.so.2
librt.so.1
__gmon_start__
_Jv_RegisterClasses
clock_gettime
libacl.so.1
...
```

2.8. Practice: File contents

1. Display the first 12 lines of /etc/X11/xorg.conf.

2. Display the last line of /etc/passwd.

3. Use cat to create a file named count.txt that looks like this:

```
One
Two
Three
Four
Five
```

4. Use cp to make a backup of this file to cnt.txt.

5. Use cat to make a backup of this file to catcnt.txt

6. Display catcnt.txt, but with all lines in reverse order (the last line first).

7. Use more to display /var/log/messages.

8. Display the readable character strings from the /usr/bin/passwd command.

9. Use ls to find the biggest file in /etc.

2.9. Solution: File contents

1. Display the first 12 lines of /etc/X11/xorg.conf.

```
head -12 /etc/X11/xorg.conf
```

2. Display the last line of /etc/passwd.

```
tail -1 /etc/passwd
```

3. Use cat to create a file named count.txt that looks like this:

```
One
Two
Three
Four
Five
```

```
cat > count.txt
```

4. Use cp to make a backup of this file to cnt.txt.

```
cp count.txt cnt.txt
```

5. Use cat to make a backup of this file to catcnt.txt

```
cat count.txt > catcnt.txt
```

6. Display catcnt.txt, but with all lines in reverse order (the last line first).

```
tac catcnt.txt
```

7. Use more to display /var/log/messages.

```
more /var/log/messages
```

8. Display the readable character strings from the /usr/bin/passwd command.

```
strings /usr/bin/passwd
```

9. Use ls to find the biggest file in /etc.

```
cd ; ls -lrS /etc
```

Chapter 3. The Linux File system Tree

3.1. About files on Linux

3.1.1. case sensitive

Linux is **case sensitive**, this means that FILE1 is different from file1, and /etc/hosts is different from /etc/Hosts (the latter one does not exist on a typical Linux computer).

This screenshot points to the difference between two files, one with uppercase W, the other with lowercase w.

```
paul@laika:~/Linux$ ls
winter.txt  Winter.txt
paul@laika:~/Linux$ cat winter.txt
It is cold.
paul@laika:~/Linux$ cat Winter.txt
It is very cold!
```

3.1.2. everything is a file

A directory is a special kind of file, but it is still a file. Even a terminal window or a hard disk are represented somewhere in de file system hierarchy by a file. It will become clear troughout this course that everything on Linux is a file.

3.1.3. root directory

All Linux systems have a directory structure that starts at the **root directory**. The root directory is represented by a slash, like this: / . Everything that exists on your linux system can be found below this root directory. Let's take a brief look at the contents of the root directory.

```
[paul@RHELv4u3 ~]$ ls /
bin   dev  home  media  mnt  proc  sbin     srv  tftpboot  usr
boot  etc  lib   misc   opt  root  selinux  sys  tmp       var
[paul@RHELv4u3 ~]$
```

3.1.4. man hier (FileSystem Hierarchy)

There are some differences between Linux distributions. For help about your machine, enter **man hier** to find information about the file system hierarchy. This manual will explain the directory structure on your computer.

3.1.5. Filesystem Hierarchy Standard

Red Hat Enterprise Linux, Fedora, Novell SLES/SLED, OpenSUSE and others (even Sun Solaris) all aim to follow the **Filesystem Hierarchy Standard** (FHS). Maybe the FHS will make more Unix file system trees unite in the future. The **FHS** is available online at http://www.pathname.com/fhs/.

3.2. Filesystem Hierarchy

On http://www.pathname.com/fhs/ we read "The filesystem hierarchy standard has been designed to be used by Unix distribution developers, package developers, and system implementors. However, it is primarily intended to be a reference and is not a tutorial on how to manage a Unix filesystem or directory hierarchy." Below we will discuss a couple of root directories. *For a complete reference, you'll have to check with every developer and system administrator in the world ;-)*

3.2.1. /bin binaries

The **/bin** directory contains binaries for use by all users. According to the FHS /bin/date should exist, and /bin should contain /bin/cat. You will find a bin subdirectory in many other directories. Binaries are sometimes called **executables**. In the screenshot below you see a lot of common unix commands like cat, cp, cpio, date, dd, echo, grep and so on. A lot of these will be covered in this book.

```
paul@laika:~$ ls /bin
archdetect       egrep            mt               setupcon
autopartition    false            mt-gnu           sh
bash             fgconsole        mv               sh.distrib
bunzip2          fgrep            nano             sleep
bzcat            fuser            nc               stralign
bzcmp            fusermount       nc.traditional   stty
bzdiff           get_mountoptions netcat           su
bzegrep          grep             netstat          sync
bzexe            gunzip           ntfs-3g          sysfs
bzfgrep          gzexe            ntfs-3g.probe    tailf
bzgrep           gzip             parted_devices   tar
bzip2            hostname         parted_server    tempfile
bzip2recover     hw-detect        partman          touch
bzless           ip               partman-commit   true
bzmore           kbd_mode         perform_recipe   ulockmgr
cat              kill             pidof            umount
...
```

3.2.2. /boot static files to boot the system

The **/boot** directory contains all files needed to boot the computer. These files don't change very often. On Linux systems you typically find the **/boot/grub** directory here. This /boot/grub contains **/boot/grub/menu.lst** (the grub configuration file is often linked to **/boot/grub/grub.conf**), which defines the bootmenu that is being displayed before the kernel starts.

3.2.3. /dev device files

Device files in **/dev** appear to be ordinary files, but are not located on the harddisk. The /dev directory is populated with files when the kernel is recognizing hardware.

3.2.3.1. Common physical devices

Common hardware such as hard disk devices are represented by device files in **/dev**. Below a screenshot of SATA device files on a laptop and then IDE attached drives on a desktop. (The detailed meaning of these devices will be discussed later.)

```
#
# SATA or SCSI
#
paul@laika:~$ ls /dev/sd*
/dev/sda  /dev/sda1  /dev/sda2  /dev/sda3  /dev/sdb  /dev/sdb1  /dev/sdb2

#
# IDE or ATAPI
#
paul@barry:~$ ls /dev/hd*
/dev/hda  /dev/hda1  /dev/hda2  /dev/hdb  /dev/hdb1  /dev/hdb2  /dev/hdc
```

Besides representing physical hardware, some device files are special. These special devices can be very useful.

3.2.3.2. /dev/tty and /dev/pts

For example **/dev/tty1** represents a terminal or console attached to the system. (Don't break your head on the exact terminology of 'terminal' or 'console', what we mean here is a commandline interface.) When typing commands in a terminal that is part of a graphical interface like Gnome or KDE, then your terminal will be represented as **/dev/pts/1** (1 can be another number).

3.2.3.3. /dev/null

On Linux you will find special devices like **/dev/null** which can be considered a black hole, it has unlimited storage, but nothing can be retrieved from it. Technically speaking, anything given to /dev/null will be discarded. /dev/null can be useful to discard unwanted output from commands. */dev/null is not a good location to store all your backups ;-)*.

3.2.4. /etc Configuration Files

All of the machine-specific configuration files should be located in **/etc**. Many times the name of a configuration files is the same as the application or daemon or protocol with .conf added as an extension. But there is much more to be found in /etc.

```
paul@laika:~$ ls /etc/*.conf
/etc/adduser.conf        /etc/ld.so.conf       /etc/scrollkeeper.conf
/etc/brltty.conf         /etc/lftp.conf        /etc/sysctl.conf
/etc/ccertificates.conf  /etc/libao.conf       /etc/syslog.conf
/etc/cvs-cron.conf       /etc/logrotate.conf   /etc/ucf.conf
/etc/ddclient.conf       /etc/ltrace.conf      /etc/uniconf.conf
/etc/debconf.conf        /etc/mke2fs.conf      /etc/updatedb.conf
/etc/deluser.conf        /etc/netscsid.conf    /etc/usplash.conf
/etc/fdmount.conf        /etc/nsswitch.conf    /etc/uswsusp.conf
/etc/hdparm.conf         /etc/pam.conf         /etc/vnc.conf
/etc/host.conf           /etc/pnm2ppa.conf     /etc/wodim.conf
/etc/inetd.conf          /etc/povray.conf      /etc/wvdial.conf
/etc/kernel-img.conf     /etc/resolv.conf
paul@laika:~$
```

3.2.4.1. /etc/X11/

The graphical display (aka **X Window System** or just **X**) is driven by software from the X.org foundation. The configuration file for your graphical display is **/etc/X11/xorg.conf**.

3.2.4.2. /etc/filesystems

When mounting a file system without specifying explicitly the file system, then **mount** will first probe **/etc/filesystems**. Mount will skip lines with the **nodev** directive, and should this file end with a single * on the last line, then mount will continue probing **/proc/filesystems**.

```
paul@RHELv4u4:~$ cat /etc/filesystems
ext3
ext2
nodev proc
nodev devpts
iso9660
vfat
hfs
paul@RHELv4u4:~$
```

3.2.4.3. /etc/skel/

The **skeleton** directory **/etc/skel** is copied to the home directory of a newly created user. It usually contains hidden files like a **.bashrc** script.

3.2.4.4. /etc/sysconfig/

This directory, which is not mentioned in the FHS, contains a lot of Red Hat Enterprise Linux configuration files. We will discuss some of them in greater detail. The screenshot below is the **/etc/sysconfig** from RHELv4u4 with everything installed.

```
paul@RHELv4u4:~$ ls /etc/sysconfig/
apmd          firstboot     irda              network      saslauthd
apm-scripts   grub          irqbalance        networking   selinux
authconfig    hidd          keyboard          ntpd         spamassassin
autofs        httpd         kudzu             openib.conf  squid
bluetooth     hwconf        lm_sensors        pand         syslog
clock         i18n          mouse             pcmcia       sys-config-sec
console       init          mouse.B           pgsql        sys-config-users
crond         installinfo   named             prelink      sys-logviewer
desktop       ipmi          netdump           rawdevices   tux
diskdump      iptables      netdump_id_dsa    rhn          vncservers
dund          iptables-cfg  netdump_id_dsa.p  samba        xinetd
paul@RHELv4u4:~$
```

The file **/etc/sysconfig/firstboot** tells the Red Hat Setup Agent to not run at boot time. If you want to run the Red Hat Setup Agent at the next reboot, then simply remove this file, and run **chkconfig --level 5 firstboot on**. The Red Hat Setup Agent allows you to install the latest updates, create a user account, join the Red Hat Network and more. It will then create the /etc/sysconfig/firstboot file again.

```
paul@RHELv4u4:~$ cat /etc/sysconfig/firstboot
RUN_FIRSTBOOT=NO
```

The file **/etc/sysconfig/harddisks** contains some parameters to tune the hard disks. The file explains itself.

You can see hardware detected by **kudzu** in **/etc/sysconfig/hwconf**. Kudzu is software from Red Hat for automatic discovery and configuration of hardware.

The keyboard type and table are set in the **/etc/sysconfig/keyboard** file. For more console keyboard information, check the manual pages of **keymaps(5)**, **dumpkeys(1)**, **loadkeys(1)** and the directory **/lib/kbd/keymaps/**.

```
root@RHELv4u4:/etc/sysconfig# cat keyboard
KEYBOARDTYPE="pc"
KEYTABLE="us"
```

We will discuss the networking files in this directory in the networking chapter.

3.2.5. /home sweet home

You will find a lot of locations with an extensive hierarchy of personal or project data under **/home**. It is common practice (but not mandatory) to name the users home directory after their username in the format /home/$USERNAME. Like in this example:

```
paul@pasha:~$ ls /home
geert  guillaume  maria  paul  tom
```

Besides giving every user (or every project or group) a location to store personal files, the home directory of a user also serves as a location to store the user profile. A typical Unix user profile contains a bunch of hidden files (files who's filename starts with a dot). The hidden files of the Unix user profile contain settings specific for that user.

```
paul@pasha:~$ ls -d /home/paul/.*
/home/paul/.              /home/paul/.bash_profile  /home/paul/.ssh
/home/paul/..             /home/paul/.bashrc        /home/paul/.viminfo
/home/paul/.bash_history  /home/paul/.lesshst       /home/paul/.Xauthority
```

3.2.6. /initrd

This empty directory is used as a mount point by Red Hat Enterprise Linux during boot time. Removing it causes a kernel panic during the next boot.

3.2.7. /lib shared libraries

Binaries, like those found in /bin, often use shared libraries located in **/lib**. Below a partial screenshot of the contents of /lib.

```
paul@laika:~$ ls /lib/libc*
/lib/libc-2.5.so     /lib/libcfont.so.0.0.0  /lib/libcom_err.so.2.1
/lib/libcap.so.1     /lib/libcidn-2.5.so     /lib/libconsole.so.0
/lib/libcap.so.1.10  /lib/libcidn.so.1       /lib/libconsole.so.0.0.0
/lib/libcfont.so.0   /lib/libcom_err.so.2    /lib/libcrypt-2.5.so
```

3.2.7.1. /lib/modules

Typically, the kernel loads kernel modules from **/lib/modules**.

3.2.7.2. /lib32 and /lib64

We are now (the year 2007) in a transition between 32-bit and 64-bit systems. So you might encounter directories named **/lib32** and **/lib64**, to clarify the register size used at compilation time of the libraries. My current 64-bit laptop has some older 32-bit binaries and libraries for compatibility with legacy applications. The screenshot uses the **file** utility to point out the difference.

```
paul@laika:~$ file /lib32/libc-2.5.so
/lib32/libc-2.5.so: ELF 32-bit LSB shared object, Intel 80386, \
version 1 (SYSV), for GNU/Linux 2.6.0, stripped
paul@laika:~$ file /lib64/libcap.so.1.10
/lib64/libcap.so.1.10: ELF 64-bit LSB shared object, AMD x86-64, \
version 1 (SYSV), stripped
```

The ELF **Executable and Linkable Format** is used in almost every Unix-like operating system since System V.

3.2.8. /media for Removable Media

The **/media** directory serves as a mount point for removable media, meaning devices such as CD-ROM's, digital cameras and various usb-attached devices. Since **/media** is rather new in the Unix world, you could very well encounter systems running without this directory. Solaris 9 does not have it, Solaris 10 does.

```
paul@laika:~$ ls /media/
cdrom  cdrom0  usbdisk
```

3.2.9. /mnt standard mount point

Older Unixes (and Linuxes) used to mount all kind of stuff under /mnt/something/. According to the FHS, **/mnt** should only be used to temporarily mount something. But you will most likely witness a lot of systems with more than one directory underneath /mnt used as a mountpoint for various local and remote filesystems.

3.2.10. /opt Optional software

Most of my systems today have an empty **/opt** directory. It is considered outdated, but you might find some systems with add-on software installed in /opt. If that is the case, the package should install all its files in the typical bin, lib, etc subdirectories in /opt/$packagename/. If for example the package is called wp, then it installs in /opt/wp, putting binaries in /opt/wp/bin and manpages in /opt/wp/man. Most of the default software which comes along with the distribution, will not be installed in /opt.

3.2.11. /proc conversation with the kernel

/proc is another special directory, appearing to be ordinary files, but not taking up diskspace. It is actually a view on the kernel, or better on what the kernel sees, and a means to talk to the kernel directly. **/proc** is a proc filesystem.

```
paul@RHELv4u4:~$ mount -t proc
none on /proc type proc (rw)
```

When listing the /proc directory, you will see a lot of numbers (on any Unix), and some interesting files (on Linux)

```
mul@laika:~$ ls /proc
1      2339   4724  5418  6587  7201       cmdline      mounts
```

```
10175  2523   4729  5421  6596  7204       cpuinfo      mtrr
10211  2783   4741  5658  6599  7206       crypto       net
10239  2975   4873  5661  6638  7214       devices      pagetypeinfo
141    29775  4874  5665  6652  7216       diskstats    partitions
15045  29792  4878  5927  6719  7218       dma          sched_debug
1519   2997   4879  6     6736  7223       driver       scsi
1548   3      4881  6032  6737  7224       execdomains  self
1551   30228  4882  6033  6755  7227       fb           slabinfo
1554   3069   5     6145  6762  7260       filesystems  stat
1557   31422  5073  6298  6774  7267       fs           swaps
1606   3149   5147  6414  6816  7275       ide          sys
180    31507  5203  6418  6991  7282       interrupts   sysrq-trigger
181    3189   5206  6419  6993  7298       iomem        sysvipc
182    3193   5228  6420  6996  7319       ioports      timer_list
18898  3246   5272  6421  7157  7330       irq          timer_stats
19799  3248   5291  6422  7163  7345       kallsyms     tty
19803  3253   5294  6423  7164  7513       kcore        uptime
19804  3372   5356  6424  7171  7525       key-users    version
1987   4      5370  6425  7175  7529       kmsg         version_signature
1989   42     5379  6426  7188  9964       loadavg      vmcore
2      45     5380  6430  7189  acpi       locks        vmnet
20845  4542   5412  6450  7191  asound     meminfo      vmstat
221    46     5414  6551  7192  buddyinfo  misc         zoneinfo
2338   4704   5416  6568  7199  bus        modules
```

Let's investigate the file properties inside /proc. Looking at the date and time will display the current date and time, meaning the files are constantly updated (A view on the kernel).

```
paul@RHELv4u4:~$ date
Mon Jan 29 18:06:32 EST 2007
paul@RHELv4u4:~$ ls -al /proc/cpuinfo
-r--r--r--  1 root root 0 Jan 29 18:06 /proc/cpuinfo
paul@RHELv4u4:~$
paul@RHELv4u4:~$  ...time passes...
paul@RHELv4u4:~$
paul@RHELv4u4:~$ date
Mon Jan 29 18:10:00 EST 2007
paul@RHELv4u4:~$ ls -al /proc/cpuinfo
-r--r--r--  1 root root 0 Jan 29 18:10 /proc/cpuinfo
```

Most files in /proc are 0 bytes, yet they contain data, sometimes a lot of data. You can see this by executing cat on files like **/proc/cpuinfo**, which contains information on the CPU.

```
paul@RHELv4u4:~$ file /proc/cpuinfo
/proc/cpuinfo: empty
paul@RHELv4u4:~$ cat /proc/cpuinfo
processor       : 0
vendor_id       : AuthenticAMD
cpu family      : 15
model           : 43
model name      : AMD Athlon(tm) 64 X2 Dual Core Processor 4600+
stepping        : 1
cpu MHz         : 2398.628
cache size      : 512 KB
fdiv_bug        : no
hlt_bug         : no
f00f_bug        : no
```

```
coma_bug        : no
fpu             : yes
fpu_exception   : yes
cpuid level     : 1
wp              : yes
flags           : fpu vme de pse tsc msr pae mce cx8 apic mtrr pge...
bogomips        : 4803.54
```

Just for fun, here is /proc/cpuinfo on a Sun Sunblade 1000...

```
paul@pasha:~$ cat /proc/cpuinfo
cpu : TI UltraSparc III (Cheetah)
fpu : UltraSparc III integrated FPU
promlib : Version 3 Revision 2
prom : 4.2.2
type : sun4u
ncpus probed : 2
ncpus active : 2
Cpu0Bogo : 498.68
Cpu0ClkTck : 000000002cb41780
Cpu1Bogo : 498.68
Cpu1ClkTck : 000000002cb41780
MMU Type : Cheetah
State:
CPU0: online
CPU1: online
```

... and on a Sony Playstation 3.

```
[root@ps3 tmp]# uname -a
Linux ps3 2.6.20-rc5 #58 SMP Thu Jan 18 13:35:01 CET 2007 ppc64 ppc64
ppc64 GNU/Linux
[root@ps3 tmp]# cat /proc/cpuinfo
processor       : 0
cpu             : Cell Broadband Engine, altivec supported
clock           : 3192.000000MHz
revision        : 5.1 (pvr 0070 0501)

processor       : 1
cpu             : Cell Broadband Engine, altivec supported
clock           : 3192.000000MHz
revision        : 5.1 (pvr 0070 0501)

timebase        : 79800000
platform        : PS3
machine         : PS3
```

Most of the files in /proc are read only, some require root privileges. But some files are writable, a lot of files in **/proc/sys** are writable. Let's discuss some of the files in /proc.

3.2.11.1. /proc/cmdline

The parameters that were passed to the kernel at boot time are in **/proc/cmdline**.

```
paul@RHELv4u4:~$ cat /proc/cmdline
ro root=/dev/VolGroup00/LogVol00 rhgb quiet
```

3.2.11.2. /proc/filesystems

The **/proc/filesystems** file displays a list of supported file systems. When you mount a file system without explicitly defining one, then mount will first try to probe **/etc/filesystems** and then probe **/proc/filesystems** for all the filesystems in there without the **nodev** label. If /etc/filesystems ends with a line containing nothing but a *, then both files are probed.

```
paul@RHELv4u4:~$ cat /proc/filesystems
nodev   sysfs
nodev   rootfs
nodev   bdev
nodev   proc
nodev   sockfs
nodev   binfmt_misc
nodev   usbfs
nodev   usbdevfs
nodev   futexfs
nodev   tmpfs
nodev   pipefs
nodev   eventpollfs
nodev   devpts
        ext2
nodev   ramfs
nodev   hugetlbfs
        iso9660
nodev   relayfs
nodev   mqueue
nodev   selinuxfs
        ext3
nodev   rpc_pipefs
nodev   vmware-hgfs
nodev   autofs
paul@RHELv4u4:~$
```

3.2.11.3. /proc/interrupts

On the x86 architecture, **/proc/interrupts** displays the interrupts.

```
paul@RHELv4u4:~$ cat /proc/interrupts
           CPU0
  0:   13876877    IO-APIC-edge  timer
  1:         15    IO-APIC-edge  i8042
  8:          1    IO-APIC-edge  rtc
  9:          0   IO-APIC-level  acpi
 12:         67    IO-APIC-edge  i8042
 14:        128    IO-APIC-edge  ide0
 15:     124320    IO-APIC-edge  ide1
169:     111993   IO-APIC-level  ioc0
177:       2428   IO-APIC-level  eth0
NMI:          0
LOC:   13878037
```

```
ERR:          0
MIS:          0
paul@RHELv4u4:~$
```

On a machine with two CPU's, the file looks like this.

```
paul@laika:~$ cat /proc/interrupts
           CPU0       CPU1
  0:     860013          0  IO-APIC-edge      timer
  1:       4533          0  IO-APIC-edge      i8042
  7:          0          0  IO-APIC-edge      parport0
  8:    6588227          0  IO-APIC-edge      rtc
 10:       2314          0  IO-APIC-fasteoi  acpi
 12:        133          0  IO-APIC-edge      i8042
 14:          0          0  IO-APIC-edge      libata
 15:      72269          0  IO-APIC-edge      libata
 18:          1          0  IO-APIC-fasteoi  yenta
 19:     115036          0  IO-APIC-fasteoi  eth0
 20:     126871          0  IO-APIC-fasteoi  libata, ohci1394
 21:      30204          0  IO-APIC-fasteoi  ehci_hcd:usb1, uhci_hcd:usb2
 22:       1334          0  IO-APIC-fasteoi  saa7133[0], saa7133[0]
 24:     234739          0  IO-APIC-fasteoi  nvidia
NMI:         72         42
LOC:     860000     859994
ERR:          0
paul@laika:~$
```

3.2.11.4. /proc/kcore

The physical memory is represented in **/proc/kcore**. Do not try to cat this file, instead use a debugger. The size of /proc/kcore is the same as your physical memory, plus four bytes.

```
paul@laika:~$ ls -lh /proc/kcore
-r-------- 1 root root 2.0G 2007-01-30 08:57 /proc/kcore
paul@laika:~$
```

3.2.11.5. /proc/mdstat

You can obtain RAID information from the kernel by displaying **/proc/mdstat**. With a RAID configured, it looks like this.

```
paul@RHELv4u2:~$ cat /proc/mdstat
Personalities : [raid5]
md0 : active raid5 sdd1[2] sdc1[1] sdb1[0]
      2088192 blocks level 5, 64k chunk, algorithm 2 [3/3] [UUU]

unused devices: <none>
paul@RHELv4u2:~$
```

When there is no RAID present, the following is displayed.

```
paul@RHELv4u4:~$ cat /proc/mdstat
Personalities :
unused devices: <none>
paul@RHELv4u4:~$
```

3.2.11.6. /proc/meminfo

You will rarely want to look at **/proc/meminfo**...

```
paul@RHELv4u4:~$ cat /proc/meminfo
MemTotal:       255864 kB
MemFree:          5336 kB
Buffers:         42396 kB
Cached:         159912 kB
SwapCached:          0 kB
Active:         104184 kB
Inactive:       119724 kB
HighTotal:           0 kB
HighFree:            0 kB
LowTotal:       255864 kB
LowFree:          5336 kB
SwapTotal:     1048568 kB
SwapFree:      1048568 kB
Dirty:              40 kB
Writeback:           0 kB
Mapped:          33644 kB
Slab:            21956 kB
CommitLimit:   1176500 kB
Committed_AS:    82984 kB
PageTables:        960 kB
VmallocTotal:   761848 kB
VmallocUsed:      2588 kB
VmallocChunk:   759096 kB
HugePages_Total:     0
HugePages_Free:      0
Hugepagesize:     4096 kB
```

...since the **free** command displays the same information in a more user friendly output.

```
paul@RHELv4u4:~$ free -om
           total       used       free     shared    buffers     cached
Mem:         249        244          5          0         41        156
Swap:       1023          0       1023
paul@RHELv4u4:~$
```

3.2.11.7. /proc/modules

/proc/modules lists all modules loaded by the kernel. The output would be too long to display here, so lets **grep** for a few. First vm (from Vmware), which tells us that vmmon and vmnet are both loaded. You can display the same information with **lsmod**.

```
paul@laika:~$ cat /proc/modules | grep vm
vmnet 36896 13 - Live 0xffffffff88b21000 (P)
vmmon 194540 0 - Live 0xffffffff88af0000 (P)
paul@laika:~$ lsmod | grep vm
vmnet                  36896  13
vmmon                 194540  0
paul@laika:~$
```

Some modules depend on others. In the following example, you can see that the nfsd module is used by exportfs, lockd and sunrpc.

```
paul@laika:~$ cat /proc/modules | grep nfsd
nfsd 267432 17 - Live 0xffffffff88a40000
exportfs 7808 1 nfsd, Live 0xffffffff88a3d000
lockd 73520 3 nfs,nfsd, Live 0xffffffff88a2a000
sunrpc 185032 12 nfs,nfsd,lockd, Live 0xffffffff889fb000
paul@laika:~$ lsmod | grep nfsd
nfsd                  267432  17
exportfs                7808  1 nfsd
lockd                  73520  3 nfs,nfsd
sunrpc                185032  12 nfs,nfsd,lockd
paul@laika:~$
```

3.2.11.8. /proc/mounts

Like the **mount** command and the **/etc/mtab** file, **/proc/mounts** lists all the mounted file systems. But /proc/mounts displays what the kernel sees, so it is always up to date and correct. You see the device, mount point, file system, read-only or read-write and two zero's.

```
paul@RHELv4u4:~$ cat /proc/mounts
rootfs / rootfs rw 0 0
/proc /proc proc rw,nodiratime 0 0
none /dev tmpfs rw 0 0
/dev/root / ext3 rw 0 0
none /dev tmpfs rw 0 0
none /selinux selinuxfs rw 0 0
/proc /proc proc rw,nodiratime 0 0
/proc/bus/usb /proc/bus/usb usbfs rw 0 0
/sys /sys sysfs rw 0 0
none /dev/pts devpts rw 0 0
/dev/sda1 /boot ext3 rw 0 0
none /dev/shm tmpfs rw 0 0
none /proc/sys/fs/binfmt_misc binfmt_misc rw 0 0
sunrpc /var/lib/nfs/rpc_pipefs rpc_pipefs rw 0 0
paul@RHELv4u4:~$
```

3.2.11.9. /proc/partitions

The **/proc/partitions** file contains a table with major and minor number of partitioned devices, their number of blocks and the device name in **/dev**. Verify with **/proc/devices** to link the major number to the proper device.

```
paul@RHELv4u4:~$ cat /proc/partitions
major minor  #blocks  name

   3     0     524288 hda
   3    64     734003 hdb
   8     0    8388608 sda
   8     1     104391 sda1
   8     2    8281507 sda2
   8    16    1048576 sdb
   8    32    1048576 sdc
   8    48    1048576 sdd
 253     0    7176192 dm-0
 253     1    1048576 dm-1
paul@RHELv4u4:~$
```

3.2.11.10. /proc/swaps

You can find information about **swap partition(s)** in **/proc/swaps**.

```
paul@RHELv4u4:~$ cat /proc/swaps
Filename                          Type        Size    Used    Priority
/dev/mapper/VolGroup00-LogVol01   partition   1048568 0       -1
paul@RHELv4u4:~$
```

3.2.12. /root the superuser's home

On many systems, **/root** is the default location for the root user's personal data and profile. If it does not exist by default, then some administrators create it.

3.2.13. /sbin system binaries

Similar to /bin, but mainly for booting and for tools to configure the system. A lot of the system binaries will require root privileges for certain tasks. You will also find a **/sbin** subdirectory in other directories.

3.2.14. /srv served by your system

You may find **/srv** to be empty on many systems, but not for long. The FHS suggests locating cvs, rsync, ftp and www data to this location. The FHS also approves administrative naming in /srv, like /srv/project55/ftp and /srv/sales/www. Red Hat plans to move some data that is currently located in /var to /srv.

3.2.15. /sys Linux 2.6 hot plugging

The **/sys** directory is created for the Linux 2.6 kernel. Since 2.6, Linux uses **sysfs** to support **usb** and **IEEE 1394** (aka **FireWire**) hot plug devices. See the manual pages of udev(8) (the successor of **devfs**) and hotplug(8) for more info (Or visit http://linux-hotplug.sourceforge.net/).

```
paul@RHELv4u4:~$ ls /sys/*
/sys/block:
dm-0 fd0 hdb md0  ram1  ram11 ram13 ram15 ram3 ram5 ram7 ram9
dm-1 hda hdc ram0 ram10 ram12 ram14 ram2  ram4 ram6 ram8 sda

/sys/bus:
i2c  ide  pci  platform  pnp  scsi  serio  usb

/sys/class:
firmware i2c-adapter input misc netlink printer scsi_device tty
graphics i2c-dev     mem   net  pci_bus raw     scsi_host   usb

/sys/devices:
pci0000:00  platform  system

/sys/firmware:
acpi

/sys/module:
ac       dm_mirror    ext3      ip_conntrack    ipt_state  md5
autofs4  dm_mod       floppy    iptable_filter  ipv6       mii
battery  dm_snapshot  i2c_core  ip_tables       jbd        mptbase
button   dm_zero      i2c_dev   ipt_REJECT      lp         mptfc

/sys/power:
state
paul@RHELv4u4:~$
```

3.2.16. /tmp for temporary files

When applications (or Users) need to store temporary data, they should use **/tmp**. /tmp might take up diskspace, then again, it might also not (as in being mounted inside RAM memory). In any case, files in /tmp can be cleared by the operating system. Never use /tmp to store data that you want to archive.

3.2.17. /usr Unix System Resources

Although **/usr** is pronounced like user, never forget that it stands for Unix System Resources. The /usr hierarchy should contain **sharable, read only** data. Some people even choose to mount /usr as read only. This can be done from its own partition, or from a read only NFS share.

3.2.18. /var variable data

Data that is unpredictable in size, such as log files (**/var/log**), print spool directories (**/var/spool**) and various caches (**/var/cache**) should be located in **/var**. But /var is much more than that, it contains Process ID files in **/var/run** and temporary files that survive a reboot in **/var/tmp**. There will be more examples of /var usage further in this book.

3.2.18.1. /var/lib/rpm

Red Hat Enterprise Linux keeps files pertaining to **RPM** in **/var/lib/rpm/**.

3.2.18.2. /var/spool/up2date

The **Red Hat Update Agent** uses files in **/var/spool/up2date**. This location is also used when files are downloaded from the **Red Hat Network**.

3.2.19. Practice: file system tree

1. Does the file /bin/cat exist ? What about /bin/dd and /bin/echo. What is the type of these files ?

2. What is the size of the Linux kernel file(s) (vmlinu*) in /boot ?

3. Create a directory ~/test. Then issue the following commands:

```
cd ~/test

dd if=/dev/zero of=zeroes.txt count=1 bs=100

od zeroes.txt
```

dd will copy one times (count=1) a block of size 100 bytes (bs=100) from the file /dev/zero to ~/test/zeroes.txt. Can you describe the functionality of /dev/zero ?

4. Now issue the following command:

```
dd if=/dev/random of=random.txt count=1 bs=100 ; od random.txt
```

dd will copy one times (count=1) a block of size 100 bytes (bs=100) from the file /dev/random to ~/test/random.txt. Can you describe the functionality of /dev/random ?

5. Issue the following two commands, and look at the first character of each output line.

```
ls -l /dev/sd* /dev/hd*

ls -l /dev/tty* /dev/input/mou*
```

The first ls will show block(b) devices, the second ls shows character(c) devices. Can you tell the difference between block and character devices ?

6. Use cat to display /etc/hosts and /etc/resolv.conf. What is your idea about the purpose of these files ?

7. Are there any files in /etc/skel/ ? Check also for hidden files.

8. Display /proc/cpuinfo. On what architecture is your Linux running ?

9. Display /proc/interrupts. What is the size of this file ? Where is this file stored ?

10. Can you enter the /root directory ? Are there (hidden) files ?

11. Are ifconfig, fdisk, parted, shutdown and grub-install present in /sbin ? Why are these binaries in /sbin and not in /bin ?

12. Is /var/log a file or a directory ? What about /var/spool ?

13. Open two command prompts (Ctrl-Shift-T in gnome-terminal) or terminals (Ctrl-Alt-F1, Ctrl-Alt-F2, ...) and issue the **who am i** in both. Then try to echo a word from one terminal to the other.

14. Read the man page of **random** and explain the difference between **/dev/random** and **/dev/urandom**.

3.2.20. Solutions: file system tree

1. Does the file /bin/cat exist ? What about /bin/dd and /bin/echo. What is the type of these files ?

```
ls /bin/cat ; file /bin/cat

ls /bin/dd ; file /bin/dd

ls /bin/echo ; file /bin/echo
```

2. What is the size of the Linux kernel file(s) (vmlinu*) in /boot ?

```
ls -lh /boot/vm*
```

3. Create a directory ~/test. Then issue the following commands:

```
cd ~/test

dd if=/dev/zero of=zeroes.txt count=1 bs=100

od zeroes.txt
```

dd will copy one times (count=1) a block of size 100 bytes (bs=100) from the file /dev/zero to ~/test/zeroes.txt. Can you describe the functionality of /dev/zero ?

/dev/zero is a Linux special device. It can be considered a source of zeroes. You cannot send something to /dev/zero, but you can read zeroes from it.

4. Now issue the following command:

```
dd if=/dev/random of=random.txt count=1 bs=100 ; od random.txt
```

dd will copy one times (count=1) a block of size 100 bytes (bs=100) from the file /dev/random to ~/test/random.txt. Can you describe the functionality of /dev/random ?

/dev/random acts as a **random number generator** on your Linux machine.

5. Issue the following two commands, and look at the first character of each output line.

```
ls -l /dev/sd* /dev/hd*
```

```
ls -l /dev/tty* /dev/input/mou*
```

The first ls will show block(b) devices, the second ls shows character(c) devices. Can you tell the difference between block and character devices ?

Block devices are always written to (or read from) in blocks. For hard disks, blocks of 512 bytes are common. Character devices act as a stream of characters (or bytes). Mouse and keyboard are typical character devices.

6. Use cat to display /etc/hosts and /etc/resolv.conf. What is your idea about the purpose of these files ?

`/etc/hosts` `contains hostnames with their ip-address`

`/etc/resolv.conf` `should contain the ip-address of a DNS name server.`

7. Are there any files in /etc/skel/ ? Check also for hidden files.

```
Issue "ls -al /etc/skel/". Yes, there should be hidden files there.
```

8. Display /proc/cpuinfo. On what architecture is your Linux running ?

```
The file should contain at least one line with Intel or other cpu.
```

9. Display /proc/interrupts. What is the size of this file ? Where is this file stored ?

The size is zero, yet the file contains data. It is not stored anywhere because /proc is a virtual file system that allows you to talk with the kernel. (If you answered "stored in RAM-memory, that is also correct...).

10. Can you enter the /root directory ? Are there (hidden) files ?

```
Try "cd /root". Yes there are (hidden) files there.
```

11. Are ifconfig, fdisk, parted, shutdown and grub-install present in /sbin ? Why are these binaries in /sbin and not in /bin ?

```
Because those files are only meant for system administrators.
```

12. Is /var/log a file or a directory ? What about /var/spool ?

```
Both are directories.
```

13. Open two command prompts (Ctrl-Shift-T in gnome-terminal) or terminals (Ctrl-Alt-F1, Ctrl-Alt-F2, ...) and issue the **who am i** in both. Then try to echo a word from one terminal to the other.

```
tty-terminal: echo Hello > /dev/tty1
```

```
pts-terminal: echo Hello > /dev/pts/1
```

14. Read the man page of **random** and explain the difference between **/dev/random** and **/dev/urandom**.

```
man 4 random
```

Chapter 4. Introduction to the shell

4.1. about shells

4.1.1. several shells

The command line interface used on most Linux systems is **bash**, which stands for **Bourne again shell**. Bash incorporates features from **sh** (the original Bourne shell), **csh** (the C shell) and **ksh** (the Korn shell). Ubuntu recently started including the **dash** (Debian ash) shell.

This chapter will explain general features of a shell using mainly the **/bin/bash** shell. Important differences in **/bin/ksh** will be mentioned seperately.

4.1.2. type

To find out whether a command given to the shell will be executed as an **external command** or as a **builtin command**, use the **type** command.

```
paul@laika:~$ type cd
cd is a shell builtin
paul@laika:~$ type cat
cat is /bin/cat
```

You can also use this shell builtin to show you whether the command is aliased or not.

```
paul@laika:~$ type ls
ls is aliased to `ls --color=auto'
```

Careful, some commands exist both as a shell builtin and as an external command.

```
paul@laika:~$ type -a echo
echo is a shell builtin
echo is /bin/echo
paul@laika:~$
```

4.1.3. which

The which command will look for binaries in the **PATH** environment variable. (Variables are explained later). In the screenshot below, it looks like cd is built-in, and ls cp rm mv mkdir pwd and which are external commands.

```
[root@RHEL4b ~]# which cp ls mv rm cd mkdir pwd which
/bin/cp
```

```
/bin/ls
/bin/mv
/bin/rm
/usr/bin/which: no cd in (/usr/kerberos/sbin:/usr/kerberos/bin:...
/bin/mkdir
/bin/pwd
/usr/bin/which
[root@RHEL4b ~]#
```

4.1.4. alias

4.1.4.1. create an alias

The shell will allow you to create aliases. Aliases can be used to create an easier to remember alias for an existing command.

```
[paul@RHELv4u3 ~]$ cat count.txt
one
two
three
[paul@RHELv4u3 ~]$ alias dog=tac
[paul@RHELv4u3 ~]$ dog count.txt
three
two
one
```

4.1.4.2. abbreviate commands

An **alias** can also be useful to abbreviate an existing command.

```
paul@laika:~$ alias ll='ls -lh --color=auto'
paul@laika:~$ alias c='clear'
paul@laika:~$
```

4.1.4.3. default options

Aliases can be used to supply commands with default options. The example below shows how to make the -i option default when typing rm.

```
[paul@RHELv4u3 ~]$ rm -i winter.txt
rm: remove regular file `winter.txt'? no
[paul@RHELv4u3 ~]$ rm winter.txt
[paul@RHELv4u3 ~]$ ls winter.txt
ls: winter.txt: No such file or directory
[paul@RHELv4u3 ~]$ touch winter.txt
[paul@RHELv4u3 ~]$ alias rm='rm -i'
[paul@RHELv4u3 ~]$ rm winter.txt
rm: remove regular empty file `winter.txt'? no
[paul@RHELv4u3 ~]$
```

Some distributions enable default aliases to protect users from accidentaly erasing files ('rm -i', 'mv -i', 'cp -i')

4.1.4.4. viewing aliases

You can give multiple aliases as arguments to the **alias** command to get a small list. Not providing any argument will give you a complete list of current aliases.

```
paul@laika:~$ alias c ll
alias c='clear'
alias ll='ls -lh --color=auto'
```

4.1.4.5. unalias

You can undo an alias with the **unalias** command.

```
[paul@RHEL4b ~]$ which rm
/bin/rm
[paul@RHEL4b ~]$ alias rm='rm -i'
[paul@RHEL4b ~]$ which rm
alias rm='rm -i'
        /bin/rm
[paul@RHEL4b ~]$ unalias rm
[paul@RHEL4b ~]$ which rm
/bin/rm
[paul@RHEL4b ~]$
```

4.1.5. echo

This book frequently uses the **echo** command to demonstrate shell features. The echo command echoes the input that it receives.

```
paul@laika:~$ echo Burtonville
Burtonville
paul@laika:~$ echo Smurfs are blue
Smurfs are blue
```

4.1.6. shell expansion

The shell is very important, because every command on your Linux system is processed and changed by the shell. After you type the command, but before the command is executed the shell might change your command line! The manual page of a typical shell contains more than one hundred pages.

One of the primary features of a shell is to perform a **command line scan**. When you enter a command on the shell's command prompt, and press the enter key, then the shell will start scanning that line. While scanning the line, the shell might make a lot of changes to the command line you typed. This process is called **shell expansion**. After the shell has finished scanning and changing that line, the line will be executed. Shell expansion is influenced by the following topics : control operators, white space removal, filename generation, variables, escaping, embedding and shell aliases. All these topics are discussed in the next sections.

4.1.7. internal and external commands

Not all commands are external to the shell, some are built-in.

External commands are programs that have their own binary and reside somewhere on the system in a directory. Many external commands are located in /bin or /sbin. **Builtin commands** are functions inside the shell program. When an external command exists with the same name as a built-in command, then the builtin will take priority. You will need to provide the full path to execute the external command.

4.1.8. displaying shell expansion

You can display the shell expansion with **set -x**, and stop displaying it with **set +x**. You might want to use this further on in this course, or when in doubt about what exactly the shell is doing with your command.

```
[paul@RHELv4u3 ~]$ set -x
++ echo -ne '\033]0;paul@RHELv4u3:~\007'
[paul@RHELv4u3 ~]$ echo $USER
+ echo paul
paul
++ echo -ne '\033]0;paul@RHELv4u3:~\007'
[paul@RHELv4u3 ~]$ echo \$USER
+ echo '$USER'
$USER
++ echo -ne '\033]0;paul@RHELv4u3:~\007'
[paul@RHELv4u3 ~]$ set +x
+ set +x
[paul@RHELv4u3 ~]$ echo $USER
paul
```

4.2. Practice: about shells

1. Is tac a shell builtin command ?

2. Is there an existing alias for rm ?

3. Read the man page of rm, make sure you understand the -i option of rm. Create and rm a file to test the -i option.

4. Execute: **alias rm='rm -i'** . Test your alias with a testfile. Does this work as expected ?

5. List all current aliases.

6. Create an alias called 'city' that echoes your hometown.

7. Use your alias to test that it works.

8. Remove your city alias.

9. What is the location of the cat and the passwd commands ?

10. Explain the difference between the following commands:

```
echo
```

```
/bin/echo
```

The 'echo' will be interpreted by the shell as the builtin echo command. The '/bin/echo' will make the shell execute the echo binary located in the /bin directory.

11. Explain the difference between the following commands:

```
echo Hello
```

```
echo -n Hello
```

The -n option of the echo command will prevent echo from echoing a trailing newline. 'echo Hello' will echo six characters in total, 'echo -n hello' only echoes five characters.

4.3. Solution: about shells

1. Is tac a shell builtin command ?

```
type tac
```

2. Is there an existing alias for rm ?

```
alias rm
```

3. Read the man page of rm, make sure you understand the -i option of rm. Create and rm a file to test the -i option.

```
man rm
```

```
touch testfile
```

```
rm -i testfile
```

4. Execute: **alias rm='rm -i'** . Test your alias with a testfile. Does this work as expected ?

```
touch testfile
```

```
rm testfile (should ask for confirmation!)
```

5. List all current aliases.

```
alias
```

6. Create an alias called 'city' that echoes your hometown.

```
alias city='echo Antwerpen'
```

7. Use your alias to test that it works.

```
city (it should display Antwerpen)
```

8. Remove your city alias.

```
unalias city
```

9. What is the location of the cat and the passwd commands ?

```
which cat (probably /bin/cat)
```

```
which passwd (probably /usr/bin/passwd)
```

10. Explain the difference between the following commands:

```
echo
```

```
/bin/echo
```

The **echo** will be interpreted by the shell as the builtin echo command. The **/bin/echo** will make the shell execute the echo binary located in the /bin directory.

11. Explain the difference between the following commands:

```
echo Hello
```

```
echo -n Hello
```

The -n option of the echo command will prevent echo from echoing a trailing newline. **echo Hello** will echo six characters in total, **echo -n hello** only echoes five characters.

4.4. control operators

4.4.1. ; semicolon

You can put two or more commands on the same line, separated by a semicolon **;**. The scan will then go until each semicolon, and the lines will be executed sequentially, with the shell waiting for each command to end before starting the next one.

```
[paul@RHELv4u3 ~]$ echo Hello
Hello
[paul@RHELv4u3 ~]$ echo World
World
[paul@RHELv4u3 ~]$ echo Hello ; echo World
Hello
World
```

```
[paul@RHELv4u3 ~]$
```

4.4.2. & ampersand

When on the other hand you end a line with an ampersand **&**, then the shell will not wait for the command to finish. You will get your shell prompt back, and the command is executed in background. You will get a message when it has finished executing in background.

```
[paul@RHELv4u3 ~]$ sleep 20 &
[1] 7925
[paul@RHELv4u3 ~]$
...wait 20 seconds...
[paul@RHELv4u3 ~]$
[1]+  Done                    sleep 20
```

The technical explanation of what happens in this case is explained in the chapter about **processes**.

4.4.3. && double ampersand

You can control execution of commands with **&&** denoting a logical AND. With && the second command is only executed when the first one succeeds (returns a zero exit status).

```
paul@barry:~$ echo first && echo second ; echo third
first
second
third
paul@barry:~$ zecho first && echo second ; echo third
-bash: zecho: command not found
third
paul@barry:~$
```

Another example of the same **logical AND** principle.

```
[paul@RHELv4u3 ~]$ cd gen && ls
file1  file3  File55  fileab  FileAB   fileabc
file2  File4  FileA   Fileab  fileab2
[paul@RHELv4u3 gen]$ cd gen && ls
-bash: cd: gen: No such file or directory
[paul@RHELv4u3 gen]$
```

4.4.4. || double vertical bar

The reverse is true for || . Meaning the second command is only executed when the first command fails (or in other words: returns a non-zero exit status).

```
paul@barry:~$ echo first || echo second ; echo third
first
third
paul@barry:~$ zecho first || echo second ; echo third
-bash: zecho: command not found
second
third
paul@barry:~$
```

Another example of the same **logical OR** principle.

```
[paul@RHELv4u3 ~]$ cd gen || ls
[paul@RHELv4u3 gen]$ cd gen || ls
-bash: cd: gen: No such file or directory
file1  file3  File55  fileab  FileAB   fileabc
file2  File4  FileA   Fileab  fileab2
[paul@RHELv4u3 gen]$
```

4.4.5. Combining && and ||

You can use the logical AND and OR to echo whether a command worked or not.

```
paul@laika:~/test$ rm file1 && echo It worked! || echo It failed!
It worked!
paul@laika:~/test$ rm file1 && echo It worked! || echo It failed!
rm: cannot remove `file1': No such file or directory
It failed!
paul@laika:~/test$
```

4.4.6. # pound sign

Anything written after a pound sign (#) is ignored by the shell. This is useful to write **shell comment**, but has no influence on the command execution or shell expansion.

```
paul@barry:~$ mkdir test    # we create a directory
paul@barry:~$ cd test       #### we enter the directory
paul@barry:~/test$ ls       # is it empty ?
paul@barry:~/test$
```

4.4.7. \ escaping special characters

When you want to use one of the shell control characters, but without the shell interpreting them, then you can **escape** them with a backslash \.

```
[paul@RHELv4u3 ~]$ echo hello \; world
hello ; world
[paul@RHELv4u3 ~]$ echo hello\ \ \ world
hello   world
```

```
[paul@RHELv4u3 ~]$ echo escaping \\\ \#\ \&\ \"\ \'
escaping \ # & " '
[paul@RHELv4u3 ~]$ echo escaping \\\?\*\"\'
escaping \?*"'
```

4.4.8. end of line backslash

Lines ending in a backslash are continued on the next line. The shell is not interpreting the newline character and will wait with shell expansion and execution of the command line until a newline without backslash is encountered.

```
[paul@RHEL4b ~]$ echo This command line \
> is split in three \
> parts
This command line is split in three parts
[paul@RHEL4b ~]$
```

4.5. Practice: control operators

0. All these questions can be answered by one command line!!

1. When you type 'passwd', which file is executed ?

2. What kind of file is that ?

3. Execute the pwd command twice. (remember 0.)

4. Execute ls after cd /etc, but only if cd /etc did not error.

5. Execute cd /etc after cd etc, but only if cd etc fails.

6. Execute sleep 10, what is this command doing ?

7. Execute sleep 200 in background (do not wait for it to finish).

8. Use echo to display "Hello World with strange' characters \ * [} ~ \\ ." (including all quotes)

9. Use one echo command to display three words on three lines.

4.6. Solution: control operators

0. Each question can be answered by one command line!

1. When you type 'passwd', which file is executed ?

```
which passwd
```

2. What kind of file is that ?

```
file $(which passwd)
```

3. Execute the pwd command twice. (remember 0.)

```
pwd ; pwd
```

4. Execute ls after cd /etc, but only if cd /etc did not error.

```
cd /etc && ls
```

5. Execute cd /etc after cd etc, but only if cd etc fails.

```
cd etc || cd /etc
```

6. Execute sleep 10, what is this command doing ?

```
pausing for ten seconds
```

7. Execute sleep 200 in background (do not wait for it to finish).

```
sleep 200 &
```

8. Use echo to display "Hello World with strange' characters \ * [} ~ \\ ." (including all quotes)

```
echo \"Hello World with strange\' characters \\ \* \[ \} \~ \\\\ \. \"
```

```
echo \""Hello World with strange' characters \ * [ } ~ \\ . "\"
```

9. Use one echo command to display three words on three lines.

```
echo -e "one \ntwo \nthree"
```

4.7. shell variables

4.7.1. $ dollar sign

Another important character interpreted by the shell is the dollar sign **$**. The shell will look for an **environment variable** named like the string behind the dollar sign and replace it with the value of the variable (or with nothing if the variable does not exist).

An example of the $USER variable. The example shows that shell variables are case sensitive!

```
[paul@RHELv4u3 ~]$ echo Hello $USER
Hello paul
[paul@RHELv4u3 ~]$ echo Hello $user
Hello
```

4.7.2. common variables

Some more examples using $HOSTNAME, $USER, $UID, $SHELL and $HOME.

```
[paul@RHELv4u3 ~]$ echo This is the $SHELL shell
This is the /bin/bash shell
[paul@RHELv4u3 ~]$ echo This is $SHELL on computer $HOSTNAME
This is /bin/bash on computer RHELv4u3.localdomain
[paul@RHELv4u3 ~]$ echo The userid of $USER is $UID
The userid of paul is 500
[paul@RHELv4u3 ~]$ echo My homedir is $HOME
My homedir is /home/paul
```

4.7.3. $? dollar question mark

The exit code of the previous command is stored in the shell variable $?. Actually **$?** is a shell parameter and not a variable, you cannot assign a value to $?.

```
paul@laika:~/test$ touch file1 ; echo $?
0
paul@laika:~/test$ rm file1 ; echo $?
0
paul@laika:~/test$ rm file1 ; echo $?
rm: cannot remove `file1': No such file or directory
1
```

4.7.4. unbound variables

The example below tries to display the value of the $MyVar variable, but it fails because the variable does not exist. By default the shell will display nothing when a variable is unbound (does not exist).

```
[paul@RHELv4u3 gen]$ echo $MyVar

[paul@RHELv4u3 gen]$
```

There is however the **nounset** shell attribute that you can use to generate an error when a variable does not exist.

```
paul@laika:~$ set -u
paul@laika:~$ echo $Myvar
bash: Myvar: unbound variable
paul@laika:~$ set +u
paul@laika:~$ echo $Myvar

paul@laika:~$
```

In the bash shell **set -u** is identical to **set -o nounset** and likewise **set +u** is identical to **set +o nounset**.

4.7.5. creating and setting variables

The example creates the variable $MyVar and sets its value.

```
[paul@RHELv4u3 gen]$ MyVar=555
[paul@RHELv4u3 gen]$ echo $MyVar
555
[paul@RHELv4u3 gen]$
```

4.7.6. set

You can use the **set** command to display a list of environment variables. On Ubuntu and Debian systems, the set command will end the list of shell variables with a list of shell functions, use **set | more** to see the variables then.

4.7.7. unset

Use the **unset** command to remove a variable from your shell environment.

```
[paul@RHEL4b ~]$ MyVar=8472
[paul@RHEL4b ~]$ echo $MyVar;unset MyVar;echo $MyVar
8472

[paul@RHEL4b ~]$
```

4.7.8. env

The **env** command can also be useful for other neat things, like starting a clean shell (a shell without any inherited environment). The **env -i** command clears the environment for the subshell. Notice that bash will set the $SHELL variable on startup.

```
[paul@RHEL4b ~]$ bash -c 'echo $SHELL $HOME $USER'
/bin/bash /home/paul paul
[paul@RHEL4b ~]$ env -i bash -c 'echo $SHELL $HOME $USER'
/bin/bash
[paul@RHEL4b ~]$
```

You can also use the env tool to set the LANG variable (or any other) for an instance of bash with one command. The example below uses this to show the influence of the LANG variable on file globbing.

```
[paul@RHEL4b test]$ env LANG=C bash -c 'ls File[a-z]'
Filea  Fileb
[paul@RHEL4b test]$ env LANG=en_US.UTF-8 bash -c 'ls File[a-z]'
Filea  FileA  Fileb  FileB
[paul@RHEL4b test]$
```

4.7.9. exporting variables

You can export shell variables to other shells with the **export** command. This will export the variable to child shells.

```
[paul@RHEL4b ~]$ var3=three
[paul@RHEL4b ~]$ var4=four
[paul@RHEL4b ~]$ export var4
[paul@RHEL4b ~]$ echo $var3 $var4
three four
[paul@RHEL4b ~]$ bash
[paul@RHEL4b ~]$ echo $var3 $var4
four
```

But it will not export to the parent shell (previous screenshot continued).

```
[paul@RHEL4b ~]$ export var5=five
[paul@RHEL4b ~]$ echo $var3 $var4 $var5
four five
[paul@RHEL4b ~]$ exit
exit
[paul@RHEL4b ~]$ echo $var3 $var4 $var5
three four
[paul@RHEL4b ~]$
```

4.7.10. delineate variables

Until now, we have seen that bash interpretes a variable starting from a dollar sign, until the first occurence of a non-alphanumerical character that is not an underscore. In some situations, this can be a problem. This issue can be resolved with curly braces like in this example.

```
[paul@RHEL4b ~]$ prefix=Super
[paul@RHEL4b ~]$ echo Hello $prefixman and $prefixgirl
Hello  and
[paul@RHEL4b ~]$ echo Hello ${prefix}man and ${prefix}girl
Hello Superman and Supergirl
[paul@RHEL4b ~]$
```

4.7.11. quotes and variables

Notice that double quotes still allow the parsing of variables, whereas single quotes prevent this.

```
[paul@RHELv4u3 ~]$ MyVar=555
[paul@RHELv4u3 ~]$ echo $MyVar
555
```

```
[paul@RHELv4u3 ~]$ echo "$MyVar"
555
[paul@RHELv4u3 ~]$ echo '$MyVar'
$MyVar
```

The bash shell will replace variables with their value in double quoted lines, but not in single quoted lines.

```
paul@laika:~$ city=Burtonville
paul@laika:~$ echo "We are in $city today."
We are in Burtonville today.
paul@laika:~$ echo 'We are in $city today.'
We are in $city today.
```

4.8. Practice: shell variables

1. Use echo to display Hello followed by your username. (use a bash variable!)

2. Copy the value of $LANG to $MyLANG.

3. List all current shell variables.

4. Create a variable MyVar with a value of 1201.

5. Do the env and set commands display your variable ?

6. Destroy your variable.

7. Find the list of shell options in the man page of bash. What is the difference between "set -u" and "set -o nounset" ?

8. Create two variables, and export one of them.

9. Display the exported variable in an interactive child shell.

10. Create a variable, give it the value 'Dumb', create another variable with value 'do'. Use echo and the two variables to echo Dumbledore.

11. Activate **nounset** in your shell. Test that it shows an error message when using non-existing variables.

12. Deactivate nounset.

4.9. Solution: shell variables

1. Use echo to display Hello followed by your username. (use a bash variable!)

```
echo Hello $USER
```

2. Copy the value of $LANG to $MyLANG.

```
MyLANG=$LANG
```

3. List all current shell variables.

```
set
```

4. Create a variable MyVar with a value of 1201.

```
MyVar=1201
```

5. Do the env and set commands display your variable ?

```
env | grep 1201 ; echo -- ; set | grep 1201
```

You will notice that **set** displays all variables, whereas **env** does not.

6. Destroy your variable.

```
unset MyVar
```

7. Find the list of shell options in the man page of bash. What is the difference between "set -u" and "set -o nounset" ?

read the manual of bash (man bash), search for nounset -- both mean the same thing.

8. Create two variables, and export one of them.

```
var1=1; export var2=2
```

9. Display the exported variable in an interactive child shell.

```
bash
```

```
echo $var2
```

10. Create a variable, give it the value 'Dumb', create another variable with value 'do'. Use echo and the two variables to echo Dumbledore.

```
varx=Dumb; vary=do
```

```
echo ${varx}le${vary}re
```

11. Activate **nounset** in your shell. Test that it shows an error message when using non-existing variables.

```
set -u
```

```
set -o nounset
```

Both these lines have the same effect (read the manual of bash, search for nounset).

12. Deactivate nounset.

```
set +u
```

```
set +o nounset
```

4.10. white space and quoting

4.10.1. white space removal

Before execution, the shell looks at the command line. Parts that are seperated by one or more consecutive **white spaces** (or tabs) are considered separate arguments. The first argument is the command to be executed, the other arguments are given to the command as arguments.

This explains why the following four different command lines are the same after **shell expansion**.

```
[paul@RHELv4u3 ~]$ echo Hello World
Hello World
[paul@RHELv4u3 ~]$ echo Hello   World
Hello World
[paul@RHELv4u3 ~]$ echo   Hello   World
Hello World
[paul@RHELv4u3 ~]$    echo      Hello      World
Hello World
[paul@RHELv4u3 ~]$
```

The echo command will separate the different arguments it recieves from the shell by a white space.

4.10.2. single quotes

You can prevent the removal of white spaces by quoting the spaces.

```
[paul@RHEL4b ~]$ echo 'A line with      single    quotes'
A line with      single    quotes
[paul@RHEL4b ~]$
```

4.10.3. double quotes

You can also prevent the removal of white spaces by double quoting the spaces.

```
[paul@RHEL4b ~]$ echo "A line with      double    quotes"
A line with      double    quotes
[paul@RHEL4b ~]$
```

The only difference between single and double quotes is the parsing of shell variables. You can already see what happens in this screenshot.

```
paul@laika:~$ echo 'My user is $USER'
My user is $USER
```

```
paul@laika:~$ echo "My user is $USER"
My user is paul
```

4.10.4. echo and quotes

Quoted lines can include special escaped characters recognized by the **echo** command (when using **echo -e**). The screenshot below shows how to use escaped n for a newline and escaped t for a tab (usually eight white spaces).

```
[paul@RHEL4b ~]$ echo -e "A line with \na newline"
A line with
a newline
[paul@RHEL4b ~]$ echo -e 'A line with \na newline'
A line with
a newline
[paul@RHEL4b ~]$ echo -e "A line with \ta tab"
A line with     a tab
[paul@RHEL4b ~]$ echo -e 'A line with \ta tab'
A line with     a tab
[paul@RHEL4b ~]$
```

The echo command can generate more than white spaces, tabs and newlines. Look in the man page for a list of options (and don't forget that echo can be both builtin and external).

4.10.5. shell embedding

Shells can be embedded on the command line, or in other words the command line scan can spawn new processes, containing a fork of the current shell. You can use variables to prove that new shells are created. In the screenshot below, the variable $var1 only exists in the (temporary) sub shell.

```
[paul@RHELv4u3 gen]$ echo $var1

[paul@RHELv4u3 gen]$ echo $(var1=5;echo $var1)
5
[paul@RHELv4u3 gen]$ echo $var1

[paul@RHELv4u3 gen]$
```

You can embed a shell in an **embedded shell**, this is called **nested embedding** of shells.

```
$ P=Parent;
$ echo $P$C$G - $(C=Child;echo $P$C$G - ;echo $(G=Grand;echo $P$C$G))
Parent - ParentChild - ParentChildGrand
```

Single embedding can be useful to avoid changing your current directory. The screenshot below uses back ticks instead of dollar-bracket to embed.

```
[paul@RHELv4u3 ~]$ echo `cd /etc; ls -d * | grep pass`
passwd passwd- passwd.OLD
[paul@RHELv4u3 ~]$
```

4.10.6. backticks and single quotes

Placing the embedding between **backticks** uses one character less than the dollar and parenthesis combo. Be careful however, backticks are often confused with single quotes. The technical difference between ' and ` is significant! You can not use backticks to nest embedded shells.

```
[paul@RHELv4u3 gen]$ echo `var1=5;echo $var1`
5
[paul@RHELv4u3 gen]$ echo 'var1=5;echo $var1'
var1=5;echo $var1
[paul@RHELv4u3 gen]$
```

4.11. Practice: white space and quoting

1. Display **A B C** with two spaces between B and C.

2. Complete the following command (do not use spaces) to display exactly the following output:

```
echo -e "4+4=8" ; echo -e "10+14=24"

4+4     =8

10+14   =24
```

3. Use echo to display the following exactly:

```
""\\`;     "_+
```

4. Use one echo command to display three words on three lines.

5. Execute **cd /var** and **ls** in an embedded shell.

6. Create the variable embvar in an embedded shell and echo it. Does the variable exist in your current shell now ?

7. Explain what "set -x" does. Can this be useful ?

8. Given the following screenshot, add exactly four characters to that command line so that the total output is FirstMiddleLast.

```
[paul@RHEL4b ~]$ echo  First; echo  Middle; echo  Last
First
Middle
Last
[paul@RHEL4b ~]$
```

4.12. Solution: white space and quoting

1. Display **A B C** with two spaces between B and C.

```
echo "A B  C"
```

2. Complete the following command (do not use spaces) to display exactly the following output:

```
echo -e "4+4=8" ; echo -e "10+14=24"
```

```
4+4     =8
```

```
10+14   =24
```

The solution is to use tabs with \t.

```
echo -e "4+4\t=8" ; echo -e "10+14\t=24"
```

3. Use echo to display the following exactly:

```
""\\`;     "_+
```

```
echo \"\"\\\\\`\;\ \ \ \ \"\_\+
```

4. Use one echo command to display three words on three lines.

```
echo -e "one \ntwo \nthree"
```

5. Execute **cd /var** and **ls** in an embedded shell.

```
$(cd /var ; ls)
```

6. Create the variable embvar in an embedded shell and echo it. Does the variable exist in your current shell now ?

```
$(embvar=emb;echo $embvar) ; echo $embvar (the last echo fails).
```

```
$embvar does not exist in your current shell
```

7. Explain what "set -x" does. Can this be useful ?

```
I can be useful to display shell expansion for troubleshooting your command.
```

8. Given the following screenshot, add exactly four characters to that command line so that the total output is FirstMiddleLast.

```
[paul@RHEL4b ~]$ echo  First; echo  Middle; echo  Last
First
Middle
Last
[paul@RHEL4b ~]$
```

```
echo -n First; echo -n Middle; echo Last
```

4.13. file globbing

4.13.1. * asterisk

The shell is also responsible for **file globbing** (or dynamic filename generation). The asterisk ***** is interpreted by the shell as a sign to generate filenames, matching the asterisk to any combination of characters (even none). When no path is given, the shell will use filenames in the current directory. See the man page of **glob(7)** for more information. (This is part of LPI topic 1.103.3.)

```
[paul@RHELv4u3 gen]$ ls
file1  file2  file3  File4  File55  FileA  fileab  Fileab  FileAB  fileabc
[paul@RHELv4u3 gen]$ ls File*
File4  File55  FileA  Fileab  FileAB
[paul@RHELv4u3 gen]$ ls file*
file1  file2  file3  fileab  fileabc
[paul@RHELv4u3 gen]$ ls *ile55
File55
[paul@RHELv4u3 gen]$ ls F*ile55
File55
[paul@RHELv4u3 gen]$ ls F*55
File55
[paul@RHELv4u3 gen]$
```

4.13.2. ? question mark

Similar to the asterisk, the question mark **?** is interpreted by the shell as a sign to generate filenames, matching the question mark with exactly one character.

```
[paul@RHELv4u3 gen]$ ls
file1  file2  file3  File4  File55  FileA  fileab  Fileab  FileAB  fileabc
[paul@RHELv4u3 gen]$ ls File?
File4  FileA
[paul@RHELv4u3 gen]$ ls Fil?4
File4
[paul@RHELv4u3 gen]$ ls Fil??
File4  FileA
[paul@RHELv4u3 gen]$ ls File??
File55  Fileab  FileAB
[paul@RHELv4u3 gen]$
```

4.13.3. [] square brackets

The square bracket **[** is interpreted by the shell as a sign to generate filenames, matching any of the characters between **[** and the first subsequent **]**. The order in this list between the brackets is not important. Each pair of brackets is replaced by exactly one character.

```
[paul@RHELv4u3 gen]$ ls
file1  file2  file3  File4  File55  FileA  fileab  Fileab  FileAB  fileabc
[paul@RHELv4u3 gen]$ ls File[5A]
FileA
[paul@RHELv4u3 gen]$ ls File[A5]
FileA
[paul@RHELv4u3 gen]$ ls File[A5][5b]
File55
[paul@RHELv4u3 gen]$ ls File[a5][5b]
File55  Fileab
[paul@RHELv4u3 gen]$ ls File[a5][5b][abcdefghijklm]
ls: File[a5][5b][abcdefghijklm]: No such file or directory
[paul@RHELv4u3 gen]$ ls file[a5][5b][abcdefghijklm]
fileabc
[paul@RHELv4u3 gen]$
```

You can also exclude characters from a list between square brackets with the exclamation mark **!**. And you are allowed to make combinations of these **wild cards**.

```
[paul@RHELv4u3 gen]$ ls
file1  file2  file3  File4  File55  FileA  fileab  Fileab  FileAB  fileabc
[paul@RHELv4u3 gen]$ ls file[a5][!Z]
fileab
[paul@RHELv4u3 gen]$ ls file[!5]*
file1  file2  file3  fileab  fileabc
[paul@RHELv4u3 gen]$ ls file[!5]?
fileab
[paul@RHELv4u3 gen]$
```

4.13.4. a-z and 0-9 ranges

The bash shell will also understand ranges of characters between brackets.

```
[paul@RHELv4u3 gen]$ ls
file1  file3  File55  fileab  FileAB   fileabc
file2  File4  FileA   Fileab  fileab2
[paul@RHELv4u3 gen]$ ls file[a-z]*
fileab  fileab2  fileabc
[paul@RHELv4u3 gen]$ ls file[0-9]
file1  file2  file3
[paul@RHELv4u3 gen]$ ls file[a-z][a-z][0-9]*
fileab2
[paul@RHELv4u3 gen]$
```

4.13.5. $LANG and square brackets

But, don't forget the influence of the **LANG** variable. Some languages include lowercase letters in an uppercase range (and vice versa).

```
paul@RHELv4u4:~/test$ ls [A-Z]ile?
file1  file2  file3  File4
paul@RHELv4u4:~/test$ ls [a-z]ile?
```

```
file1  file2  file3  File4
paul@RHELv4u4:~/test$ echo $LANG
en_US.UTF-8
paul@RHELv4u4:~/test$ LANG=C
paul@RHELv4u4:~/test$ echo $LANG
C
paul@RHELv4u4:~/test$ ls [a-z]ile?
file1  file2  file3
paul@RHELv4u4:~/test$ ls [A-Z]ile?
File4
paul@RHELv4u4:~/test$
```

4.14. Bash shell options

Both **set** and **unset** are built-in shell commands. They can be used to set options of the bash shell itself. The next example will clarify this. By default, the shell will treat unset variables as a variable having no value. By setting the -u option, the shell will treat any reference to unset variables as an error. See the man page of bash for more information.

```
[paul@RHEL4b ~]$ echo $var123

[paul@RHEL4b ~]$ set -u
[paul@RHEL4b ~]$ echo $var123
-bash: var123: unbound variable
[paul@RHEL4b ~]$ set +u
[paul@RHEL4b ~]$ echo $var123

[paul@RHEL4b ~]$
```

To list all the set options for your Bash shell, use **echo $-**. The noclobber option will be explained later in this book (in the I/O redirection chapter).

```
[paul@RHEL4b ~]$ echo $-
himBH
[paul@RHEL4b ~]$ set -C ; set -u
[paul@RHEL4b ~]$ echo $-
himuBCH
[paul@RHEL4b ~]$ set +C ; set +u
[paul@RHEL4b ~]$ echo $-
himBH
[paul@RHEL4b ~]$
```

4.15. shell history

4.15.1. history variables

The **bash** shell will remember the commands you type, so you can easily repeat previous commands. Some variables are defining this process: **$HISTFILE** points to the location of the history file, **$HISTSIZE** will tell you how many commands will be remembered in your current shell session, **$HISTFILESIZE** is the truncate limit for the number of commands in the history file. Your shell session history is written to the file when exiting the shell. This screenshot lists some history variables in bash.

```
[paul@RHELv4u3 ~]$ echo $HISTFILE
/home/paul/.bash_history
[paul@RHELv4u3 ~]$ echo $HISTFILESIZE
1000
[paul@RHELv4u3 ~]$ echo $HISTSIZE
1000
[paul@RHELv4u3 ~]$
```

The $HISTFILE (defaults to .sh_history in the home directory) and $HISTSIZE variables are also used by the Korn shell (ksh).

```
$ set | grep -i hist
HISTSIZE=5000
```

4.15.2. repeating commands in bash

To repeat the last command in bash, type **!!**. This is pronounced as **bang bang**. To repeat older commands, use **history** to display your history and type **!** followed by a number. The shell will echo the command and execute it.

```
[paul@RHELv4u3 ~]$ history
2  cat /etc/redhat-release
3  uname -r
4  rpm -qa | grep ^parted
...
[paul@RHELv4u3 ~]$ !3
uname -r
2.6.9-34.EL
[paul@RHELv4u3 ~]$
```

You can also use the bash with one or more characters, bash will then repeat the last command that started with those characters. But this can be very very dangerous, you have to be sure about the last command in your current shell history that starts with those characters!

```
[paul@RHEL4b ~]$ ls file4
file4
[paul@RHEL4b ~]$ !ls
ls file4
file4
```

You can also use a colon followed by a regular expression to manipulate the previous command.

```
[paul@RHEL4b ~]$ !ls:s/4/5
```

```
ls file5
file5
```

The history command can also receive the count of history lines required.

```
[paul@RHEL4b ~]$ history 4
  422  ls file4
  423  ls file4
  424  ls file5
  425  history 4
[paul@RHEL4b ~]$
```

4.15.3. repeating commands in ksh

Repeating a command in the Korn shell is very similar. The Korn shell also knows the **history** command, but uses the letter **r** to recall lines from history.

This screenshot shows the history command. Note the different meaning of the parameter.

```
$ history 17
17      clear
18      echo hoi
19      history 12
20      echo world
21      history 17
```

Repeating with r can be combined with the line numbers given by the history command, or with the first few letters of the command.

```
$ r e
echo world
world
$ cd /etc
$ r
cd /etc
$
```

4.16. Practice: discover the shell

1. Create a testdir and enter it.

2. Create files file1 file10 file11 file2 File2 File3 file33 fileAB filea fileA fileAAA file(file 2 (the last one has 6 characters including a space)

3. List (with ls) all files starting with file

4. List (with ls) all files starting with File

5. List (with ls) all files starting with file and ending in a number.

6. List (with ls) all files starting with file and ending with a letter

7. List (with ls) all files starting with File and having a digit as fifth character.

8. List (with ls) all files starting with File and having a digit as fifth character and nothing else.

9. List (with ls) all files starting with a letter and ending in a number.

10. List (with ls) all files that have exactly five characters.

11. List (with ls) all files that start with f or F and end with 3 or A.

12. List (with ls) all files that start with f have i or R as second character and end in a number.

13. List all files that do not start with the letter F.

14. Copy the value of $LANG to $MyLANG.

15. Show the influence of $LANG in listing A-Z or a-z ranges.

16. Write a command line that executes 'rm file55'. Your command line should print 'success' if file55 is removed, and print 'failed' if there was a problem.

17. You receive information that one of your servers was cracked, the cracker probably replaced the ls command. You know that the echo command is safe to use. Can echo replace ls ? How can you list the files in the current directory with echo ?

18. The cd command is also compromised, can echo be used to list files in other directories ? Explain how this works (list the contents of /etc and /bin without ls).

19. Is there another command besides cd to change directories ?

20. Make sure bash remembers the last 5000 commands you typed.

21. Open more than one console (press Ctrl-shift-t in gnome terminal) with the same user account. When is command history written to the history file ?

22. Issue the date command. Now display the date in YYYY/MM/DD format.

23. Issue the cal command. Display a calendar of 1582 and 1752. Notice anything special ?

4.17. Solution: discover the shell

1. Create a testdir and enter it.

```
mkdir testdir; cd testdir
```

2. Create files file1 file10 file11 file2 File2 File3 file33 fileAB filea fileA fileAAA file(file 2 (the last one has 6 characters including a space)

```
touch file1 file10 file11 file2 File2 File3 file33 fileAB filea fileA fileAAA
```

```
touch "file("
```

```
touch "file 2"
```

3. List (with ls) all files starting with file

```
ls file*
```

4. List (with ls) all files starting with File

```
ls File*
```

5. List (with ls) all files starting with file and ending in a number.

```
ls file*[0-9]
```

6. List (with ls) all files starting with file and ending with a letter

```
ls file*[a-z]
```

7. List (with ls) all files starting with File and having a digit as fifth character.

```
ls File[0-9]*
```

8. List (with ls) all files starting with File and having a digit as fifth character and nothing else.

```
ls File[0-9]
```

9. List (with ls) all files starting with a letter and ending in a number.

```
ls [a-z]*[0-9]
```

10. List (with ls) all files that have exactly five characters.

```
ls ?????
```

11. List (with ls) all files that start with f or F and end with 3 or A.

```
ls [fF]*[3A]
```

12. List (with ls) all files that start with f have i or R as second character and end in a number.

```
ls f[iR]*[0-9]
```

13. List all files that do not start with the letter F.

```
ls [!F]*
```

14. Copy the value of $LANG to $MyLANG.

```
MyLANG=$LANG
```

15. Show the influence of $LANG in listing A-Z or a-z ranges.

```
see example in book
```

16. Write a command line that executes 'rm file55'. Your command line should print 'success' if file55 is removed, and print 'failed' if there was a problem.

```
rm file55 && echo success || echo failed
```

17. You receive information that one of your servers was cracked, the cracker probably replaced the ls command. You know that the echo command is safe to use. Can echo replace ls ? How can you list the files in the current directory with echo ?

```
echo *
```

18. The cd command is also compromised, can echo be used to list files in other directories ? Explain how this works (list the contents of /etc and /bin without ls).

```
echo /etc/*.conf # the shell expands the directory for you
```

19. Is there another command besides cd to change directories ?

```
pushd popd
```

20. Make sure bash remembers the last 5000 commands you typed.

```
HISTSIZE=5000
```

21. Open more than one console (press Ctrl-shift-t in gnome terminal) with the same user account. When is command history written to the history file ?

```
when you type exit
```

22. Issue the date command. Now display the date in YYYY/MM/DD format.

```
date +%Y/%m/%d
```

23. Issue the cal command. Display a calendar of 1582 and 1752. Notice anything special ?

```
cal 1582
```

The calendars are different depending on the country. Check http://www.linux-training.be/studentfiles/dates.txt

Chapter 5. Introduction to vi

5.1. About vi

The editor **vi** is installed on almost every Unix system in the world. Linux will very often install **vim** (**Vi IMproved**) which is very similar, but improved. Every Linux system administrator should know vi (or rather vim), because it is often an easy tool to solve problems.

Many Unix and Linux distributions will also have **emacs**, **nano**, **pico**, **joe** or other editors installed. The choice of favorite editor is often a cause for **flame wars** or polls. Feel free to use any of the alternatives to vi(m).

The vi editor is not intuitive to novices, but once you get to know it, vi becomes a very powerful application. Some basic commands are a A i I o O r x G 'n G' b w dw dd d0 d$ yw yy y0 y$ 3dd p P u U :w :q :w! :q! :wq ZZ :r :!cmd ':r !cmd' ddp yyp /pattern. Most Linux distributions will include the **vimtutor** which is a 45 minute lesson in vi.

5.2. Introduction to using vi

5.2.1. command mode and insert mode

The vi editor starts in **command mode**. In command mode, you can type commands. The commands a A i I o O will bring you into **insert mode**. In insert mode, you can type text. The escape key will bring you back to command mode. When in insert mode, vi will display **-- INSERT --** in the bottom left corner.

5.2.2. Start typing (a A i I o O)

The difference between a A i I o and O is the location where you can start typing. a will append after the current character and A will append at the end of the line. i will insert before the current character and I will insert at the beginning of the line. o will put you in a new line after the current line and O will put you in a new line before the current line.

5.2.3. Replace and delete a character (r x)

When in command mode (it doensn't hurt to hit the escape key more than once) you can use the x key to delete the current character. Big X key (or shift x) will delete the character left of the cursor. Also when in command mode, you can use the r key to replace one single character. The r key will bring you in insert mode for just one key press, and will return you immediately to command mode.

5.2.4. Undo and repeat (u .)

When in command mode, you can undo your mistakes with u. You can do your mistakes twice with . (in other words the . will repeat your last command).

5.2.5. Cut, copy and paste a line (dd yy p P)

When in command mode, dd will cut the current line. yy will copy the current line. You can paste the last copied or cut line after (p) or before (P) the current line.

5.2.6. Cut, copy and paste lines (3dd 2yy)

When in command mode, before typing dd or yy, you can type a number to repeat the command a number of times. Thus, 5dd will cut 5 lines and 4yy will copy (yank) 4 lines. That last one will be noted by vi in the bottom left corner as "4 line yanked".

5.2.7. Start and end of a line (0 or ^ and $)

When in command mode, the 0 and the caret ^ will bring you to the start of the current line, whereas the $ will put the cursor at the end of the current line. You can add 0 and $ to the d command, d0 will delete every character between the current character and the start of the line. Likewise d$ will delete everything from the current character till the end of the line. Similarly y0 and y$ will yank till start and end of the current line.

5.2.8. Join two lines (J)

When in command mode, pressing J will append the next line to the current line.

5.2.9. Words (w b)

When in command mode, w will jump you to the next word, and b will get you to the previous word. w and b can also be combined with d and y to copy and cut words (dw db yw yb).

5.2.10. Save (or not) and exit (:w :q :q!)

Pressing the colon : will allow you to give instructions to vi. :w will write (save) the file, :q will quit un unchanged file without saving, :q! will quit vi discarding changes. :wq will save and quit and is the same as typing ZZ in command mode.

5.2.11. Searching (/ ?)

When in command mode typing / will allow you to search in vi for strings (can be a regular expression). Typing /foo will do a forward search for the string foo, typing ? bar will do a backward search for bar.

5.2.12. Replace all (:1,$ s/foo/bar/g)

To replace all occurences of the string foo in bar, first switch to ex mode with : . Then tell vi which lines to use, for example 1,$ will do the replace all from the first to the last line. You can write 1,5 to only process the first five lines. The s/foo/bar/g will replace all occurences of foo with bar.

5.2.13. Reading files (:r :r !cmd)

When in command mode, :r foo will read the file named foo, :r !foo will execute the command foo. The result will be put at the current location. Thus :r !ls will put a listing of the current directory in your textfile.

5.2.14. Setting options

Some options that you can set in vim.

```
:set number  ( also try :se nu )
:set nonumber
:syntax on
:syntax off
:set all  (list all options)
:set tabstop=8
:set tx   (CR/LF style endings)
:set notx
```

You can set these options (and much more) in **~/.vimrc**

```
paul@barry:~$ cat ~/.vimrc
set number
paul@barry:~$
```

5.3. Practice

1. Start the vimtutor and do some or all of the exercises.

2. What 3 key combination in command mode will duplicate the current line.

3. What 3 key combination in command mode will switch two lines' place (line five becomes line six and line six becomes line five).

4. What 2 key combination in command mode will switch a character's place with the next one.

5. vi can understand macro's. A macro can be recorded with q followed by the name of the macro. So qa will record the macro named a. Pressing q again will end the

recording. You can recall the macro with @ followed by the name of the macro. Try this example: i 1 'Escape Key' qa yyp 'Ctrl a' q 5@a (Ctrl a will increase the number with one).

6. Copy /etc/passwd to your ~/passwd. Open the last one in vi and press Ctrl v. Use the arrow keys to select a Visual Block, you can copy this with y or delete it with d. Try pasting it.

7. What does dwwP do when you are at the beginning of a word in a sentence ?

5.4. Solutions to the Practice

2. yyp

3. ddp

4. xp

7. dwwP can switch the current word with the next word.

Chapter 6. Introduction to Users

6.1. Users

6.1.1. User management

User management on any Unix can essentialy be done in three ways. You can use the **graphical** tools provided by your distribution. These tools have a look and feel that depends on the distribution. If you are a novice linux user on your home system, then use the graphical tool that is provided by your distribution. This will make sure that you do not run into problems.

Another option is to use **command line tools** like useradd, usermod, gpasswd, passwd and others. Server administrators are likely to use these tools, since they are familiar and very similar accross many different distributions. This chapter will focus on these command line tools.

A third and rather extremist way is to **edit the local configuration files** directly using vi (or vipw/vigr). Do not attempt this as a novice on production systems!

6.1.2. /etc/passwd

The local user database on Linux (and on most Unixes) is **/etc/passwd**.

```
[root@RHEL5 ~]# tail /etc/passwd
inge:x:518:524:art dealer:/home/inge:/bin/ksh
ann:x:519:525:flute player:/home/ann:/bin/bash
frederik:x:520:526:rubius poet:/home/frederik:/bin/bash
steven:x:521:527:roman emperor:/home/steven:/bin/bash
pascale:x:522:528:artist:/home/pascale:/bin/ksh
geert:x:524:530:kernel developer:/home/geert:/bin/bash
wim:x:525:531:master damuti:/home/wim:/bin/bash
sandra:x:526:532:radish stresser:/home/sandra:/bin/bash
annelies:x:527:533:sword fighter:/home/annelies:/bin/bash
laura:x:528:534:art dealer:/home/laura:/bin/ksh
```

As you can see, this file contains seven columns seperated by a colon. The columns contain the username, an x, the user id, the primary group id, a description, the name of the home directory and the login shell.

6.1.3. root

The **root** user also called the **superuser** is the most powerful account on your Linux system. This user can do almost everything, including the creation of other users. The root user always has userid 0 (regardless of the name of the account).

```
[root@RHEL5 ~]# head -1 /etc/passwd
```

```
root:x:0:0:root:/root:/bin/bash
```

6.1.4. useradd

You can add users with the **useradd** command. The example below shows how to add a user named yanina (last parameter), and at the same time forcing the creation of the home directory (-m), setting the name of the home directory (-d) and setting a description (-c).

```
[root@RHEL5 ~]# useradd -m -d /home/yanina -c "yanina wickmayer" yanina
[root@RHEL5 ~]# tail -1 /etc/passwd
yanina:x:529:529:yanina wickmayer:/home/yanina:/bin/bash
```

The user named yanina received userid 529 and **primary group** id 529.

6.1.5. default useradd options

Both Red Hat Enterprise Linux and Debian/Ubuntu have a file called **/etc/default/useradd** that contains some default user options. Besides using cat to display this file, you can also use **useradd -D**.

```
[root@RHEL4 ~]# useradd -D
GROUP=100
HOME=/home
INACTIVE=-1
EXPIRE=
SHELL=/bin/bash
SKEL=/etc/skel
```

6.1.6. userdel

You can delete the user yanina with **userdel**. The -r option of userdel will also remove the home directory.

```
[root@RHEL5 ~]# userdel -r yanina
```

6.1.7. usermod

You can modify the properties of a user with the **usermod** command. This example uses **usermod** to change the description of the user harry.

```
[root@RHEL4 ~]# tail -1 /etc/passwd
harry:x:516:520:harry potter:/home/harry:/bin/bash
[root@RHEL4 ~]# usermod -c 'wizard' harry
[root@RHEL4 ~]# tail -1 /etc/passwd
harry:x:516:520:wizard:/home/harry:/bin/bash
```

6.2. Identify yourself

6.2.1. whoami

The **whoami** command exists to tell you your username.

```
[root@RHEL5 ~]# whoami
root
[root@RHEL5 ~]# su - paul
[paul@RHEL5 ~]$ whoami
paul
```

6.2.2. who

The **who** command will give you information about who is logged on to the system.

```
[paul@RHEL5 ~]$ who
root     tty1         2008-06-24 13:24
sandra   pts/0        2008-06-24 14:05 (192.168.1.34)
paul     pts/1        2008-06-24 16:23 (192.168.1.37)
```

6.2.3. who am i

With **who am i** the who command will display only the line pointing to your current session.

```
[paul@RHEL5 ~]$ who am i
paul     pts/1        2008-06-24 16:23 (192.168.1.34)
```

6.2.4. w

The **w** command shows you who is logged on and what they are doing.

```
$ w
 05:13:36 up 3 min,  4 users,  load average: 0.48, 0.72, 0.33
USER   TTY    FROM          LOGIN@   IDLE  JCPU  PCPU WHAT
root   tty1   -             05:11    2.00s 0.32s 0.27s find / -name shad
inge   pts/0  192.168.1.33  05:12    0.00s 0.02s 0.02s -ksh
laura  pts/1  192.168.1.34  05:12   46.00s 0.03s 0.03s -bash
paul   pts/2  192.168.1.34  05:13   25.00s 0.07s 0.04s top
```

6.2.5. id

The **id** command will give you your user id, primary group id and a list of the groups that you belong to.

```
root@laika:~# id
uid=0(root) gid=0(root) groups=0(root)
root@laika:~# su - brel
brel@laika:~$ id
uid=1001(brel) gid=1001(brel) groups=1001(brel),1008(chanson),11578(wolf)
```

6.3. Passwords

6.3.1. passwd

Passwords of users can be set with the **passwd** command. Users will have to provide their old password before entering the new one twice.

```
[harry@RHEL4 ~]$ passwd
Changing password for user harry.
Changing password for harry
(current) UNIX password:
New UNIX password:
BAD PASSWORD: it's WAY too short
New UNIX password:
Retype new UNIX password:
passwd: all authentication tokens updated successfully.
[harry@RHEL4 ~]$
```

As you can see, the passwd tool will do some basic verification to prevent users from using too simple passwords. The root user does not have to follow these rules (there will be a warning though). The root user also does not need to provide the old password before entering the new password twice.

6.3.2. /etc/shadow

User passwords are encrypted and kept in **/etc/shadow**. The /etc/shadow file is read only, and can only be read by root. We will see in the file permissions section how it is possible for users to change their password. For now, you will have to know that users can change their password with the **/usr/bin/passwd** command.

```
[root@RHEL5 ~]# tail /etc/shadow
inge:$1$yWMSimOV$YsYvcVKqByFVYLKnU3ncd0:14054:0:99999:7:::
ann:!!:14054:0:99999:7:::
frederik:!!:14054:0:99999:7:::
steven:!!:14054:0:99999:7:::
pascale:!!:14054:0:99999:7:::
geert:!!:14054:0:99999:7:::
wim:!!:14054:0:99999:7:::
sandra:!!:14054:0:99999:7:::
annelies:!!:14054:0:99999:7:::
laura:$1$Tvby1Kpa$lL.WzgobujUS3LClIRmdv1:14054:0:99999:7:::
```

The /etc/shadow file contains nine colums, seperated by a colon. The nine fields contain (from left to right) the user name, the encrypted password (note that only inge and laura have an encrypted password), the day the password was last changed (day 1 is January 1, 1970), number of days the password has to be left unchanged, password expiry day, warning number of days before password expiry, number of days after expiry before disabling the account, the day the account was disabled (since 1970 again). The last field has no meaning yet.

6.3.3. password encryption

6.3.3.1. encryption with passwd

Passwords are stored in an encrypted format. This encryption is done by the **crypt** function. The easiest (and recommended) way to add a user with a password to the system is to add the user with the **useradd -m user** command, and then set the user's password with **passwd**.

```
[root@RHEL4 ~]# useradd -m xavier
[root@RHEL4 ~]# passwd xavier
Changing password for user xavier.
New UNIX password:
Retype new UNIX password:
passwd: all authentication tokens updated successfully.
[root@RHEL4 ~]#
```

6.3.3.2. encryption with openssl

Another way to create users with a password is to use the -p option of useradd, but that option requires an encrypted password. You can generate this encrypted password with the **openssl passwd** command.

```
[root@RHEL4 ~]# openssl passwd stargate
ZZNX16QZVgUQg
[root@RHEL4 ~]# useradd -m -p ZZNX16QZVgUQg mohamed
```

6.3.3.3. encryption with crypt

A third option is to create your own C program using the crypt function, and compile this into a command.

```
[paul@laika ~]$ cat MyCrypt.c
#include <stdio.h>
#include <unistd.h>

int main(int argc, char** argv)
{
  printf("%s\n", crypt(argv[1], argv[2] ));
  return 0;
}
```

This little program can be compiled with g++ like this.

```
[paul@laika ~]$ g++ MyCrypt.c -o MyCrypt -lcrypt
```

To use it, we need to give two parameters to MyCript. The first is the unencrypted password, the second is the salt. The salt is used to perturb the encryption algorithm in one of 4096 different ways. This variation prevents two users with the same password from having the same entry in /etc/shadow.

```
paul@laika:~$ ./MyCrypt stargate 12
12L4FoTS3/k9U
paul@laika:~$ ./MyCrypt stargate 01
01Y.yPnlQ6R.Y
paul@laika:~$ ./MyCrypt stargate 33
330asFUbzgVeg
paul@laika:~$ ./MyCrypt stargate 42
42XFxoT4R75gk
```

Did you notice that the first two characters of the password are the salt ?

The standard output of the crypt function is using the DES algorithm, which is old and can be cracked in minutes. A better method is to use MD5 passwords, which can be recognized by a salt starting with $1$.

```
paul@laika:~$ ./MyCrypt stargate '$1$12'
$1$12$xUIQ4116Us.Q5Osc2Khbm1
paul@laika:~$ ./MyCrypt stargate '$1$01'
$1$01$yNs8brjp4b4TEw.v9/IlJ/
paul@laika:~$ ./MyCrypt stargate '$1$33'
$1$33$tLh/Ldy2wskdKAJR.Ph4M0
paul@laika:~$ ./MyCrypt stargate '$1$42'
$1$42$Hb3nvP0KwHSQ7fQmIlY7R.
```

The MD5 salt can be up to eight characters long. The salt is displayed in /etc/shadow between the second and third $, so never use the password as the salt!

```
paul@laika:~$ ./MyCrypt stargate '$1$stargate'
$1$stargate$qqxoLqiSVNvGr5ybMxEVM1
```

6.3.4. password defaults

6.3.4.1. /etc/login.defs

The **/etc/login.defs** file contains some default settings for user passwords like password aging and length settings. (You will also find the numerical limits of user id's and group id's and whether or not a home directory should be created by default).

```
[root@RHEL4 ~]# grep -i pass /etc/login.defs
# Password aging controls:
# PASS_MAX_DAYS  Maximum number of days a password may be used.
# PASS_MIN_DAYS  Minimum number of days allowed between password changes.
# PASS_MIN_LEN   Minimum acceptable password length.
# PASS_WARN_AGE  Number of days warning given before a password expires.
PASS_MAX_DAYS   99999
PASS_MIN_DAYS   0
PASS_MIN_LEN    5
PASS_WARN_AGE   7
```

6.3.4.2. chage

The **chage** command can be used to set an expiration date for a user account (-E), set a mimimum (-m) and maximum (-M) password age, a password expiration date, and set the number of warning days before the password expiration date. A lot of this functionality is also available via the passwd command. The -l option of chage will list these settings for a user.

```
[root@RHEL4 ~]# chage -l harry
Minimum:        0
Maximum:        99999
Warning:        7
Inactive:       -1
Last Change:            Jul 23, 2007
Password Expires:       Never
Password Inactive:      Never
Account Expires:        Never
[root@RHEL4 ~]#
```

6.3.5. disabling a password

Passwords in /etc/shadow cannot begin with an exclamation mark. When the second field in /etc/passwd starts with an exclamation mark, then the password can not be used.

Using this feature is often called **locking**, **disabling** or **suspending** a user account. Bisides vi (or vipw) you can also accomplish this with **usermod**.

The first line in the next screenshot will disable the password of user harry, making it impossible for harry to authenticate using this password.

```
[root@RHEL4 ~]# usermod -L harry
[root@RHEL4 ~]# tail -1 /etc/shadow
harry:!$1$143TO9IZ$RLm/FpQkpDrV4/Tkhku5e1:13717:0:99999:7:::
```

The root user (and users with sudo rights on su) will still be able to su to harry (because the password is not needed here). Note also that harry will still be able to login if he set up passwordless ssh!

```
[root@RHEL4 ~]# su - harry
[harry@RHEL4 ~]$
```

You can unlock the account again with **usermod -U**.

Watch out for tiny differences in the command line options of passwd, usermod and useradd on different distributions! Verify the local files when using features like 'disabling', 'suspending' or 'locking' users and passwords!

6.3.6. editing local files

If after knowing all these commands for password management you still want to edit the /etc/passwd or /etc/shadow manually, then use **vipw** instead of vi(m) directly. The vipw tool will do proper locking of the file.

```
[root@RHEL5 ~]# vipw /etc/passwd
vipw: the password file is busy (/etc/ptmp present)
```

6.4. About Home Directories

6.4.1. creating home directories

The easy way to create a home directory is to supply the -m option with **useradd** (it is likely set as a default option on Linux).

The not so easy way is to create a home directory manually with **mkdir**, which also requires setting the owner and the permissions on the directory with **chmod** and **chown** (both commands are discussed in detail in another chapter).

```
[root@RHEL5 ~]# mkdir /home/laura
[root@RHEL5 ~]# chown laura:laura /home/laura
[root@RHEL5 ~]# chmod 700 /home/laura
[root@RHEL5 ~]# ls -ld /home/laura/
drwx------ 2 laura laura 4096 Jun 24 15:17 /home/laura/
```

6.4.2. /etc/skel/

When using useradd with the -m option, then the **/etc/skel/** directory is copied to the newly created home directory. The /etc/skel/ directory contains some (usually hidden) files that contain profile settings and default values for applications. In this way /etc/skel/ serves as a default home directory and as a default user profile.

```
[root@RHEL5 ~]# ls -la /etc/skel/
total 48
```

```
drwxr-xr-x  2 root root  4096 Apr  1 00:11 .
drwxr-xr-x 97 root root 12288 Jun 24 15:36 ..
-rw-r--r--  1 root root    24 Jul 12  2006 .bash_logout
-rw-r--r--  1 root root   176 Jul 12  2006 .bash_profile
-rw-r--r--  1 root root   124 Jul 12  2006 .bashrc
```

6.4.3. deleting home directories

The -r option of userdel will make sure that the home directory is deleted together with the user account.

```
[root@RHEL5 ~]# ls -ld /home/wim/
drwx------ 2 wim wim 4096 Jun 24 15:19 /home/wim/
[root@RHEL5 ~]# userdel -r wim
[root@RHEL5 ~]# ls -ld /home/wim/
ls: /home/wim/: No such file or directory
```

6.5. User Shell

6.5.1. login shell

The **/etc/passwd** file determines the login shell for the user. In the screenshot below you can see that user annelies receives the /bin/bash shell, and user laura receives the /bin/ksh shell when they login.

```
[root@RHEL5 ~]# tail -2 /etc/passwd
annelies:x:527:533:sword fighter:/home/annelies:/bin/bash
laura:x:528:534:art dealer:/home/laura:/bin/ksh
```

You can use the usermod command to change the shell for a user.

```
[root@RHEL5 ~]# usermod -s /bin/bash laura
[root@RHEL5 ~]# tail -1 /etc/passwd
laura:x:528:534:art dealer:/home/laura:/bin/bash
```

6.5.2. chsh

Users can change their own login shell with the **chsh** command. User harry here is first obtaining a list of available shells (the user could have done a cat **/etc/shells**) and then changes his login shell to the **Korn shell** (/bin/ksh). At the next login, harry will default into ksh instead of bash.

```
[harry@RHEL4 ~]$ chsh -l
/bin/sh
/bin/bash
/sbin/nologin
```

```
/bin/ash
/bin/bsh
/bin/ksh
/usr/bin/ksh
/usr/bin/pdksh
/bin/tcsh
/bin/csh
/bin/zsh
[harry@RHEL4 ~]$ chsh -s /bin/ksh
Changing shell for harry.
Password:
Shell changed.
[harry@RHEL4 ~]$
```

6.6. Switch users with su

The **su** command allows a user to run a shell as another user. Running a shell as another user requires that you know the password of the other user, unless you are root. The root user can become any other user without knowing the user's password.

```
[paul@RHEL4b ~]$ su harry
Password:
[harry@RHEL4b paul]$ su root
Password:
[root@RHEL4b paul]# su serena
[serena@RHEL4b paul]$
```

By default, the su command keeps the same shell environment. To become another user and also get the target user's environment, issue the **su -** command followed by the target username.

```
[paul@RHEL4b ~]$ su - harry
Password:
[harry@RHEL4b ~]$
```

When no username is provided to su or su - then the command will assume root is the target.

```
[harry@RHEL4b ~]$ su -
Password:
[root@RHEL4b ~]#
```

6.7. Run a program as another user

6.7.1. About sudo

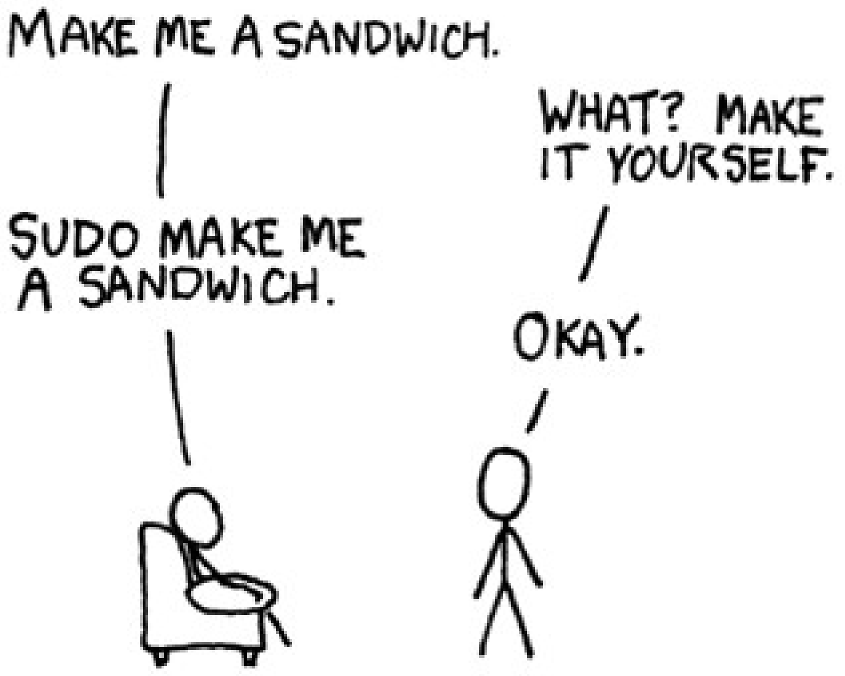
The sudo program allows a user to start a program with the credentials of another user. Before this works, the system administrator has to set up the **/etc/sudoers** file.

This can be useful to delegate administrative tasks to another user (without giving the root password).

The screenshot below shows the usage of sudo. User paul received the right to run useradd with the credentials of root. This allows paul to create new users on the system without becoming root and without knowing the root password.

```
paul@laika:~$ useradd -m inge
useradd: unable to lock password file
paul@laika:~$ sudo useradd -m inge
[sudo] password for paul:
paul@laika:~$
```

Image copied from **xkcd.com**.



6.7.2. setuid on sudo

The sudo binary has the setuid bit set, so any user can run it with effective userid set as root.

```
paul@laika:~$ ls -l `which sudo`
-rwsr-xr-x 2 root root 107872 2008-05-15 02:41 /usr/bin/sudo
paul@laika:~$
```

6.7.3. visudo

Check the man page of **visudo** before playing with the /etc/sudoers file.

6.7.4. sudo su

On some linux systems like Ubuntu and Kubuntu, the root user does not have a password set. This means that it is not possible to login as root (extra security). To perform tasks as root, the first user is given all **sudo rights** via the **/etc/sudoers**. In fact all users that are members of the admin group can use sudo to run all commands as root.

```
root@laika:~# grep admin /etc/sudoers
# Members of the admin group may gain root privileges
%admin ALL=(ALL) ALL
```

The end result of this is that the user can type **sudo su -** and become root without having to enter the root password. The sudo command does require you to enter your own password. Thus the password prompt in the screenshot below is for sudo, not for su.

```
paul@laika:~$ sudo su -
Password:
root@laika:~#
```

6.8. Practice Users

1. Create the users Serena Williams, Venus Williams and Justine Henin. all of them with password set to stargate, with username (lowercase!) as their first name, and their full name in the comment. Verify that the users and their home directory are properly created.

2. Create a user called kornuser, give him the Korn shell (/bin/ksh) as his default shell. Log on with this user (on a command line or in a tty).

3. Create a user named Einstime without home directory, give him /bin/date as his default logon shell. What happens when you log on with this user ? Can you think of a useful real world example for changing a user's login shell to an application ?

4. Try the commands who, whoami, who am i, w, id, echo $USER $UID .

5a. Lock the Venus user account with usermod.

5b. Use passwd -d to disable the serena password. Verify the serena line in /etc/shadow before and after disabling.

5c. What is the difference between locking a user account and disabling a user account's password ?

6. As root change the password of Einstime to stargate.

7. Now try changing the password of serena to serena as serena.

8. Make sure every new user needs to change his password every 10 days.

9. Set the warning number of days to four for the kornuser.

10a. Set the password of two seperate users to stargate. Look at the encrypted stargate's in /etc/shadow and explain.

10b. Take a backup as root of /etc/shadow. Use vi to copy an encrypted stargate to another user. Can this other user now log on with stargate as a password ?

11. Put a file in the skeleton directory and check whether it is copied to user's home directory. When is the skeleton directory copied ?

12. Why use vipw instead of vi ? What could be the problem when using vi or vim ?

13. Use chsh to list all shells, and compare to cat /etc/shells. Change your login shell to the Korn shell, log out and back in. Now change back to bash.

14. Which useradd option allows you to name a home directory ?

15. How can you see whether the password of user harry is locked or unlocked ? Give a solution with grep and a solution with passwd.

6.9. Solutions to the practice

1. Create the users Serena Williams, Venus Williams and Justine Henin. all of them with password set to stargate, with username (lowercase) as their first name, and their full name in the comment. Verify that the users and their home directory are properly created.

useradd -c "Serena Williams" serena ; passwd serena

useradd -c "Venus Williams" venus ; passwd venus

useradd -c "Justine Henin" justine ; passwd justine

tail /etc/passwd ; tail /etc/shadow ; ls /home

Keep user logon names in lowercase!

2. Create a user called kornuser, give him the Korn shell (/bin/ksh) as his default shell. Log on with this user (on a command line or in a tty).

useradd -s /bin/ksh kornuser ; passwd kornuser

3. Create a user named Einstime without home directory, give him /bin/date as his default logon shell. What happens when you log on with this user ? Can you think of a useful real world example for changing a user's login shell to an application ?

useradd -s /bin/date einstime ; passwd einstime

It can be useful when users need to access only one application on the server. Just logging on opens the application for them, and closing the application automatically logs them off.

4. Try the commands who, whoami, who am i, w, id, echo $USER $UID .

who ; whoami ; who am i ; w ; id ; echo $USER $UID

5a. Lock the venus user account with usermod.

usermod -L venus

5b. Use passwd -d to disable the serena password. Verify the serena line in /etc/shadow before and after disabling.

grep serena /etc/shadow; passwd -d serena ; grep serena /etc/shadow

5c. What is the difference between locking a user account and disabling a user account's password ?

Locking will prevent the user from logging on to the system with his password (by putting a ! in front of the password in /etc/shadow). Disbling with passwd will erase the password from /etc/shadow.

6. As root change the password of Einstime to stargate.

Log on as root and type: passwd Einstime

7. Now try changing the password of serena to serena as serena.

log on as serena, then execute: passwd serena... it should fail!

8. Make sure every new user needs to change his password every 10 days.

For an existing user: chage -M 10 serena

For all new users: vi /etc/login.defs (and change PASS_MAX_DAYS to 10)

9. Set the warning number of days to four for the kornuser.

chage -W 4 kornuser

10a. Set the password of two seperate users to stargate. Look at the encrypted stargate's in /etc/shadow and explain.

If you used passwd, then the salt will be different for the two encrypted passwords.

10b. Take a backup as root of /etc/shadow. Use vi to copy an encrypted stargate to another user. Can this other user now log on with stargate as a password ?

Yes.

11. Put a file in the skeleton directory and check whether it is copied to user's home directory. When is the skeleton directory copied ?

When you create a user account with a new home directory.

12. Why use vipw instead of vi ? What could be the problem when using vi or vim ?

vipw will give a warning when someone else is already using vipw on that file.

13. Use chsh to list all shells, and compare to cat /etc/shells. Change your login shell to the Korn shell, log out and back in. Now change back to bash.

On Red Hat Enterprise Linux: chsh -l

14. Which useradd option allows you to name a home directory ?

-d

15. How can you see whether the password of user harry is locked or unlocked ? Give a solution with grep and a solution with passwd.

grep harry /etc/shadow

passwd -S harry

Chapter 7. Introduction to Groups

7.1. About Groups

Users can be listed in groups. Groups allow you to set permissions on the group level instead of setting permissions for every individual users. Every Unix or Linux distribution will have a graphical tool to manage groups. Novice users are adviced to use this graphical tool. More experienced users can use commandline tools to manage users, but be careful: some distribution do not allow the mixed use of GUI and CLI tools to manage groups (YaST in Novell Suse). Senior administrators can edit the relevant files directly with vi or vigr.

7.2. groupadd

Groups can be created with the **groupadd** command. The example below shows the creation of five (empty) groups.

```
root@laika:~# groupadd tennis
root@laika:~# groupadd football
root@laika:~# groupadd snooker
root@laika:~# groupadd formula1
root@laika:~# groupadd salsa
```

7.3. /etc/group

Users can be a member of several groups. Group membership is defined by the **/etc/group** file.

```
root@laika:~# tail -5 /etc/group
tennis:x:1006:
football:x:1007:
snooker:x:1008:
formula1:x:1009:
salsa:x:1010:
root@laika:~#
```

Before the colon is the group's name. The second field is the group's (encrypted) password (can be empty). The third field is the group identification or **GID**. The fourth field is the list of members, these groups have no members.

7.4. usermod

Group membership can be modified with the useradd or **usermod** command.

```
root@laika:~# usermod -a -G tennis inge
root@laika:~# usermod -a -G tennis katrien
```

```
root@laika:~# usermod -a -G salsa katrien
root@laika:~# usermod -a -G snooker sandra
root@laika:~# usermod -a -G formula1 annelies
root@laika:~# tail -5 /etc/group
tennis:x:1006:inge,katrien
football:x:1007:
snooker:x:1008:sandra
formula1:x:1009:annelies
salsa:x:1010:katrien
root@laika:~#
```

Be careful when using usermod to add people to groups. The usermod command will by default **remove** users from the groups that are not listed in the command! Using the **-a** (append) switch prevents this behaviour.

7.5. groupmod

You can change the group name with the **groupmod** command.

```
root@laika:~# groupmod -n darts snooker
root@laika:~# tail -5 /etc/group
tennis:x:1006:inge,katrien
football:x:1007:
formula1:x:1009:annelies
salsa:x:1010:katrien
darts:x:1008:sandra
```

7.6. groupdel

You can permanently remove a group with the **groupdel** command.

```
root@laika:~# groupdel tennis
root@laika:~#
```

7.7. groups

A user can type the **groups** command to see a list of groups where the user belongs to.

```
[harry@RHEL4b ~]$ groups
harry sports
[harry@RHEL4b ~]$
```

7.8. gpasswd

You can delegate control of group membership to another user with the **gpasswd** command. In the example below we delegate permissions to add and remove group members to the sports group to serena. Then we su to serena and add harry to the sports group.

```
[root@RHEL4b ~]# gpasswd -A serena sports
[root@RHEL4b ~]# su - serena
[serena@RHEL4b ~]$ id harry
uid=516(harry) gid=520(harry) groups=520(harry)
[serena@RHEL4b ~]$ gpasswd -a harry sports
Adding user harry to group sports
[serena@RHEL4b ~]$ id harry
uid=516(harry) gid=520(harry) groups=520(harry),522(sports)
[serena@RHEL4b ~]$ tail -1 /etc/group
sports:x:522:serena,venus,harry
[serena@RHEL4b ~]$
```

Group administrators do not need to be a member of the group. They can even remove themselves from the group, this does not influence their ability to add or remove members.

```
[serena@RHEL4b ~]$ gpasswd -d serena sports
Removing user serena from group sports
[serena@RHEL4b ~]$ exit
```

Information about group administrators is kept in the **/etc/gshadow** file.

```
[root@RHEL4b ~]# tail -1 /etc/gshadow
sports:!:serena:venus,harry
[root@RHEL4b ~]#
```

7.9. vigr

Similar to vipw, the **vigr** must be used to manually edit the /etc/group file, since it will do proper locking of the file. Only experienced senior administrators should use vi or vigr to manage groups.

7.10. Practice: Groups

1. Create the groups tennis, football and sports.

2. In one command, make venus a member of tennis and sports.

3. Rename the football group to foot.

4. Use vi to add serena to the tennis group.

5. Use the id command to verify that serena is a member of tennis.

6. Make someone responsible for managing group membership of foot and sports. Test that it works.

7.11. Solution: Groups

1. Create the groups tennis, football and sports.

groupadd tennis ; groupadd football ; groupadd sports

2. In one command, make venus a member of tennis and sports.

usermod -a -G tennis,sports venus

3. Rename the football group to foot.

groupmod -n foot football

4. Use vi to add serena to the tennis group.

vi /etc/group

5. Use the id command to verify that serena is a member of tennis.

id (and after logoff logon serena should be member)

6. Make someone responsible for managing group membership of foot and sports. Test that it works.

gpasswd -A (to make member)

gpasswd -a (to add member)

Chapter 8. Standard File Permissions

8.1. file ownership

8.1.1. user owner and group owner

The **users** and **groups** of a system can be locally managed in **/etc/passwd** and **/etc/group**, or they can be in a NIS, LDAP or Samba domain. These users and groups can **own** files. Actually, every file has a **user owner** and a **group owner**, as can be seen in the following screenshot.

```
paul@RHELv4u4:~/test$ ls -l
total 24
-rw-rw-r--  1 paul paul  17 Feb  7 11:53 file1
-rw-rw-r--  1 paul paul 106 Feb  5 17:04 file2
-rw-rw-r--  1 paul proj 984 Feb  5 15:38 data.odt
-rw-r--r--  1 root root   0 Feb  7 16:07 stuff.txt
paul@RHELv4u4:~/test$
```

User paul owns three files, two of those are also owned by the group paul, data.odt is owned by the group proj. The root user owns the file stuff.txt, as does the group root.

8.1.2. chgrp

You can change the group owner of a file using the **chgrp** command.

```
root@laika:/home/paul# touch FileForPaul
root@laika:/home/paul# ls -l FileForPaul
-rw-r--r-- 1 root root 0 2008-08-06 14:11 FileForPaul
root@laika:/home/paul# chgrp paul FileForPaul
root@laika:/home/paul# ls -l FileForPaul
-rw-r--r-- 1 root paul 0 2008-08-06 14:11 FileForPaul
```

8.1.3. chown

The user owner of a file can be changed with **chown** command.

```
root@laika:/home/paul# ls -l FileForPaul
-rw-r--r-- 1 root paul 0 2008-08-06 14:11 FileForPaul
root@laika:/home/paul# chown paul FileForPaul
root@laika:/home/paul# ls -l FileForPaul
-rw-r--r-- 1 paul paul 0 2008-08-06 14:11 FileForPaul
```

You can also use chown to change both the user owner and the group owner.

```
root@laika:/home/paul# ls -l FileForPaul
-rw-r--r-- 1 paul paul 0 2008-08-06 14:11 FileForPaul
root@laika:/home/paul# chown root:project42 FileForPaul
root@laika:/home/paul# ls -l FileForPaul
-rw-r--r-- 1 root project42 0 2008-08-06 14:11 FileForPaul
```

8.2. special files

When you use **ls -l**, then you can see, for each file, ten characters before the user and group owner. The first character tells us the type of file. Regular files get a **-**, directories get a **d**, symbolic links are shown with an **l**, pipes get a **p**, character devices a **c**, block devices a **b** and sockets an **s**.

Table 8.1. Unix special files

first character	file type
-	normal file
d	directory
l	symbolic link
p	named pipe
b	block device
c	character device
s	socket

8.3. permissions

The nine characters following the file type denote the permissions in three triplets. A permission can be **r** for read access, **w** for write access and **x** for execute. You need the r permission to list (ls) the contents of a directory and x permission to enter (cd) a directory, and you need the w permission to create files in or remove files from a directory.

Table 8.2. standard Unix file permissions

permission	on a file	on a directory
r (read)	read file contents (cat)	read directory contents (ls)
w (write)	change file contents (vi)	create files in (touch)
x (execute)	execute the file	enter the directory (cd)

Some example combinations on files and directories in this screenshot. The name of the file explains the permissions.

```
paul@laika:~/perms$ ls -lh
total 12K
drwxr-xr-x 2 paul paul 4.0K 2007-02-07 22:26 AllEnter_UserCreateDelete
-rwxrwxrwx 1 paul paul    0 2007-02-07 22:21 EveryoneFullControl.txt
-r--r----- 1 paul paul    0 2007-02-07 22:21 OnlyOwnersRead.txt
-rwxrwx--- 1 paul paul    0 2007-02-07 22:21 OwnersAll_RestNothing.txt
dr-xr-x--- 2 paul paul 4.0K 2007-02-07 22:25 UserAndGroupEnter
dr-x------ 2 paul paul 4.0K 2007-02-07 22:25 OmlyUserEnter
paul@laika:~/perms$
```

It is important to know that the first triplet represents the **user owner**, the second is the **group owner**, and the third triplet is all the **other** users that are not the user owner and are not a member of the group owner.

8.4. setting permissions (chmod)

Permissions can be changed with **chmod**. The first example gives the user owner execute permissions.

```
paul@laika:~/perms$ ls -l permissions.txt
-rw-r--r-- 1 paul paul 0 2007-02-07 22:34 permissions.txt
paul@laika:~/perms$ chmod u+x permissions.txt
paul@laika:~/perms$ ls -l permissions.txt
-rwxr--r-- 1 paul paul 0 2007-02-07 22:34 permissions.txt
```

This example removes the group owners read permission.

```
paul@laika:~/perms$ chmod g-r permissions.txt
paul@laika:~/perms$ ls -l permissions.txt
-rwx---r-- 1 paul paul 0 2007-02-07 22:34 permissions.txt
```

This example removes the others read permission.

```
paul@laika:~/perms$ chmod o-r permissions.txt
paul@laika:~/perms$ ls -l permissions.txt
-rwx------ 1 paul paul 0 2007-02-07 22:34 permissions.txt
```

This example gives all of them the write permission.

```
paul@laika:~/perms$ chmod a+w permissions.txt
paul@laika:~/perms$ ls -l permissions.txt
-rwx-w--w- 1 paul paul 0 2007-02-07 22:34 permissions.txt
```

You don't even have to type the a.

```
paul@laika:~/perms$ chmod +x permissions.txt
paul@laika:~/perms$ ls -l permissions.txt
-rwx-wx-wx 1 paul paul 0 2007-02-07 22:34 permissions.txt
```

You can also set explicit permissions.

```
paul@laika:~/perms$ chmod u=rw permissions.txt
paul@laika:~/perms$ ls -l permissions.txt
-rw--wx-wx 1 paul paul 0 2007-02-07 22:34 permissions.txt
```

Feel free to make any kind of combinations.

```
paul@laika:~/perms$ chmod u=rw,g=rw,o=r permissions.txt
paul@laika:~/perms$ ls -l permissions.txt
-rw-rw-r-- 1 paul paul 0 2007-02-07 22:34 permissions.txt
```

Even the fishy combinations are accepted by chmod.

```
paul@laika:~/perms$ chmod u=rwx,ug+rw,o=r permissions.txt
paul@laika:~/perms$ ls -l permissions.txt
-rwxrw-r-- 1 paul paul 0 2007-02-07 22:34 permissions.txt
```

8.5. setting octal permissions

Most Unix administrators will use the **old school** octal system to talk about and set permissions. Look at the triplet bitwise, equaling r to 4, w to 2 and x to 1.

Table 8.3. Octal permissions

binary	octal	permission
000	0	---
001	1	--x
010	2	-w-
011	3	-wx
100	4	r--
101	5	r-x
110	6	rw-
111	7	rwx

This makes **777** equal to rwxrwxrwx and by the same logic has 654 mean rw-r-xr-- . The **chmod** command will accept these numbers.

```
paul@laika:~/perms$ chmod 777 permissions.txt
paul@laika:~/perms$ ls -l permissions.txt
-rwxrwxrwx 1 paul paul 0 2007-02-07 22:34 permissions.txt
paul@laika:~/perms$ chmod 664 permissions.txt
paul@laika:~/perms$ ls -l permissions.txt
-rw-rw-r-- 1 paul paul 0 2007-02-07 22:34 permissions.txt
paul@laika:~/perms$ chmod 750 permissions.txt
paul@laika:~/perms$ ls -l permissions.txt
```

```
-rwxr-x--- 1 paul paul 0 2007-02-07 22:34 permissions.txt
```

8.6. umask

When creating a file or directory, a set of default permissions are applied. These default permissions are determined by the **umask**. The umask specifies permissions that you do not want set by default. You can display the umask with the umask command.

```
[Harry@RHEL4b ~]$ umask
0002
[Harry@RHEL4b ~]$ touch test
[Harry@RHEL4b ~]$ ls -l test
-rw-rw-r--  1 Harry Harry 0 Jul 24 06:03 test
[Harry@RHEL4b ~]$
```

As you can see, the file is also not executable by default. This is a general security feature among Unixes, newly created files are never executable by default. You have to explicitely do a **chmod +x** to make a file executable. This also means that the 1 bit in the umask has no meaning, a umask of 0022 is the same as 0033.

8.7. Practice: File Permissions

1. As normal user, create a directory ~/permissions. Create a file owned by yourself in there.

2. Copy a file owned by root from /etc/ to your permissions dir, who owns this file now ?

3. As root, create a file in the users ~/permissions directory.

4. As normal user, look at who owns this file created by root.

5. Change the ownership of all files in ~/permissions to yourself.

6. Make sure you have all rights to these files, and others can only read.

7. With chmod, is 770 the same as rwxrwx--- ?

8. With chmod, is 664 the same as r-xr-xr-- ?

9. With chmod, is 400 the same as r-------- ?

10. With chmod, is 734 the same as rwxr-xr-- ?

11a. Display the umask in octal and in symbolic form.

11b. Set the umask to 077, but use the symbolic format to set it. Verify that this works.

12. Create a file as root, give only read to others. Can a normal user read this file ? Test writing to this file with vi.

13a. Create a file as normal user, give only read to others. Can another normal user read this file ? Test writing to this file with vi.

13b. Can root read this file ? Can root write to this file with vi ?

14. Create a directory that belongs to a group, where every member of that group can read and write to files, and create files. Make sure that people can only delete their own files.

8.8. Solutions: File Permissions

1. As normal user, create a directory ~/permissions. Create a file owned by yourself in there.

```
mkdir ~/permissions ; touch ~/permissions/myfile.txt
```

2. Copy a file owned by root from /etc/ to your permissions dir, who owns this file now ?

```
cp /etc/hosts ~/permissions/
```

The copy is owned by you.

3. As root, create a file in the users ~/permissions directory.

```
(become root)# touch /home/username/permissions/rootfile
```

4. As normal user, look at who owns this file created by root.

```
ls -l ~/permissions
```

The file created by root is owned by root.

5. Change the ownership of all files in ~/permissions to yourself.

```
chown user ~/permissions/*
```

You cannot become owner of the file that belongs to root.

6. Make sure you have all rights to these files, and others can only read.

```
chmod 644 (on files)
```

```
chmod 755 (on directories)
```

7. With chmod, is 770 the same as rwxrwx--- ?

yes

8. With chmod, is 664 the same as r-xr-xr-- ?

No

9. With chmod, is 400 the same as r-------- ?

yes

10. With chmod, is 734 the same as rwxr-xr-- ?

no

11a. Display the umask in octal and in symbolic form.

```
umask ; umask -S
```

11b. Set the umask to 077, but use the symbolic format to set it. Verify that this works.

```
umask -S u=rwx,go=
```

12. Create a file as root, give only read to others. Can a normal user read this file ? Test writing to this file with vi.

```
(become root)

# echo hello > /home/username/root.txt

# chmod 744 /home/username/root.txt

(become user)

vi ~/root.txt
```

13a. Create a file as normal user, give only read to others. Can another normal user read this file ? Test writing to this file with vi.

```
echo hello > file ; chmod 744 file
```

Yes, others can read this file

13b. Can root read this file ? Can root write to this file with vi ?

Yes, root can read and write to this file. Permissions do not apply to root.

14. Create a directory that belongs to a group, where every member of that group can read and write to files, and create files. Make sure that people can only delete their own files.

```
mkdir /home/project42 ; groupadd project42

chgrp project42 /home/project42 ; chmod 775 /home/project42
```

You can not yet do the last part of this exercise...

8.9. Sticky bit on directory

You can set the **sticky bit** on a directory to prevent users from removing files that they do not own as a user owner. The sticky bit is displayed at the same location as the x permission for others. The sticky bit is represented by a **t** (meaning x is also there) or a **T** (when there is no x for others).

```
root@RHELv4u4:~# mkdir /project55
root@RHELv4u4:~# ls -ld /project55
drwxr-xr-x  2 root root 4096 Feb  7 17:38 /project55
root@RHELv4u4:~# chmod +t /project55/
root@RHELv4u4:~# ls -ld /project55
drwxr-xr-t  2 root root 4096 Feb  7 17:38 /project55
root@RHELv4u4:~#
```

The sticky bit can also be set with octal permissions, it is binary 1 in the first of four triplets.

```
root@RHELv4u4:~# chmod 1775 /project55/
root@RHELv4u4:~# ls -ld /project55
drwxrwxr-t  2 root root 4096 Feb  7 17:38 /project55
root@RHELv4u4:~#
```

8.10. SetGID bit on directory

The **SetGID** can be used on directories to make sure that all files inside the directory are group owned by the group owner of the directory. The SetGID bit is displayed at the same location as the x permission for group owner. The SetGID bit is represented by an **s** (meaning x is also there) or a **S** (when there is no x for the group owner). Like this example shows, even though root does not belong to the group proj55, the files created by root in /project55 will belong to proj55 when the SetGID is set.

```
root@RHELv4u4:~# groupadd proj55
root@RHELv4u4:~# chown root:proj55 /project55/
root@RHELv4u4:~# chmod 2775 /project55/
root@RHELv4u4:~# touch /project55/fromroot.txt
root@RHELv4u4:~# ls -ld /project55/
drwxrwsr-x  2 root proj55 4096 Feb  7 17:45 /project55/
root@RHELv4u4:~# ls -l /project55/
total 4
-rw-r--r--  1 root proj55 0 Feb  7 17:45 fromroot.txt
root@RHELv4u4:~#
```

You can use the **find** command to find all **setgid** programs.

```
paul@laika:~$ find / -type d -perm -2000 2> /dev/null
/var/log/mysql
/var/log/news
/var/local
...
```

8.11. Practice: sticky and setGID on directory

1. Set up a directory, owned by the group sports.

2. Members of the sports group should be able to create files in this directory.

3. All files created in this directory should be group-owned by the sports group.

4. Users should be able to delete only their own user-owned files.

5. Test that this works!

6. If time permits (or if you are waiting for other students to finish this practice), read about file attributes in the man page of chattr and lsattr. Try setting the i attribute on a file and test that it works.

8.12. Solution: sticky and setGID on directory

1. Set up a directory, owned by the group sports.

```
groupadd sports ; mkdir /home/sports ; chown root:sports /home/sports
```

2. Members of the sports group should be able to create files in this directory.

```
chmod 770 /home/sports
```

3. All files created in this directory should be group-owned by the sports group.

```
chmod 2770 /home/sports
```

4. Users should be able to delete only their own user-owned files.

```
chmod +t /home/sports
```

5. Test that this works!

Log in with different users (group members and others and root), create files and watch the permissions. Try changing and deleting files...

6. If time permits (or if you are waiting for other students to finish this practice), read about file attributes in the man page of chattr and lsattr. Try setting the i attribute on a file and test that it works.

```
paul@laika:~$ sudo su -
[sudo] password for paul:
root@laika:~# mkdir attr
```

```
root@laika:~# cd attr/
root@laika:~/attr# touch file42
root@laika:~/attr# lsattr
------------------ ./file42
root@laika:~/attr# chattr +i file42
root@laika:~/attr# lsattr
----i------------- ./file42
root@laika:~/attr# rm -rf file42
rm: cannot remove `file42': Operation not permitted
root@laika:~/attr# chattr -i file42
root@laika:~/attr# rm -rf file42
root@laika:~/attr#
```

Chapter 9. File Links

9.1. about inodes

To understand links in a file system, you first have to understand what an **inode** is. When the file system stores a new file on the hard disk, it stores not only the contents (data) of the file, but also some extra properties like the name of the file, the creation date, the permissions, the owner of the file... and more. All this information (except the name of the file and the data) is stored in the inode of the file.

All the inodes are created when you create the file system (with mkfs). Most of them are unused and empty, each inode has a unique number (the inode number). You can see the inode numbers with the **ls -li** command.

```
paul@RHELv4u4:~/test$ touch file1
paul@RHELv4u4:~/test$ touch file2
paul@RHELv4u4:~/test$ touch file3
paul@RHELv4u4:~/test$ ls -li
total 12
817266 -rw-rw-r--  1 paul paul 0 Feb  5 15:38 file1
817267 -rw-rw-r--  1 paul paul 0 Feb  5 15:38 file2
817268 -rw-rw-r--  1 paul paul 0 Feb  5 15:38 file3
paul@RHELv4u4:~/test$
```

Three files created one after the other get three different inodes (the first column). All the information you see with this ls command resides in the inode, except for the filename (which is contained in the directory). Let's put some data in one of the files.

```
paul@RHELv4u4:~/test$ ls -li
total 16
817266 -rw-rw-r--  1 paul paul  0 Feb  5 15:38 file1
817270 -rw-rw-r--  1 paul paul 92 Feb  5 15:42 file2
817268 -rw-rw-r--  1 paul paul  0 Feb  5 15:38 file3
paul@RHELv4u4:~/test$ cat file2
It is winter now and it is very cold.
We do not like the cold, we prefer hot summer nights.
paul@RHELv4u4:~/test$
```

The data that is displayed by the cat commend is not in the inode, but somewhere else on the disk. But the inode contains a pointer to the data.

9.2. about directories

A **directory** is a special kind of file. It contains a table mapping filenames to inodes. Looking at our current directory with **ls -ali** will display the contents of the directory file.

```
paul@RHELv4u4:~/test$ ls -ali
total 32
817262 drwxrwxr-x   2 paul paul 4096 Feb  5 15:42 .
800768 drwx------  16 paul paul 4096 Feb  5 15:42 ..
817266 -rw-rw-r--   1 paul paul    0 Feb  5 15:38 file1
817270 -rw-rw-r--   1 paul paul   92 Feb  5 15:42 file2
817268 -rw-rw-r--   1 paul paul    0 Feb  5 15:38 file3
paul@RHELv4u4:~/test$
```

You can see five names, and the mapping to their five inodes. The dot **.** is a mapping to itself, and the dotdot **..** is a mapping to the parent directory. The three others are mappings to files.

9.3. hard links

When we create a **hard link** to a file with **ln**, then an extra entry is added in the directory. A new file name is mapped to an existing inode.

```
paul@RHELv4u4:~/test$ ln file2 hardlink_to_file2
paul@RHELv4u4:~/test$ ls -li
total 24
817266 -rw-rw-r--  1 paul paul  0 Feb  5 15:38 file1
817270 -rw-rw-r--  2 paul paul 92 Feb  5 15:42 file2
817268 -rw-rw-r--  1 paul paul  0 Feb  5 15:38 file3
817270 -rw-rw-r--  2 paul paul 92 Feb  5 15:42 hardlink_to_file2
paul@RHELv4u4:~/test$
```

Both files have the same inode, so they will always have the same permissions and the same owner. And they will both have the same content. Actually, both files are equal now, meaning you can safely remove the original file, the hardlinked file will remain. The inode contains a counter, counting the number of hard links to itself. When the counter drops to zero, then the inode is emptied.

You can use the **find** command to look for files with a certain inode. The screenshot below proves that / and /boot are different partitions, since every **inode** number is unique to the partition.

```
paul@RHELv4u4:~/test$ find / -inum 2 2> /dev/null
/
/boot
/var/lib/nfs/rpc_pipefs/lockd
/proc/self
paul@RHELv4u4:~/test$
```

9.4. symbolic links

Symbolic links (sometimes called **soft links**) do not link to inodes, but create a name to name mapping. Symbolic links are created with **ln -s**. As you can see below, the **symbolic link** gets an inode of its own.

```
paul@RHELv4u4:~/test$ ln -s file2 symlink_to_file2
paul@RHELv4u4:~/test$ ls -li
total 32
817273 -rw-rw-r--  1 paul paul  13 Feb  5 17:06 file1
817270 -rw-rw-r--  2 paul paul 106 Feb  5 17:04 file2
817268 -rw-rw-r--  1 paul paul   0 Feb  5 15:38 file3
817270 -rw-rw-r--  2 paul paul 106 Feb  5 17:04 hardlink_to_file2
817267 lrwxrwxrwx  1 paul paul   5 Feb  5 16:55 symlink_to_file2 -> file2
paul@RHELv4u4:~/test$
```

Permissions on a symbolic link have no meaning, since the permissions of the target apply. Hard links are limited to their own partition (because they point to an inode), symbolic links can link anywhere (other file systems, even networked).

9.5. removing links

Links can be removed with **rm**.

```
paul@laika:~$ touch data.txt
paul@laika:~$ ln -s data.txt sl_data.txt
paul@laika:~$ ln data.txt hl_data.txt
paul@laika:~$ rm sl_data.txt
paul@laika:~$ rm hl_data.txt
```

9.6. Practice: Links

1. Create two files named winter.txt and summer.txt, put some text in them.

2. Create a hard link to winter.txt named hlwinter.txt.

3. Display the inode numbers of these three files, the hard links should have the same inode.

4. Use the find command to list the two hardlinked files

5. Everything about a file is in the inode, except two things : name them!

6. Create a symbolic link to summer.txt called slsummer.txt.

7. Find all files with inode number 2. What does this information tell you ?

8. Look at the directories /etc/init.d/ /etc/rc.d/ /etc/rc3.d/ ... do you see the links ?

9. Look in /lib with ls -l...

9.7. Solutions: Links

1. Create two files named winter.txt and summer.txt, put some text in them.

```
echo cold > winter.txt ; echo hot > summer.txt
```

2. Create a hard link to winter.txt named hlwinter.txt.

```
ln winter.txt hlwinter.txt
```

3. Display the inode numbers of these three files, the hard links should have the same inode.

```
ls -li winter.txt summer.txt hlwinter.txt
```

4. Use the find command to list the two hardlinked files

```
find . -inum xyz
```

5. Everything about a file is in the inode, except two things : name them!

The name of the file is in a directory, and the contents is somewhere on the disk.

6. Create a symbolic link to summer.txt called slsummer.txt.

```
ln -s summer.txt slsummer.txt
```

7. Find all files with inode number 2. What does this information tell you ?

It tells you there is more than one inode table (one for every formatted partition + virtual file systems)

8. Look at the directories /etc/init.d/ /etc/rc.d/ /etc/rc3.d/ ... do you see the links ?

```
ls -l /etc/init.d
```

```
ls -l /etc/rc.d
```

```
ls -l /etc/rc3.d
```

9. Look in /lib with ls -l...

```
ls -l /lib
```

Chapter 10. Introduction to Scripting

10.1. About scripting

Bash has support for programming constructs that can be saved as scripts. These scripts in turn then become more bash commands. In fact, a lot of linux commands are scripts. This means that system administrators also need a basic knowledge of scripting to understand how their servers and their applications are started, updated, upgraded, patched, maintained, configured and removed.

10.2. Hello World

Just like in every programming course, we start with a simple Hello World script. The following script will output Hello World.

```
#!/bin/bash
# Hello World Script
echo Hello World
```

After creating this simple script in vi, you'll have to **chmod +x** the script to make it executable. And unless you add the scripts directory to your path, you'll have to type the path to the script for the shell to be able to find it.

```
[paul@RHEL4a ~]$ chmod +x hello_world
[paul@RHEL4a ~]$ ./hello_world
Hello World
[paul@RHEL4a ~]$
```

10.3. Variables

```
#!/bin/bash
var1=4
echo var1 = $var1
```

Scripts can contain variables, but since scripts are run in their own shell, the variables do not survive the end of the script.

```
[paul@RHEL4a ~]$ echo $var1

[paul@RHEL4a ~]$ ./vars
var1 = 4
[paul@RHEL4a ~]$ echo $var1

[paul@RHEL4a ~]$
```

Luckily you can force a script to run in the same shell, this is called sourcing a script.

```
[paul@RHEL4a ~]$ source ./vars
var1 = 4
[paul@RHEL4a ~]$ echo $var1
4
[paul@RHEL4a ~]$
```

The above is identical to the below.

```
[paul@RHEL4a ~]$ . ./vars
var1 = 4
[paul@RHEL4a ~]$ echo $var1
4
[paul@RHEL4a ~]$
```

10.4. Shell

You can never be sure which shell a user is running. A script that works flawlessly in bash, might not work in ksh or csh or dash. To instruct a shell to run your script in a certain shell, you can start your script with a shebang followed by the shell it is supposed to run in. This script will run in a bash shell.

```
#!/bin/bash
echo -n hello
echo A bash subshell `echo -n hello`
```

This script will run in a Korn shell (unless /bin/ksh is a link to /bin/bash). The **/etc/shells** file contains a list of shells on your system.

```
#!/bin/ksh
echo -n hello
echo a Korn subshell `echo -n hello`
```

Some user may perform setuid based script root spoofing. This is a rare but possible attack. To improve script security, and to avoid interpreter spoofing you need to add -- to #!/bin/bash, which disables further option processing, so bash will not accept any options.

```
#!/bin/bash -
```

or

```
#!/bin/bash --
```

Any arguments after the -- are treated as filenames and arguments. An argument of - is equivalent to --.

10.5. for loop

The example below shows the syntax of a classical **for loop** in bash.

```
for i in 1 2 4
do
   echo $i
done
```

An example of a for loop combined with an embedded shell to generate the list.

```
for file in `ls *.txt`
do
   cp $file $file.bak
   echo Backup of $file put in $file.bak
done
```

10.6. while loop

Below a simple example of a **while loop**.

```
let i=100;
while [ $i -ge 0 ] ;
do
   echo Counting down, from 100 to 0, now at $i;
   let i--;
done
```

10.7. until loop

Below a simple example of an **until loop**.

```
let i=100;
until [ $i -le 0 ] ;
do
   echo Counting down, from 100 to 1, now at $i;
   let i--;
done
```

10.8. parameters

A bash shell script can have parameters. The numbering you see in the script below continues if you have more parameters. You also have special parameters for the number of parameters, a string of all of them, and also the process id and the last error code. The man page of bash has a full list.

```
#!/bin/bash
echo The first argument is $1
echo The second argument is $2
echo The third argument is $3

echo \$ $$  PID of the script
echo \# $#  count arguments
echo \? $?  last error code
echo \* $*  all the arguments
```

Below is the output of the script above in action.

```
[paul@RHEL4a scripts]$ ./pars one two three
The first argument is one
The second argument is two
The third argument is three
$ 5610 PID of the script
# 3 count arguments
? 0 last error code
* one two three all the arguments
[paul@RHEL4a scripts]$ ./pars a b c
The first argument is a
The second argument is b
The third argument is c
$ 5611 PID of the script
# 3 count arguments
? 0 last error code
* a b c all the arguments
[paul@RHEL4a scripts]$ ./pars 1 2
The first argument is 1
The second argument is 2
The third argument is
$ 5612 PID of the script
# 2 count arguments
? 0 last error code
* 1 2 all the arguments
[paul@RHEL4a scripts]$
```

10.9. test [ ]

The **test** command can test whether something is true or false. Let's start by testing whether 10 is greater than 55.

```
[paul@RHEL4b ~]$ test 10 -gt 55 ; echo $?
1
[paul@RHEL4b ~]$
```

The test command returns 1 if the test fails. And as you see in the next screenshot, test returns 0 when a test succeeds.

```
[paul@RHEL4b ~]$ test 56 -gt 55 ; echo $?
0
[paul@RHEL4b ~]$
```

If you prefer true and false, then write the test like this.

```
[paul@RHEL4b ~]$ test 56 -gt 55 && echo true || echo false
true
[paul@RHEL4b ~]$ test 6 -gt 55 && echo true || echo false
false
```

The test command can also be written as square brackets, the screenshot below is identical to the one above.

```
[paul@RHEL4b ~]$ [ 56 -gt 55 ] && echo true || echo false
true
[paul@RHEL4b ~]$ [ 6 -gt 55 ] && echo true || echo false
false
```

Below are some example tests. Take a look at **man test** to see more options for tests.

```
[ -d foo ]              Does the directory foo exist ?
[ '/etc' = $PWD ]       Is the string /etc equal to the variable $PWD ?
[ $1 != 'secret' ]      Is the first parameter different from secret ?
[ 55 -lt $bar ]         Is 55 less than the value of $bar ?
[ $foo -ge 1000 ]       Is the value of $foo greater or equal to 1000 ?
[ "abc" < $bar ]  Does abc sort before the value of $bar ?
[ -f foo ]              Is foo a regular file ?
[ -r bar ]              Is bar a readable file ?
[ foo -nt bar ]         Is file foo newer than file bar ?
[ -o nounset ]          Is the shell option nounset set ?
```

Tests can be combined with logical AND and OR.

```
[paul@RHEL4b ~]$ [ 66 -gt 55 -a 66 -lt 500 ] && echo true || echo false
true
[paul@RHEL4b ~]$ [ 66 -gt 55 -a 660 -lt 500 ] && echo true || echo false
false
[paul@RHEL4b ~]$ [ 66 -gt 55 -o 660 -lt 500 ] && echo true || echo false
true
```

10.10. if if, then then, or else

The **if then else** construction is about choice. If a certain condition is met, then execute something, else execute something else. The example below tests whether a file exists, if the file exists then a proper message is echoed.

```
#!/bin/bash

if [ -f isit.txt ]
then echo isit.txt exists!
else echo isit.txt not found!
fi
```

If we name the above script 'choice', then it executes like this.

```
[paul@RHEL4a scripts]$ ./choice
isit.txt not found!
[paul@RHEL4a scripts]$ touch isit.txt
[paul@RHEL4a scripts]$ ./choice
isit.txt exists!
[paul@RHEL4a scripts]$
```

10.11. let

The **let** command allows for evalutation of arithmetic expressions.

```
[paul@RHEL4b ~]$ let x="3 + 4" ; echo $x
7
[paul@RHEL4b ~]$ let x="10 + 100/10" ; echo $x
20
[paul@RHEL4b ~]$ let x="10-2+100/10" ; echo $x
18
[paul@RHEL4b ~]$ let x="10*2+100/10" ; echo $x
30
```

The let command can also convert between different bases.

```
[paul@RHEL4b ~]$ let x="0xFF" ; echo $x
255
[paul@RHEL4b ~]$ let x="0xC0" ; echo $x
192
[paul@RHEL4b ~]$ let x="0xA8" ; echo $x
168
[paul@RHEL4b ~]$ let x="8#70" ; echo $x
56
[paul@RHEL4b ~]$ let x="8#77" ; echo $x
63
[paul@RHEL4b ~]$ let x="16#c0" ; echo $x
192
```

10.12. runtime input

You can ask the user for input with the **read** command in a script.

```
#!/bin/bash
echo -n Enter a number:
read number
```

10.13. sourcing a config file

```
[paul@RHEL4a scripts]$ cat myApp.conf
# The config file of myApp

# Enter the path here
myAppPath=/var/myApp

# Enter the number of quines here
quines=5

[paul@RHEL4a scripts]$ cat myApp.bash
#!/bin/bash
#
# Welcome to the myApp application
#

. ./myApp.conf

echo There are $quines quines

[paul@RHEL4a scripts]$ ./myApp.bash
There are 5 quines
[paul@RHEL4a scripts]$
```

10.14. case

You can sometimes simplify nested if statements with a case construct.

```
[paul@RHEL4b ~]$ ./help
What animal did you see ? lion
You better start running fast!
[paul@RHEL4b ~]$ ./help
What animal did you see ? dog
Don't worry, give it a cookie.
[paul@RHEL4b ~]$ cat help
#!/bin/bash
#
# Wild Animals Helpdesk Advice
#
echo -n "What animal did you see ? "
read animal
case $animal in
        "lion" | "tiger")
                echo "You better start running fast!"
        ;;
        "cat")
                echo "Let that mouse go..."
        ;;
        "dog")
                echo "Don't worry, give it a cookie."
        ;;
        "chicken" | "goose" | "duck" )
                echo "Eggs for breakfast!"
        ;;
        "liger")
                echo "Approach and say 'Ah you big fluffy kitty...'."
        ;;
        "babelfish")
                echo "Did it fall out your ear ?"
        ;;
        *)
```

```
                echo "You discovered an unknown animal, name it!"
        ;;
esac
[paul@RHEL4b ~]$
```

10.15. shopt

You can toggle the values of variables controlling optional shell behavior with the **shopt** built-in shell command. The example below first verifies whether the cdspell option is set, it is not. The next shopt command sets the value, and the third shopt command verifies that the option really is set. You can now use minor spelling mistakes in the cd command. The man page of bash has a complete list of options.

```
paul@laika:~$ shopt -q cdspell ; echo $?
1
paul@laika:~$ shopt -s cdspell
paul@laika:~$ shopt -q cdspell ; echo $?
0
paul@laika:~$ cd /Etc
/etc
paul@laika:/etc$
```

10.16. Practice : scripts

0. Give each script a different name, keep them for later!

1. Write a script that receives four parameters, and outputs them in reverse order.

2. Write a script that receives two parameters (two filenames) and outputs whether those files exist.

3. Write a script that counts the number of files ending in .txt in the current directory.

4. Write a script that asks for two numbers, and outputs the sum and product (as shown here).

```
Enter a number: 5
Enter another number: 2

Sum:        5 + 2 = 7
Product:    5 x 2 = 10
```

5. Improve the previous script to test that the numbers are between 1 and 100, exit with an error if necessary.

6. Improve the previous script to congratulate the user if the sum equals the product.

7. Improve the script from question 2. to complain if it does not receive exactly two parameters.

8. Write a script that counts from 3 to 7 and then from 7 to 3, and all this three times, once with a for loop, once with a while loop and once with a until loop. Show the teacher that it works!

9. Write a script that asks for a filename. Verify existance of the file, then verify that you own the file, and whether it is writable. If not, then make it writable.

10. Make a configuration file for the previous script. Put a logging switch in the config file, logging means writing detailed output of everything the script does to a log file in /tmp.

11. Make the case statement in "Wild Animals Helpdesk Advice" case insensitive. Use shopt (with the correct toggled option) for this, but reset the value back to it's original after the end of the case statement. (A solution is available in appendix 1, but try to find it yourself.)

12. If time permits (or if you are waiting for other students to finish this practice), take a look at linux system scripts in /etc/init.d and /etc/rc.d and try to understand them. Where does execution of a script start in /etc/init.d/samba ? There are also some hidden scripts in ~, we will discuss them later.

10.17. Solutions

1. Write a script that receives four parameters, and outputs them in reverse order.

```
echo $4 $3 $2 $1
```

2. Write a script that receives two parameters (two filenames) and outputs whether those files exist.

```
#!/bin/bash

if [ -f $1 ]
then echo $1 exists!
else echo $1 not found!
fi

if [ -f $2 ]
then echo $2 exists!
else echo $2 not found!
fi
```

3. Write a script that counts the number of files ending in .txt in the current directory.

```
#!/bin/bash

ls *.txt > /dev/null 2>&1
if [ $? -ne 0 ]
then echo "There are 0 files ending in .txt"
else
 let i=0
 for file in *.txt
 do
  let i++
 done
 echo "There are $i files ending in .txt"
fi
```

4. Write a script that asks for two numbers, and outputs the sum and product (as shown here).

```
Enter a number: 5
Enter another number: 2

Sum:       5 + 2 = 7
Product:   5 x 2 = 10
```

```
#!/bin/bash

echo -n "Enter a number : "
read n1

echo -n "Enter another number : "
read n2

let sum="$n1+$n2"
let pro="$n1*$n2"

echo -e "Sum\t: $n1 + $n2 = $sum"
echo -e "Product\t: $n1 * $n2 = $pro"
```

5. Improve the previous script to test that the numbers are between 1 and 100, exit with an error if necessary.

```
echo -n "Enter a number between 1 and 100 : "
read n1

if [ $n1 -lt 1 -o $n1 -gt 100 ]
then
       echo Wrong number...
       exit 1
fi
```

6. Improve the previous script to congratulate the user if the sum equals the product.

```
if [ $sum -eq $pro ]
then echo Congratulations $sum == $pro
fi
```

7. Improve the script from question 2. to complain if it does not receive exactly two parameters.

```
if [ $# -ne 2 ]
then
 echo Must get two parameters...
 exit 1
fi
```

8. Write a script that counts from 3 to 7 and then from 7 to 3, and all this three times, once with a for loop, once with a while loop and once with a until loop. Show the teacher that it works!

9. Write a script that asks for a filename. Verify existance of the file, then verify that you own the file, and whether it is writable. If not, then make it writable.

10. Make a configuration file for the previous script. Put a logging switch in the config file, logging means writing detailed output of everything the script does to a log file in /tmp.

11. A script with a case insensitive case statement, using the shopt nocasematch option. The nocasematch option is reset to the value it had before the scripts started.

```
#!/bin/bash
#
# Wild Animals Case Insensitive Helpdesk Advice
#

if shopt -q nocasematch; then
  nocase=yes;
else
  nocase=no;
  shopt -s nocasematch;
fi

echo -n "What animal did you see ? "
read animal

case $animal in
  "lion" | "tiger")
    echo "You better start running fast!"
  ;;
  "cat")
    echo "Let that mouse go..."
  ;;
  "dog")
    echo "Don't worry, give it a cookie."
  ;;
  "chicken" | "goose" | "duck" )
    echo "Eggs for breakfast!"
  ;;
  "liger")
    echo "Approach and say 'Ah you big fluffy kitty.'"
  ;;
  "babelfish")
    echo "Did it fall out your ear ?"
  ;;
  *)
    echo "You discovered an unknown animal, name it!"
  ;;
esac

if [ nocase = yes ] ; then
```

```
        shopt -s nocasematch;
else
        shopt -u nocasematch;
fi
```

12. If time permits (or if you are waiting for other students to finish this practice), take a look at linux system scripts in /etc/init.d and /etc/rc.d and try to understand them. Where does execution of a script start in /etc/init.d/samba ? There are also some hidden scripts in ~, we will discuss them later.

Chapter 11. Process Management

11.1. About processes

A **process** is compiled source code that is currently running on the system. All processes have a **process ID** or **PID**, and a parent process (with a **PPID**). The **child** process is often started by the **parent** process. The **init** process always has process ID 1, and does not have a parent. But init serves as a **foster parent** for **orphaned** processes. When a process stops running, the process dies, when you want a process to die, you **kill** it. Processes that start at system startup and keep running forever are called **daemon** processes. Daemons never die. When a process is killed, but it still shows up on the system, then the process is referred to as **zombie**. You cannot kill zombies, because they are already dead.

11.2. Process ID

Some shell environment variables contain information about processes. The **$$** variable will hold your current process ID (PID), and **$PPID** contains the parent PID. Actually $$ is a shell parameter and not a variable, you cannot assign a value to $$.

```
[paul@RHEL4b ~]$ echo $$ $PPID
4224 4223
[paul@RHEL4b ~]$ bash
[paul@RHEL4b ~]$ echo $$ $PPID
4812 4224
[paul@RHEL4b ~]$ bash
[paul@RHEL4b ~]$ echo $$ $PPID
4830 4812
[paul@RHEL4b ~]$ exit
exit
[paul@RHEL4b ~]$ echo $$ $PPID
4812 4224
[paul@RHEL4b ~]$ exit
exit
[paul@RHEL4b ~]$ echo $$ $PPID
4224 4223
[paul@RHEL4b ~]$
```

11.3. fork

A process starts another process in two fases. First the process creates a **fork** of itself, an identical copy. Then the forked process executes an **exec** to replace the forked process with the target child process.

```
[paul@RHEL4b ~]$ echo $$
4224
[paul@RHEL4b ~]$ bash
[paul@RHEL4b ~]$ echo $$ $PPID
```

```
5310 4224
[paul@RHEL4b ~]$
```

11.4. exec

With the **exec** command, you can execute a process without forking a new process. In the following screenshot i start a Korn shell (ksh) and replace it with a bash shell using the exec command. The PID of the bash shell is the same as the PID of the Korn shell. Exiting the child bash shell will get me back to the parent bash, not to the Korn (which does not exist anymore).

```
[paul@RHEL4b ~]$ echo $$
4224
[paul@RHEL4b ~]$ ksh
$ echo $$ $PPID
5343 4224
$ exec bash
[paul@RHEL4b ~]$ echo $$ $PPID
5343 4224
[paul@RHEL4b ~]$ exit
exit
[paul@RHEL4b ~]$ echo $$
4224
```

11.5. ps

One of the most common tools on Unix to look at processes is **ps**. The following screenshot shows the parent child relationship between three bash processes.

```
[paul@RHEL4b ~]$ echo $$ $PPID
4224 4223
[paul@RHEL4b ~]$ bash
[paul@RHEL4b ~]$ echo $$ $PPID
4866 4224
[paul@RHEL4b ~]$ bash
[paul@RHEL4b ~]$ echo $$ $PPID
4884 4866
[paul@RHEL4b ~]$ ps fx
  PID TTY      STAT   TIME COMMAND
 4223 ?        S      0:01 sshd: paul@pts/0
 4224 pts/0    Ss     0:00  \_ -bash
 4866 pts/0    S      0:00      \_ bash
 4884 pts/0    S      0:00          \_ bash
 4902 pts/0    R+     0:00              \_ ps fx
[paul@RHEL4b ~]$ exit
exit
[paul@RHEL4b ~]$ ps fx
  PID TTY      STAT   TIME COMMAND
 4223 ?        S      0:01 sshd: paul@pts/0
 4224 pts/0    Ss     0:00  \_ -bash
 4866 pts/0    S      0:00      \_ bash
 4903 pts/0    R+     0:00          \_ ps fx
[paul@RHEL4b ~]$ exit
exit
```

```
[paul@RHEL4b ~]$ ps fx
  PID TTY      STAT   TIME COMMAND
 4223 ?        S      0:01 sshd: paul@pts/0
 4224 pts/0    Ss     0:00  \_ -bash
 4904 pts/0    R+     0:00      \_ ps fx
[paul@RHEL4b ~]$
```

On Linux, **ps fax** is often used. On Solaris **ps -ef** is common. Here is a partial output from ps fax.

```
[paul@RHEL4a ~]$ ps fax
PID TTY      STAT   TIME COMMAND
1 ?        S      0:00 init [5]

...

3713 ?        Ss     0:00 /usr/sbin/sshd
5042 ?        Ss     0:00  \_ sshd: paul [priv]
5044 ?        S      0:00      \_ sshd: paul@pts/1
5045 pts/1    Ss     0:00          \_ -bash
5077 pts/1    R+     0:00              \_ ps fax
```

11.6. top

Another popular tool on Linux is **top**. The **top** tool can order processes according to CPU usage or other properties. You can also kill processes from within top. In case of trouble, top is often the first tool to fire up, since it also provides you memory and swap space information.

11.7. priority and nice values

All processes have a certain **priority** and a **nice** value. Higher priority processes will get more CPU time than low priority processes. You can influence this with the **nice** and **renice** commands.

The **top** screenshot below shows four processes, all of then using approximately 25 percent of the CPU. PID 5087 and 5088 are catting the letter x to each other, PID 5090 and 5091 do the same with the letter z.

```
PID  USER PR NI VIRT RES SHR S %CPU %MEM TIME+   COMMAND
5088 paul 25  0 4128 404 348 R 25.6  0.2 0:13.99 cat
5091 paul 25  0 3628 400 348 R 25.6  0.2 0:07.99 cat
5090 paul 15  0 4484 404 348 S 24.6  0.2 0:07.78 cat
5087 paul 15  0 3932 400 348 S 24.3  0.2 0:14.16 cat
```

Since the processes are already running, we need to use the **renice** command to change their nice value. The **nice** command can only be used when starting a process. The screenshot below shows how to make two running processes nice.

```
[paul@RHEL4a ~]$ renice +5 5090
5090: old priority 0, new priority 5
[paul@RHEL4a ~]$ renice +5 5091
5091: old priority 0, new priority 5
```

Two processes (5090 and 5091) are playing nice now, they allow other processes to use more CPU time.

```
PID  USER PR NI VIRT RES SHR S %CPU %MEM  TIME+   COMMAND
5087 paul 15  0 3932 400 348 S 37.3  0.2  1:19.97 cat
5088 paul 25  0 4128 404 348 R 36.6  0.2  1:19.20 cat
5090 paul 21  5 4484 404 348 S 13.7  0.2  1:10.64 cat
5091 paul 29  5 3628 400 348 R 12.7  0.2  1:10.64 cat
```

Be careful when playing with negative nice values (the range is from -20 to 19), the responsiveness of your system can be affected. Luckily only root can issue negative nice values, in other words, you can only lower the priority of your running processes.

11.8. signals (kill)

Running processes can receive signals from each other, or from the users. You can have a list of signals by typing **kill -l**, that is a letter l, not the number 1.

```
[paul@RHEL4a ~]$ kill -l
1) SIGHUP       2) SIGINT       3) SIGQUIT      4) SIGILL
5) SIGTRAP      6) SIGABRT      7) SIGBUS       8) SIGFPE
9) SIGKILL     10) SIGUSR1     11) SIGSEGV     12) SIGUSR2
13) SIGPIPE     14) SIGALRM     15) SIGTERM     17) SIGCHLD
18) SIGCONT     19) SIGSTOP     20) SIGTSTP     21) SIGTTIN
22) SIGTTOU     23) SIGURG      24) SIGXCPU     25) SIGXFSZ
26) SIGVTALRM   27) SIGPROF     28) SIGWINCH    29) SIGIO
30) SIGPWR      31) SIGSYS      34) SIGRTMIN    35) SIGRTMIN+1
36) SIGRTMIN+2  37) SIGRTMIN+3  38) SIGRTMIN+4  39) SIGRTMIN+5
40) SIGRTMIN+6  41) SIGRTMIN+7  42) SIGRTMIN+8  43) SIGRTMIN+9
44) SIGRTMIN+10 45) SIGRTMIN+11 46) SIGRTMIN+12 47) SIGRTMIN+13
48) SIGRTMIN+14 49) SIGRTMIN+15 50) SIGRTMAX-14 51) SIGRTMAX-13
52) SIGRTMAX-12 53) SIGRTMAX-11 54) SIGRTMAX-10 55) SIGRTMAX-9
56) SIGRTMAX-8  57) SIGRTMAX-7  58) SIGRTMAX-6  59) SIGRTMAX-5
60) SIGRTMAX-4  61) SIGRTMAX-3  62) SIGRTMAX-2  63) SIGRTMAX-1
64) SIGRTMAX
[paul@RHEL4a ~]$
```

It is common on Linux to use the first signal SIGHUP (or HUP or 1) to tell a process that it should re-read its configuration file. Thus, the **kill -1 1** command forces the init process to re-read its configuration file. It is up to the developer of the process to decide whether the process can do this running, or whether it needs to stop and start. The **killall** command will also default to sending a signal 15 to the processes.

The SIGTERM (15) is used to ask a process to stop running, normally the process should die. If it refuses to die, then you can issue the **kill -9** command (aka the **sure kill**). The SIGKILL (9) signal is the only one that a developer cannot intercept. The signal goes directly to the kernel, which will stop the running process (without giving it a chance to save data). When using the kill command without specifying a signal, it defaults to SIGTERM (15).

```
[paul@RHEL4a ~]$ ps fax | grep cat
5087 pts/1    S     10:04              \_ cat - pipe1
5088 pts/1    R     10:06              \_ cat
5090 pts/1    SN     4:26              \_ cat - pipe3
5091 pts/1    RN     4:28              \_ cat
5220 pts/1    S+     0:00              \_ grep cat
[paul@RHEL4a ~]$ kill 5087
[1]   Terminated              echo -n x | cat - pipe1 >pipe2
[paul@RHEL4a ~]$
```

11.9. pgrep

Similar to the **ps -C**, you can also use **pgrep** to search for a process by its command name.

```
[paul@RHEL5 ~]$ sleep 1000 &
[1] 32558
[paul@RHEL5 ~]$ pgrep sleep
32558
[paul@RHEL5 ~]$ ps -C sleep
  PID TTY          TIME CMD
32558 pts/3    00:00:00 sleep
[paul@RHEL5 ~]$
```

You can also list the command name of the process with pgrep.

```
paul@laika:~$ pgrep -l sleep
9661 sleep
paul@laika:~$
```

11.10. pkill

You can use the **pkill** command to kill a process by its command name.

```
[paul@RHEL5 ~]$ sleep 1000 &
[1] 30203
[paul@RHEL5 ~]$ pkill sleep
[1]+  Terminated              sleep 1000
[paul@RHEL5 ~]$
```

11.11. jobs

Some processes can be frozen with the **Ctrl-Z** key combination. This sends a SIGSTOP to the process. When doing this in vi, the vi goes to the background, and can be seen with the **jobs** command. Processes started with an ampersand (&) at the end of the command line can also be seen with **jobs**.

```
[paul@RHEL4a ~]$ vi procdemo.txt

[5]+  Stopped                 vim procdemo.txt
[paul@RHEL4a ~]$ jobs
[5]+  Stopped                 vim procdemo.txt
[paul@RHEL4a ~]$ find / > allfiles.txt 2> /dev/null &
[6] 5230
[paul@RHEL4a ~]$ jobs
[5]+  Stopped                 vim procdemo.txt
[6]-  Running                 find / >allfiles.txt 2>/dev/null &
[paul@RHEL4a ~]$
```

An interesting option is **jobs -p** to see the PID of background jobs.

```
[paul@RHEL4b ~]$ sleep 500 &
[1] 4902
[paul@RHEL4b ~]$ sleep 400 &
[2] 4903
[paul@RHEL4b ~]$ jobs -p
4902
4903
[paul@RHEL4b ~]$ ps `jobs -p`
  PID TTY      STAT   TIME COMMAND
 4902 pts/0    S      0:00 sleep 500
 4903 pts/0    S      0:00 sleep 400
[paul@RHEL4b ~]$
```

11.12. fg

Running the **fg** command will bring a background job to the foreground. The number of the background job to bring forward is the parameter of fg.

```
[paul@RHEL5 ~]$ jobs
[1]   Running                 sleep 1000 &
[2]-  Running                 sleep 1000 &
[3]+  Running                 sleep 2000 &
[paul@RHEL5 ~]$ fg 3
sleep 2000
```

11.13. bg

Jobs that are stopped in background can be started in background with **bg**. Below an example of the sleep command (stopped with Ctrl-Z) that is reactivated in background with bg.

```
[paul@RHEL5 ~]$ jobs
[paul@RHEL5 ~]$ sleep 5000 &
[1] 6702
[paul@RHEL5 ~]$ sleep 3000

[2]+  Stopped                 sleep 3000
```

```
[paul@RHEL5 ~]$ jobs
[1]-  Running                 sleep 5000 &
[2]+  Stopped                 sleep 3000
[paul@RHEL5 ~]$ bg 2
[2]+ sleep 3000 &
[paul@RHEL5 ~]$ jobs
[1]-  Running                 sleep 5000 &
[2]+  Running                 sleep 3000 &
[paul@RHEL5 ~]$
```

11.14. Zombiecreator ;-)

This is a procedure to create zombies on Linux.

First create a script that contains one line: sleep 5000. Make it executable (chmod +x). Then start this script and use Ctrl-Z to background the script. Use ps -C sleep to find the PID of the sleep process. Then do a kill -9 of this PID. ps -C sleep now shows a defunct sleep. ps -fax and top will show a zombie sleep process.

11.15. Practice

1. Explain in detail where the numbers come from in the next screenshot. When are the variables replaced by their value ? By which shell ?

```
[paul@RHEL4b ~]$ echo $$ $PPID
4224 4223
[paul@RHEL4b ~]$ bash -c "echo $$ $PPID"
4224 4223
[paul@RHEL4b ~]$ bash -c 'echo $$ $PPID'
5059 4224
[paul@RHEL4b ~]$ bash -c `echo $$ $PPID`
4223: 4224: command not found
```

2. Write a script that echoes its process ID and then sleeps for an hour. Find your script with ps.

3. Read the man page of ps and find your script by command name with ps.

4. Kill your script with the kill command.

5. Run your script again, now use top to display only your script and the init process.

6. Use top to kill your script.

7. Use top, organise all processes by memory usage.

8. Write a script with a 'while true' loop that does some calculation. Copy this script.

9. Start the while script. Start the copy of it in a nice way. Do you see the difference with top ? with ps ?

10. Kill all your running scripts.

11. Start editing the while script, put it in background. Same for the copy script. List your background jobs.

12. Start the sleep script in background. List the background jobs. Activate the copy script to foreground.

11.16. Solutions to the Practice

1. The current bash shell will replace the $$ and $PPID while scanning the line, and before executing the echo command.

```
   [paul@RHEL4b ~]$ echo $$ $PPID
   4224 4223
```

The variables are now double quoted, but the current bash shell will replace $$ and $PPID while scanning the line, and before executing the bach -c command.

```
   [paul@RHEL4b ~]$ bash -c "echo $$ $PPID"
   4224 4223
```

The variables are now single quoted. The current bash shell will not replace the $$ and the $PPID. The bash -c command will be executed before the variables replaced with their value. This latter bash is the one replacing the $$ and $PPID with their value.

```
   [paul@RHEL4b ~]$ bash -c 'echo $$ $PPID'
   5059 4224
```

With backticks the shell will still replace both variable before the embedded echo is executed. The result of this echo is the two process id's. These are given as commands to bash -c. But two numbers are not commands!

```
[paul@RHEL4b ~]$ bash -c `echo $$ $PPID`
4223: 4224: command not found
```

2. The script can look like this.

```
#!/bin/bash

echo My Process ID = $$
sleep 3600
```

3. ps -C sleep

4. kill (followed by the PID)

5. top p 1,6705 (replace 6705 with your PID)

6. when inside top, press k

7. use the greater than and smaller than keys from within top

8. an example of an (almost) endless loop:

```
let i=1;
while [ $i -gt 0 ] ;
do
   let i++;
done
```

9. it will show up as 'bash' in top

Chapter 12. More about Bash

12.1. bash shell environment

It is nice to have all these preset and custom aliases and variables, but where do they all come from ? Bash has a number of startup files that are checked (and executed) whenever bash is invoked. Bash first reads and executes **/etc/profile**. Then bash searches for **.bash_profile**, **.bash_login** and **.profile** in the home directory. Bash will execute the first of these three that it finds. Typically these files will expand your $PATH environment variable.

```
[paul@RHELv4u3 ~]$ cat .bash_profile | grep PATH
PATH=$PATH:$HOME/bin
export PATH
[paul@RHELv4u3 ~]$
```

If this is an interactive shell, then bash will also execute **.bashrc**. In the case of Red Hat, the .bashrc file will source **/etc/bashrc**.

```
[paul@RHELv4u3 ~]$ cat .bashrc
# .bashrc

# User specific aliases and functions

# Source global definitions
if [ -f /etc/bashrc ]; then
. /etc/bashrc
fi
[paul@RHELv4u3 ~]$
```

When you exit the shell, then **~/.bash_logout** is executed.

A similar system exists for the Korn shell with .kshrc and other files. Actually a similar system exists for almost all shells.

12.2. path

The **$PATH** variable is very important, it determines where the shell is looking for commands to execute (unless the command is built-in). The shell will not look in the current directory for commands to execute! (Looking for executables in the current directory provided an easy way to crack DOS computers). If you want the shell to look in the current directory, then add a . to your path.

```
[[paul@RHEL4b ~]$ echo $PATH
/usr/kerberos/bin:/usr/local/bin:/bin:/usr/bin:
[paul@RHEL4b ~]$ PATH=$PATH:.
[paul@RHEL4b ~]$ echo $PATH
/usr/kerberos/bin:/usr/local/bin:/bin:/usr/bin:.
```

```
[paul@RHEL4b ~]$
```

Your path might be different when using su instead of **su -** because the latter will take on the environment of the target user. The root user will have some sbin directories added to the PATH variable.

```
[paul@RHEL3 ~]$ su
Password:
[root@RHEL3 paul]# echo $PATH
/usr/local/bin:/bin:/usr/bin:/usr/X11R6/bin
[root@RHEL3 paul]# exit
[paul@RHEL3 ~]$ su -
Password:
[root@RHEL3 ~]# echo $PATH
/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin:
[root@RHEL3 ~]#
```

12.3. Shell I/O redirection

The shell (and almost every other Linux command) takes input from **stdin** (stream **0**) and sends output to **stdout** (stream **1**) and error messages to **stderr** (stream **2**) . Stdin is usually the keyboard, stdout and stderr are the screen. The shell allows you to redirect these streams.

12.3.1. output redirection

Stdout can be redirected with a **greater than** sign. While scanning the line, the shell will see the > sign and will clear the file.

```
[paul@RHELv4u3 ~]$ echo It is cold today!
It is cold today!
[paul@RHELv4u3 ~]$ echo It is cold today! > winter.txt
[paul@RHELv4u3 ~]$ cat winter.txt
It is cold today!
[paul@RHELv4u3 ~]$
```

Let me repeat myself here: While scanning the line, the shell will see the > sign and will clear the file! This means that even when the command fails, the file will be cleared!

```
[paul@RHELv4u3 ~]$ cat winter.txt
It is cold today!
[paul@RHELv4u3 ~]$ zcho It is cold today! > winter.txt
-bash: zcho: command not found
[paul@RHELv4u3 ~]$ cat winter.txt
[paul@RHELv4u3 ~]$
```

Note that the > notation is in fact the abbreviation of **1>** (**stdout** being referred to as stream **1**.

12.3.2. noclobber

This can be prevented by setting the **noclobber** option.

```
[paul@RHELv4u3 ~]$ cat winter.txt
It is cold today!
[paul@RHELv4u3 ~]$ set -o noclobber
[paul@RHELv4u3 ~]$ echo It is cold today! > winter.txt
-bash: winter.txt: cannot overwrite existing file
[paul@RHELv4u3 ~]$ set +o noclobber
[paul@RHELv4u3 ~]$
```

The noclobber can be overruled with >|.

```
[paul@RHELv4u3 ~]$ set -o noclobber
[paul@RHELv4u3 ~]$ echo It is cold today! > winter.txt
-bash: winter.txt: cannot overwrite existing file
[paul@RHELv4u3 ~]$ echo It is very cold today! >| winter.txt
[paul@RHELv4u3 ~]$ cat winter.txt
It is very cold today!
[paul@RHELv4u3 ~]$
```

12.3.3. append

You can always use >> to append output to a file.

```
[paul@RHELv4u3 ~]$ echo It is cold today! > winter.txt
[paul@RHELv4u3 ~]$ cat winter.txt
It is cold today!
[paul@RHELv4u3 ~]$ echo Where is the summer ? >> winter.txt
[paul@RHELv4u3 ~]$ cat winter.txt
It is cold today!
Where is the summer ?
[paul@RHELv4u3 ~]$
```

12.3.4. error redirection

Redirecting stderr is done with **2**>. This can be very useful to prevent error messages from cluttering your screen. The screenshot below shows redirection of stdout to a file, and stderr to /dev/null. Writing **1>** is the same as >.

```
[paul@RHELv4u3 ~]$ find / > allfiles.txt 2> /dev/null
[paul@RHELv4u3 ~]$
```

To redirect both stdout and stderr to the same file, use **2>&1**.

```
[paul@RHELv4u3 ~]$ find / > allfiles_and_errors.txt 2>&1
[paul@RHELv4u3 ~]$
```

12.3.5. input redirection

Redirecting stdin is done with < (short for 0<).

```
[paul@RHEL4b ~]$ cat < text.txt
one
two
[paul@RHEL4b ~]$ tr 'onetw' 'ONEZZ' < text.txt
ONE
ZZO
[paul@RHEL4b ~]$
```

12.3.6. here document

The here document (sometimes called here-is-document) is a way to append input until a certain sequence (usually EOF) is encountered. The **EOF** marker can be typed literally or can be called with Ctrl-D.

```
[paul@RHEL4b ~]$ cat <<EOF > text.txt
> one
> two
> EOF
[paul@RHEL4b ~]$ cat text.txt
one
two
[paul@RHEL4b ~]$ cat <<brol > text.txt
> brel
> brol
[paul@RHEL4b ~]$ cat text.txt
brel
[paul@RHEL4b ~]$
```

12.4. Confusing I/O redirection

The shell will scan the whole line before applying redirection. The following command line is very readable and is correct.

```
cat winter.txt > snow.txt 2> errors.txt
```

But this one is also correct, but less readable.

```
2> errors.txt cat winter.txt > snow.txt
```

Even this will be understood perfectly by the shell.

```
< winter.txt > snow.txt 2> errors.txt cat
```

So what is the quickest way to clear a file ?

```
>foo
```

12.5. swapping stdout and stderr

When filtering an output stream, e.g. through a pipe (|) you only can filter **stdout**. Say you want to filter out some unimportant error, out of the **stderr** stream. This cannot be done directly, and you need to 'swap' **stdout** and **stderr**. This can be done by using a 4th stream referred to with number 3:

```
3>&1 1>&2 2>&3
```

This Tower Of Hanoi like construction uses a temporary stream 3, to be able to swap stdout (1) and stderr (2). The following is an example of how to filter out all lines in the stderr stream, containing $uninterestingerror.

```
/usr/bin/$somecommand 3>&1 1>&2 2>&3 | grep -v $uninterestingerror ) 3>&1 1>&2 2>&3
```

12.6. Practice: more bash

1. Take a backup copy of /etc/bashrc, /etc/profile, ~/.profile, ~/.bashrc and ~/.bash_profile (put them in ~/profilebackups).

2. Set and export a variable named profwinner in all these scripts, the value is the name of the script (profwinner=etc_bashrc in /etc/bashrc, profwinner=dot_profile in ~/.profile, and so on)

3. Set a unique variable in all these scripts (etc_bashrun=yes in /etc/bashrc, dot_profilerun=yes in ~/.profile, and so on)

4. Log on to a tty and to a gnome-terminal, and verify the values of the variables you set in questions 2 and 3. Which of the scripts were executed ? Which not ? Which was executed last ?

5. Does it matter on which line we set our variables in .bash_profile and .bashrc ?

6. Where is the command history stored ? And what about command history for Korn users ?

7. Define an alias 'dog' for the tac command in one of your profile scripts. Which script did you choose and why ?

Chapter 13. Pipes and Filters

13.1. pipes

One of the most powerful advantages of unix is the use of **pipes**, and the ability of almost any program to be used in a pipe. A pipe takes **stdout** from the previous command and sends it as **stdin** to the next command in the pipe. Pipes can have many commands, and all commands in a pipe can be running simultaneously.

What follows after the introduction to pipes is a number of small unix tools that do one specific task very well. These can be used as building blocks for more complex applications and solutions.

You still remember cat and tac right ?

```
[paul@RHEL4b pipes]$ cat count.txt
one
two
three
four
five
[paul@RHEL4b pipes]$ tac count.txt
five
four
three
two
one
[paul@RHEL4b pipes]$
```

A pipe is represented by a vertical bar **|** in between two commands. Below a very simple pipe.

```
[paul@RHEL4b pipes]$ cat count.txt | tac
five
four
three
two
one
[paul@RHEL4b pipes]$
```

But pipes can be longer, as in this example.

```
[paul@RHEL4b pipes]$ cat count.txt | tac | tac
one
two
three
four
five
[paul@RHEL4b pipes]$
```

Remember that I told you in the beginning of this book that the cat command is actually doing nothing ?

```
[paul@RHEL4b pipes]$ tac count.txt | cat | cat | cat | cat | cat
five
four
three
two
one
[paul@RHEL4b pipes]$
```

13.2. tee

Writing long pipes in unix is fun, but sometimes you might want intermediate results. This is were **tee** comes in handy, tee outputs both to a file and to stdout. So tee is almost the same as cat, except that it has two identical outputs.

```
[paul@RHEL4b pipes]$ tac count.txt | tee temp.txt | tac
one
two
three
four
five
[paul@RHEL4b pipes]$ cat temp.txt
five
four
three
two
one
[paul@RHEL4b pipes]$
```

13.3. grep

Time for the real tools now. With all the uses of **grep** you can probably fill a book. The most common use of grep is to filter results on keywords.

```
[paul@RHEL4b pipes]$ cat tennis.txt
Amelie Mauresmo, Fra
Kim Clijsters, BEL
Justine Henin, Bel
Serena Williams, usa
Venus Williams, USA
[paul@RHEL4b pipes]$ cat tennis.txt | grep Williams
Serena Williams, usa
Venus Williams, USA
[paul@RHEL4b pipes]$
```

You can write this without the cat.

```
[paul@RHEL4b pipes]$ grep Williams tennis.txt
Serena Williams, usa
Venus Williams, USA
[paul@RHEL4b pipes]$
```

One of the most useful options of grep is **grep -i** which filters in a case insensitive way.

```
[paul@RHEL4b pipes]$ grep Bel tennis.txt
Justine Henin, Bel
[paul@RHEL4b pipes]$ grep -i Bel tennis.txt
Kim Clijsters, BEL
Justine Henin, Bel
[paul@RHEL4b pipes]$
```

Another very useful option is **grep -v** which outputs lines not matching the string.

```
[paul@RHEL4b pipes]$ grep -v Fra tennis.txt
Kim Clijsters, BEL
Justine Henin, Bel
Serena Williams, usa
Venus Williams, USA
[paul@RHEL4b pipes]$
```

And of course, both options can be combined.

```
[paul@RHEL4b pipes]$ grep -vi usa tennis.txt
Amelie Mauresmo, Fra
Kim Clijsters, BEL
Justine Henin, Bel
[paul@RHEL4b pipes]$
```

13.4. cut

With **cut** you can select columns from files, depending on a delimiter or a count of bytes. The screenshot below uses cut to filter for the username and userid in the /etc/passwd file. It uses the colon as a delimiter, and select fields 1 and 3.

```
[[paul@RHEL4b pipes]$ cut -d: -f1,3 /etc/passwd | tail -4
Figo:510
Pfaff:511
Harry:516
Hermione:517
[paul@RHEL4b pipes]$
```

When using a space as the delimiter for cut, you have to quote the space.

```
[paul@RHEL4b pipes]$ cut -d" " -f1 tennis.txt
Amelie
Kim
Justine
Serena
Venus
[paul@RHEL4b pipes]$
```

One last example, cutting the second to the seventh character of /etc/passwd.

```
[paul@RHEL4b pipes]$ cut -c2-7 /etc/passwd | tail -4
igo:x:
faff:x
arry:x
ermion
[paul@RHEL4b pipes]$
```

13.5. tr

You can translate characters with **tr**. The screenshot translates all occurences of e to E.

```
[paul@RHEL4b pipes]$ cat tennis.txt
Amelie Mauresmo, Fra
Kim Clijsters, BEL
Justine Henin, Bel
Serena Williams, usa
Venus Williams, USA
[paul@RHEL4b pipes]$ cat tennis.txt | tr 'e' 'E'
AmEliE MaurEsmo, Fra
Kim ClijstErs, BEL
JustinE HEnin, BEl
SErEna Williams, usa
VEnus Williams, USA
[paul@RHEL4b pipes]$
```

Here we set all letters to uppercase by defining two ranges.

```
[paul@RHEL4b pipes]$ cat tennis.txt | tr 'a-z' 'A-Z'
AMELIE MAURESMO, FRA
KIM CLIJSTERS, BEL
JUSTINE HENIN, BEL
SERENA WILLIAMS, USA
VENUS WILLIAMS, USA
[paul@RHEL4b pipes]$
```

Here we translate all newlines to spaces.

```
[paul@RHEL4b pipes]$ cat count.txt
one
two
three
four
five
[paul@RHEL4b pipes]$ cat count.txt | tr '\n' ' '
one two three four five [paul@RHEL4b pipes]$
```

The tr filter can also be used to squeeze multiple occurences of a character to one.

```
[paul@RHEL4b pipes]$ cat spaces.txt
one    two        three
     four   five  six
[paul@RHEL4b pipes]$ cat spaces.txt | tr -s ' '
```

```
one two three
 four five six
[paul@RHEL4b pipes]$
```

You can also use tr to 'encrypt' texts with rot13.

```
[paul@RHEL4b pipes]$ cat count.txt | tr 'a-z' 'nopqrstuvwxyzabcdefghijklm'
bar
gjb
guerr
sbhe
svir
[paul@RHEL4b pipes]$ cat count.txt | tr 'a-z' 'n-za-m'
bar
gjb
guerr
sbhe
svir
[paul@RHEL4b pipes]$
```

13.6. wc

Counting words, lines and characters is easy with **wc**.

```
[paul@RHEL4b pipes]$ wc tennis.txt
  5  15 100 tennis.txt
[paul@RHEL4b pipes]$ wc -l tennis.txt
5 tennis.txt
[paul@RHEL4b pipes]$ wc -w tennis.txt
15 tennis.txt
[paul@RHEL4b pipes]$ wc -c tennis.txt
100 tennis.txt
[paul@RHEL4b pipes]$
```

How many users are logged on to this system ?

```
[paul@RHEL4b pipes]$ who
root     tty1         Jul 25 10:50
paul     pts/0        Jul 25 09:29 (laika)
Harry    pts/1        Jul 25 12:26 (barry)
paul     pts/2        Jul 25 12:26 (pasha)
[paul@RHEL4b pipes]$ who | wc -l
4
[paul@RHEL4b pipes]$
```

13.7. sort

Sorting is always useful. The **sort** filter has a lot of options. How about a sorted list of logged on users.

```
[paul@RHEL4b pipes]$ who | cut -d' ' -f1 | sort
```

```
Harry
paul
paul
root
[paul@RHEL4b pipes]$
```

Sorting on column 1 or column 2.

```
[paul@RHEL4b pipes]$ sort -k1 country.txt
Belgium, Brussels, 10
France, Paris, 60
Germany, Berlin, 100
Iran, Teheran, 70
Italy, Rome, 50
[paul@RHEL4b pipes]$ sort -k2 country.txt
Germany, Berlin, 100
Belgium, Brussels, 10
France, Paris, 60
Italy, Rome, 50
Iran, Teheran, 70
[paul@RHEL4b pipes]$
```

The screenshot below shows the difference between an alfabetical sort and a numerical sort (both on the third column).

```
[paul@RHEL4b pipes]$ sort -k3 country.txt
Belgium, Brussels, 10
Germany, Berlin, 100
Italy, Rome, 50
France, Paris, 60
Iran, Teheran, 70
[paul@RHEL4b pipes]$ sort -n -k3 country.txt
Belgium, Brussels, 10
Italy, Rome, 50
France, Paris, 60
Iran, Teheran, 70
Germany, Berlin, 100
[paul@RHEL4b pipes]$
```

13.8. uniq

With **uniq** you can remove duplicates from a sorted list. Here's a sorted list of logged on users, first with and then without duplicates.

```
[paul@RHEL4b pipes]$ who | cut -d' ' -f1 | sort
Harry
paul
paul
root
[paul@RHEL4b pipes]$ who | cut -d' ' -f1 | sort | uniq
Harry
paul
root
[paul@RHEL4b pipes]$
```

13.9. find

The **find** tool is used very often in linux. Find is useful at the start of a pipe, to search for files. Here are some examples. In real life, you will want to add 2>/dev/null to the command lines to avoid cluttering your screen with error messages.

Find all files in /etc and put the list in etcfiles.txt

```
find /etc > etcfiles.txt
```

Find all files of the entire system and put the list in allfiles.txt

```
find / > allfiles.txt
```

Find files that end in .conf in the current directory (and all subdirs).

```
find . -name "*.conf"
```

Find files of type file (so not directory or pipe...) that end in .conf.

```
find . -type f -name "*.conf"
```

Find files of type directory that end in .bak.

```
find /data -type d -name "*.bak"
```

Find files that are newer than file44.txt

```
find . -newer fil44.txt
```

Find can also execute another command on every file found. This example will look for *.odf files and copy them to /backup/.

```
find "/data/*.odf" -exec cp {} /backup/ \;
```

Find can also execute, after your confirmation, another command on every file found. This example will remove *.odf files if you approve of it for every file found.

```
find "/data/*.odf" -ok rm {} \;
```

The find tool can do much more, see the man page.

13.10. locate

The **locate** tool is very different from **find** in that it uses an index to locate files. This is a lot faster than traversing all the directories, but it also means that it is always outdated. If the index does not exist yet, then you have to create it (as root on Red Hat Enterprise Linux) with the **updatedb** command.

```
[paul@RHEL4b ~]$ locate Samba
warning: locate: could not open database: /var/lib/slocate/slocate.db:...
warning: You need to run the 'updatedb' command (as root) to create th...
Please have a look at /etc/updatedb.conf to enable the daily cron job.
[paul@RHEL4b ~]$ updatedb
fatal error: updatedb: You are not authorized to create a default sloc...
```

```
[paul@RHEL4b ~]$ su -
Password:
[root@RHEL4b ~]# updatedb
[root@RHEL4b ~]#
```

13.11. diff

To compare two files line by line, you can use **diff**. To ignore blanks, use **diff -b**, and to ignore case, use **diff -i**.

In this examples diff tells you 2c2 the second line in file one was changed with the second line in file two.

```
[paul@RHEL4b test]$ cat > count.txt
one
two
three
four
[paul@RHEL4b test]$ cat > count2.txt
one
Two
three
four
[paul@RHEL4b test]$ diff count.txt count2.txt
2c2
< two
---
> Two
[paul@RHEL4b test]$
```

Another example of diff. The second file now has one more line than the first file. After line 2, a line was added as line 3 (2a3) to the second file.

```
[paul@RHEL4b test]$ cat > count.txt
one
two
four
[paul@RHEL4b test]$ cat > count2.txt
one
two
three
four
[paul@RHEL4b test]$ diff count.txt count2.txt
2a3
> three
[paul@RHEL4b test]$
```

13.12. comm

You can use **comm** to quickly compare two sorted files. By default comm will output three columns. In this example, Abba, Cure and Queen are in both lists, Bowie and Sweet are only in the first file, Turner is only in the second.

```
[paul@RHEL4b test]$ cat > list1.txt
Abba
Bowie
Cure
Queen
Sweet
[paul@RHEL4b test]$ cat > list2.txt
Abba
Cure
Queen
Turner
[paul@RHEL4b test]$ comm list1.txt list2.txt
                Abba
Bowie
                Cure
                Queen
Sweet
        Turner
[paul@RHEL4b test]$
```

13.13. compress

Users never have enough space, so compression comes in handy. The **compress** command can make files take up less space. You can get the original back with **uncompress**. In the backup chapter we will also discuss gzip, gunzip, bzip2 and bunzip2.

```
[paul@RHEL4b test]$ ls -lh
total 19M
-rw-rw-r--  1 paul paul 19M Jul 26 04:21 allfiles.txt
[paul@RHEL4b test]$ compress allfiles.txt
[paul@RHEL4b test]$ ls -lh
total 3.2M
-rw-rw-r--  1 paul paul 3.2M Jul 26 04:21 allfiles.txt.Z
[paul@RHEL4b test]$ uncompress allfiles.txt
[paul@RHEL4b test]$ ls -lh
total 19M
-rw-rw-r--  1 paul paul 19M Jul 26 04:21 allfiles.txt
[paul@RHEL4b test]$
```

13.14. od

European humans like to work with ascii characters, but computers store files in bytes. The example below creates a simple file, and then uses **od** to show the contents of the file in hexadecimal bytes, in octal bytes and in ascii (or backslashed) characters.

```
paul@laika:~/test$ cat > text.txt
abcdefg
1234567
paul@laika:~/test$ od -t x1 text.txt
0000000 61 62 63 64 65 66 67 0a 31 32 33 34 35 36 37 0a
0000020
paul@laika:~/test$ od -b text.txt
```

```
0000000 141 142 143 144 145 146 147 012 061 062 063 064 065 066 067 012
0000020
paul@laika:~/test$ od -c text.txt
0000000   a   b   c   d   e   f   g  \n   1   2   3   4   5   6   7  \n
0000020
paul@laika:~/test$
```

13.15. other tools and filters

You might want to look at expand, unexpand, pr, nl, fmt, paste, join, sed, awk, ...

13.16. Practice tools and filters

1. Explain the difference between these two commands. This question is very important. If you don't know the answer, then look back at the bash chapters.

```
find . -name "*.txt"
```

```
find . -name *.txt
```

2. Explain the difference between these two statements. Will they both work when there are 200 .odf files in /data/ ? How about when there are 2 million .odf files ?

```
find /data -name "*.odf" > data_odf.txt
```

```
find /data/*.odf > data_odf.txt
```

3. Write a find command that finds all files created after january 30th this year.

4. Write a find command that finds all *.odf files created in september last year.

5. Put a sorted list of all bash users in bashusers.txt.

6. Put a sorted list of all bash users, with their username, userid and home directory in bashusers.info.

7. Make a list of all non-bash and non-korn users.

8. Make a list of all files (not directories) in /etc/ that contain the string smb, nmb or samba.

9. Look at the output of /sbin/ifconfig. Make an ipconfig command that shows only the nic name (eth0), the ip address and the subnet mask.

10. Make a command abc that removes all non-letters from a file (and replaces them with spaces).

11. Count the number of *.conf files in /etc and all its subdirs.

12. Two commands that do the same thing: copy *.odf files to /backup/ . What would be a reason to replace the first command with the second ? Again, this is an important question.

```
cp -r /data/*.odf /backup/
```

```
find /data -name "*.odf" -exec cp {} /backup/ \;
```

13. Create a file called loctest.txt. Can you find this file with locate ? Why not ? How do you make locate find this file ?

14. Create a file named text.txt that contains this sentence: The zun is shining today. Create a file DICT that contains the words "is shining sun the today", one word on each line. The first file is a text, the second file is a dictionary. Now create a spell checker that uses those two files and outputs the misspelled words (in this case that would be 'zun').

15. Use find and -exec to rename all .htm files to .html.

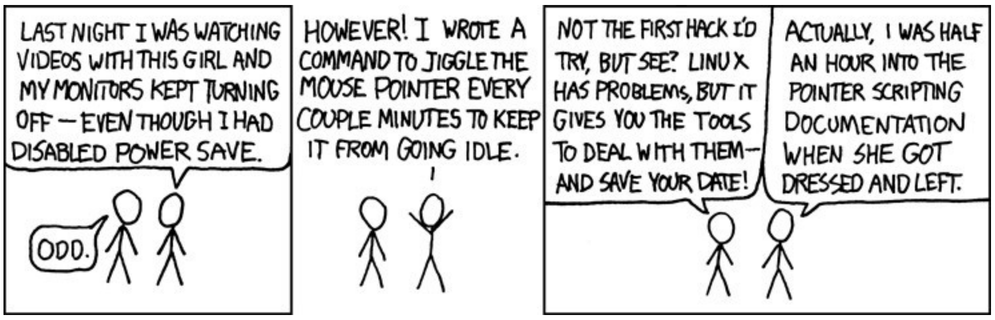
16. Find the hexadecimal byte value for ascii characters : " 'space' 'tab' A and a .

17. List all files in the current directory of size between 10 and 20 bytes.

18. List all files in your home directory that have more than one hard link (hint: use the find tool).

Always take time to properly **document** every script that you write!

Image copied from **xkcd.com**.


LAST NIGHT I WAS WATCHING VIDEOS WITH THIS GIRL AND MY MONITORS KEPT TURNING OFF — EVEN THOUGH I HAD DISABLED POWER SAVE.
ODD.
HOWEVER! I WROTE A COMMAND TO JIGGLE THE MOUSE POINTER EVERY COUPLE MINUTES TO KEEP IT FROM GOING IDLE.
!
NOT THE FIRST HACK I'D TRY, BUT SEE? LINUX HAS PROBLEMS, BUT IT GIVES YOU THE TOOLS TO DEAL WITH THEM— AND SAVE YOUR DATE!
ACTUALLY, I WAS HALF AN HOUR INTO THE POINTER SCRIPTING DOCUMENTATION WHEN SHE GOT DRESSED AND LEFT.

13.17. Solutions: tools and filters

1. The shell will not touch the *.txt because it is between double quotes. The find tool will look in the current directory for all files ending in .txt.

```
find . -name "*.txt"
```

The shell will expand the *.txt to all files in the current directory that end in .txt. Then find will give you a syntax error.

```
find . -name *.txt
```

14. The one line spell checker.

```
[paul]$ echo "The zun is shining today" > text.txt
[paul]$ cat > DICT
is
shining
sun
the
today
[paul]$ cat text.txt| tr 'A-Z ' 'a-z\n' |sort|uniq|comm -2 -3 - DICT
zun
[paul]$
```

18. Use find to look in your home directory(~) for regular files(-type f) that do not(!) have one hard link(-links 1).

```
find ~ ! -links 1 -type f
```

Chapter 14. Disk Management

14.1. Hard disk devices

14.1.1. Terminology

Data is commonly stored on magnetic or optical **disk platters**. The platters are rotated (at high speeds). Data is read by **heads**, which are very close to the surface of the platter, without touching it! The heads are mounted on an arm (sometimes called a comb).

Data is written in concentric circles or **tracks**. Track zero is (usually ?) on the inside. The time it takes to position the head over a certain track is called the **seek time**. Often the platters are stacked on top of each other, hence the set of tracks accessible at a certain position of the comb forms a **cylinder**. Tracks are divided into 512 byte **sectors**, with more unused space (**gap**) between the sectors on the outside of the platter.

When you break down the advertised **access time** of a hard drive, you will notice that most of that time is taken by movement of the heads (about 65%) and **rotational latency** (about 30%).

Random access hard disk devices have an abstraction layer called **block device** to enable formatting in fixed-size (usually 512 bytes) blocks. Blocks can be accessed independent of access to other blocks. You can recognize a block device by the letter b as first character of ls -l.

```
[root@RHEL4b ~]# ls -l /dev/sda*
brw-rw----  1 root disk 8, 0 Aug  4 22:55 /dev/sda
brw-rw----  1 root disk 8, 1 Aug  4 22:55 /dev/sda1
brw-rw----  1 root disk 8, 2 Aug  4 22:55 /dev/sda2
[root@RHEL4b ~]#
```

14.1.2. IDE or SCSI

Actually, the title should be **ATA** or **SCSI**, since IDE is an ATA-compatible device. Most desktops use ATA devices. ATA allows two devices per bus, one **master** and one **slave**. Unless your controller and devices support **cable select**, you have to set this manually with jumpers. With the introduction of **SATA** (Serial ATA), the original ATA was renamed to **Parallel ATA**. Optical drives often use **atapi**, which is an ATA interface using the SCSI communication protocol.

When using the **Small Computer System Interface**, each device gets a unique **SCSI ID**. The SCSI controller also needs a SCSI ID, do not use this ID for a SCSI-attached device. Older 8-bit SCSI is now called **narrow**, whereas 16-bit is **wide**. When the bus speeds was doubled to 10Mhz, this was known as **fast SCSI**. Doubling to 20Mhz made it **ultra SCSI**. Take a look at http://en.wikipedia.org/wiki/SCSI for more SCSI-standards.

14.1.3. Device Naming

All ATA drives on your system will start with **/dev/hd** followed by a unit letter. The master hdd on the first ATA controller is /dev/hda, the slave is /dev/hdb. For the second controller, the names of the devices are /dev/hdc and /dev/hdd.

Table 14.1. IDE device naming

Controller	Connection	Device Name
IDE0	master	/dev/hda
	slave	/dev/hdb
IDE1	master	/dev/hdc
	slave	/dev/hdd

SCSI drives follow a similar scheme, but all start with **/dev/sd**. When you run out of letters (after /dev/sdz), you can continue with /dev/sdaa and /dev/sdab and so on. (We will see later on that LVM volumes are commonly seen as /dev/md0, /dev/md1 etc)

Below a **sample** of how SCSI devices on a linux can be named. Adding a SCSI disk or RAID controller with a lower SCSI address will change the naming scheme (shifting the higher SCSI addresses one letter further in the alphabet).

Table 14.2. SCSI device naming

Device	SCSI ID	Device Name
Disk 0	0	/dev/sda
Disk 1	1	/dev/sdb
RAID Controller 0	5	/dev/sdc
RAID Controller 1	6	/dev/sdd

To get more information about the naming of devices and their major and minor number, visit http://www.lanana.org/docs/device-list/devices.txt .

14.1.4. Erasing a hard disk

Before selling your old hard disk on the internet, it might be a good idea to really erase it. By simply repartitioning or even after a new mkfs command, some people will still be able to read most of the data on the disk. Although technically the **badblocks** tool is meant to look for bad blocks, you can use it to erase a disk. Since this is really writing to every sector of the disk, it can take a long time!

```
root@RHELv4u2:~# badblocks -ws /dev/sdb
```

```
Testing with pattern 0xaa: done
Reading and comparing: done
Testing with pattern 0x55: done
Reading and comparing: done
Testing with pattern 0xff: done
Reading and comparing: done
Testing with pattern 0x00: done
Reading and comparing: done
```

14.1.5. fdisk

You can start by using **fdisk** to find out what kind of disks are seen by the kernel. Below the result on Debian, with two **ATA-IDE disks** present.

```
root@barry:~# fdisk -l | grep Disk
Disk /dev/hda: 60.0 GB, 60022480896 bytes
Disk /dev/hdb: 81.9 GB, 81964302336 bytes
```

And here an example of **SATA disks** on a laptop with Ubuntu. SATA hard disks are presented to you with the SCSI /dev/sdx notation.

```
root@laika:~# fdisk -l | grep Disk
Disk /dev/sda: 100.0 GB, 100030242816 bytes
Disk /dev/sdb: 100.0 GB, 100030242816 bytes
```

And last but not least, an overview of disks on a RHEL4u3 server with two real 72GB **SCSI disks**. This server is attached to a NAS with four **NAS disks** of half a terabyte. On the NAS disks, four LVM software RAID devices are configured.

```
[root@tsvtl1 ~]# fdisk -l | grep Disk
Disk /dev/sda: 73.4 GB, 73407488000 bytes
Disk /dev/sdb: 73.4 GB, 73407488000 bytes
Disk /dev/sdc: 499.0 GB, 499036192768 bytes
Disk /dev/sdd: 499.0 GB, 499036192768 bytes
Disk /dev/sde: 499.0 GB, 499036192768 bytes
Disk /dev/sdf: 499.0 GB, 499036192768 bytes
Disk /dev/md0: 271 MB, 271319040 bytes
Disk /dev/md2: 21.4 GB, 21476081664 bytes
Disk /dev/md3: 21.4 GB, 21467889664 bytes
Disk /dev/md1: 21.4 GB, 21476081664 bytes
```

You can also use fdisk to obtain information about one specific hard disk device.

```
[root@rhel4 ~]# fdisk -l /dev/sda

Disk /dev/sda: 12.8 GB, 12884901888 bytes
255 heads, 63 sectors/track, 1566 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

   Device Boot      Start         End      Blocks   Id  System
/dev/sda1   *           1          13      104391   83  Linux
/dev/sda2              14        1566    12474472+  8e  Linux LVM
```

Later we will use fdisk to do dangerous stuff like creating and deleting partitions.

14.1.6. hdparm

To obtain (or set) information and parameters about an ATA (or SATA) hard disk device, you can use **hdparm**. The -i and -I options will give you even more information about the physical properties of the device.

```
root@laika:~# hdparm /dev/sdb

/dev/sdb:
 IO_support   =  0 (default 16-bit)
 readonly     =  0 (off)
 readahead    = 256 (on)
 geometry     = 12161/255/63, sectors = 195371568, start = 0
```

14.1.7. dmesg

Kernel boot messages can be seen after boot with **dmesg**. Since hard disk devices are detected by the kernel during boot, you can also use dmesg to find information.

```
root@barry:~# dmesg | grep "[hs]d[a-z]"
Kernel command line: root=/dev/hda1 ro
    ide0: BM-DMA at 0xfc00-0xfc07, BIOS settings: hda:DMA, hdb:DMA
    ide1: BM-DMA at 0xfc08-0xfc0f, BIOS settings: hdc:DMA, hdd:DMA
hda: ST360021A, ATA DISK drive
hdb: Maxtor 6Y080L0, ATA DISK drive
hdc: SONY DVD RW DRU-510A, ATAPI CD/DVD-ROM drive
hdd: SONY DVD RW DRU-810A, ATAPI CD/DVD-ROM drive
hda: max request size: 128KiB
hda: 117231408 sectors (60022 MB) w/2048KiB Cache, CHS=65535/16/63, UDMA
 hda: hda1 hda2
hdb: max request size: 128KiB
hdb: 160086528 sectors (81964 MB) w/2048KiB Cache, CHS=65535/16/63, UDMA
 hdb: hdb1 hdb2
hdc: ATAPI 32X DVD-ROM DVD-R CD-R/RW drive, 8192kB Cache, UDMA(33)
hdd: ATAPI 40X DVD-ROM DVD-R CD-R/RW drive, 2048kB Cache, UDMA(33)
...
```

14.1.8. /proc/scsi/scsi

You can also look at the contents of **/proc/scsi/scsi**.

```
root@shaka:~# cat /proc/scsi/scsi
Attached devices:
Host: scsi0 Channel: 00 Id: 00 Lun: 00
  Vendor: Adaptec  Model: RAID5            Rev: V1.0
  Type:   Direct-Access                    ANSI SCSI revision: 02
Host: scsi1 Channel: 00 Id: 00 Lun: 00
```

```
  Vendor: SEAGATE  Model: ST336605FSUN36G  Rev: 0438
  Type:   Direct-Access                    ANSI SCSI revision: 03
  root@shaka:~#
```

14.1.9. scsi_info

You can also use **scsi_info**.

```
root@shaka:~# scsi_info /dev/sdb
SCSI_ID="0,0,0"
HOST="1"
MODEL="SEAGATE ST336605FSUN36G"
FW_REV="0438"
root@shaka:~#
```

14.1.10. lsscsi

And even **lsscsi** if it is installed.

```
root@shaka:~# lsscsi
[0:0:0:0]    disk    Adaptec  RAID5            V1.0  /dev/sda
[1:0:0:0]    disk    SEAGATE  ST336605FSUN36G  0438  /dev/sdb
root@shaka:~#
```

14.1.11. Practice hard disk devices

1. Use dmesg to make a list of hard disk devices (ide,ata,sata,scsi) detected at bootup.

2. Use fdisk to find the total size of all hard disk devices on your system.

3. Stop a virtual machine, add a virtual 10 gigabyte SCSI hard disk and a virtual 100 megabyte SCSI hard disk.

4. Use dmesg and fdisk (with grep) to display some information about the new disks.

5. Use badblocks to completely erase the 100 mb hard disk.

6. Look at /proc/scsi/scsi.

14.2. Partitions

14.2.1. About Partitions

Linux requires you to create one or more **partitions** aka **slices**. *Please don't break your head on the difference between a partition and a slice. Different tools have* *different interpretations of which is which.* Although partitions reside on the same hard disk device, you can (almost) see them as independent of each other.

A partition's **geometry** and size is usually defined by a starting and ending cylinder (sometimes by head or even sector). Partitions can be of type **primary** (maximum four), **extended** (maximum one) or **logical** (contained within the extended partition). Each partition has a **type field** that contains a code. This determines the computers operating system or the partitions file system.

14.2.2. Partition naming

We saw before that hard disk devices are named /dev/hdx or /dev/sdx with x depending on the hardware configuration. Next is the partition number, starting the count at 1. Hence the four (possible) primary partitions are numbered 1 to 4. Logical partition counting always starts at 5. Thus /dev/hda2 is the second partition on the first ATA hard disk device, and /dev/hdb5 is the first logical partition on the second ATA hard disk device. Same for SCSI, /dev/sdb3 is the third partition on the second SCSI disk.

Table 14.3. Partition naming

Partition Type	naming
Primary (max 4)	1-4
Extended (max 1)	1-4
Logical	5-

14.2.3. fdisk -l

In the **fdisk -l** example below you can see that two partitions exist on /dev/sdb2. The first partition spans 31 cylinders and contains a Linux swap partition. The second partition is much bigger.

```
root@laika:~# fdisk -l /dev/sdb

Disk /dev/sdb: 100.0 GB, 100030242816 bytes
255 heads, 63 sectors/track, 12161 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

   Device Boot    Start       End     Blocks   Id  System
/dev/sdb1             1        31     248976   82  Linux swap / Solaris
/dev/sdb2            32     12161   97434225   83  Linux
root@laika:~#
```

14.2.4. other tools

You might be interested in more GUI-oriented alternatives to fdisk and parted like cfdisk, sfdisk and gparted.

14.2.5. Partitioning new disks

In the example below, we bought a new disk for our system. After the new hardware is properly attached, you can use **fdisk** and **parted** to create the necessary partition(s). This example uses fdisk, but there is nothing wrong with using parted.

First, we check with **fdisk -l** whether Linux can see the new disk. Yes it does, the new disk is seen as /dev/sdb, but it does not have any partitions yet.

```
root@RHELv4u2:~# fdisk -l

Disk /dev/sda: 12.8 GB, 12884901888 bytes
255 heads, 63 sectors/track, 1566 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

Device Boot      Start      End      Blocks   Id  System
/dev/sda1   *       1        13      104391   83  Linux
/dev/sda2          14      1566    12474472+  8e  Linux LVM

Disk /dev/sdb: 1073 MB, 1073741824 bytes
255 heads, 63 sectors/track, 130 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

Disk /dev/sdb doesn't contain a valid partition table
```

Then we create a partition with fdisk on /dev/sdb. First we start the fdisk tool with /dev/sdb as argument. Be very very careful not to partition the wrong disk!!

```
root@RHELv4u2:~# fdisk /dev/sdb
Device contains neither a valid DOS partition table, nor Sun, SGI...
Building a new DOS disklabel. Changes will remain in memory only,
until you decide to write them. After that, of course, the previous
content won't be recoverable.

Warning: invalid flag 0x0000 of partition table 4 will be corrected...
```

Inside the fdisk tool, we can issue the **p** command to see the current disks partition table.

```
Command (m for help): p

Disk /dev/sdb: 1073 MB, 1073741824 bytes
255 heads, 63 sectors/track, 130 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

Device Boot      Start         End      Blocks   Id  System
```

No partitions exist yet, so we issue **n** to create a new partition. We choose p for primary, 1 for the partition number, 1 for the start cylinder and 14 for the end cylinder.

```
Command (m for help): n
Command action
e   extended
```

```
p   primary partition (1-4)
p
Partition number (1-4): 1
First cylinder (1-130, default 1):
Using default value 1
Last cylinder or +size or +sizeM or +sizeK (1-130, default 130): 14
```

We can now issue p again to verify our changes, but they are not yet written to disk. This means we can still cancel this operation! But it looks good, so we use **w** to write the changes to disk, and then quit the fdisk tool.

```
Command (m for help): p

Disk /dev/sdb: 1073 MB, 1073741824 bytes
255 heads, 63 sectors/track, 130 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

Device Boot      Start         End      Blocks   Id  System
/dev/sdb1               1          14      112423+  83  Linux

Command (m for help): w
The partition table has been altered!

Calling ioctl() to re-read partition table.
Syncing disks.
root@RHELv4u2:~#
```

Let's verify again with **fdisk -l** to make sure reality fits our dreams. Indeed, the screenshot below now shows a partition on /dev/sdb.

```
root@RHELv4u2:~# fdisk -l

Disk /dev/sda: 12.8 GB, 12884901888 bytes
255 heads, 63 sectors/track, 1566 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

Device Boot      Start        End      Blocks   Id  System
/dev/sda1   *        1         13      104391   83  Linux
/dev/sda2           14       1566    12474472+  8e  Linux LVM

Disk /dev/sdb: 1073 MB, 1073741824 bytes
255 heads, 63 sectors/track, 130 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

Device Boot      Start         End      Blocks   Id  System
/dev/sdb1               1          14      112423+  83  Linux
root@RHELv4u2:~#
```

14.2.6. Master Boot Record

The partition table information (primary and extended partitions) is written in the **Master Boot Record** or **MBR**. You can use **dd** to copy the mbr to a file.

This example copies the master boot record from the first SCSI hard disk.

```
dd if=/dev/sda of=/SCSIdisk.mbr bs=512 count=1
```

The same tool can also be used to wipe out all information about partitions on a disk. This example writes zeroes over the master boot record.

```
dd if=/dev/zero of=/dev/sda bs=512 count=1
```

14.2.7. partprobe

Don't forget that after restoring a Master Boot Record with dd, that you need to force the kernel to reread the partition table with **partprobe**. After running partprobe, the partitions can be used again.

```
[root@RHEL5 ~]# partprobe
[root@RHEL5 ~]#
```

14.2.8. logical drives

The partition table does not contain information about logical drives. So the dd backup of the mbr only works for primary and extended partitions. To backup the partition table inlcuding the logical drives, you can use **sfdisk**. The following command will copy the mbs and the logical drive information from /dev/sda to /dev/sdb.

```
sfdisk -d /dev/sda | sfdisk /dev/sdb
```

14.2.9. Practice Partitions

1. Use fdisk and df to display existing partitions and sizes. Compare the output of the two commands.

2. Create a 50MB primary partition on a small disk.

3. Create a 200MB primary partition and two 100MB logical drives on a big disk.

4. Use df and fdisk -l to verify your work.

5. Take a backup of the partition table containing your 200MB primary and 100MB logical drives. Destroy the partitions with fdisk. Then restore your backup.

14.3. File Systems

14.3.1. About file systems

After you are finished partitioning the hard disk, you can put a **file system** on each partition. A file system is a way of organizing files on your partition. Besides file-based storage, file systems usually include **directories** and **access control**, and contain meta information about files like access times, modification times and file ownership.

The properties (length, character set, ...) of filenames are determined by the file system you choose. Directories are usually implemented as files, you will have to learn how this is implemented! Access control in file systems is tracked by user ownership (and group owner- and membership) in combination with one or more access control lists.

14.3.2. Common file systems

14.3.2.1. ext2 and ext3

Once the most common Linux file systems is the **ext2** (the second extended) file system. A disadvantage is that file system checks on ext2 can take a long time. You will see that ext2 is being replaced by **ext3** on most Linux machines. They are essentially the same, except for the **journaling** which is only present in ext3.

Journaling means that changes are first written to a journal on the disk. The journal is flushed regularly, writing the changes in the file system. Journaling keeps the file system in a consistent state, so you don't need a file system check after an unclean shutdown or power failure.

You can create these file systems with the **/sbin/mkfs** or **/sbin/mke2fs** commands. Use **mke2fs -j** to create an ext3 file system. You can convert an ext2 to ext3 with **tune2fs -j**. You can mount an ext3 file system as ext2, but then you lose the journaling. Do not forget to run **mkinitrd** if you are booting from this device.

14.3.2.2. vfat

The **vfat** file system exists in a couple of forms : FAT12 for floppy disks, **FAT16** on DOS, and **FAT32** for larger disks. The Linux VFAT implementation supports all of these, but vfat lacks a lot of features like security and links. FAT disks can be read by every operating system, and are used a lot for digital cameras, USB sticks and to exchange data between different OS'ses on a home user's computer.

14.3.2.3. ISO 9660

ISO 9660 is the standard format for CD-ROM's. Chances are you will encounter this file system also on your harddisk in the form of images of CD-ROM's (often with the .ISO extension). The ISO 9660 standard limits filenames to the 8.3 format. The Unix world didn't like this, and thus added the **Rock Ridge** extensions, which allows for filenames up to 255 characters and Unix-style file-modes, ownership and symbolic links. Another extensions to ISO 9660 is **Joliet**, which adds 64 unicode characters to the filename. The **El Torito** standard extends ISO 9660 to be able to boot from CD-ROM's.

14.3.2.4. UDF

Most optical media today (including CD's and DVD's) use **UDF**, the Universal Disk Format.

14.3.2.5. swap

All things considered, swap is not a file system. But to use a partition as a **swap partition** it must be formatted as swap space.

14.3.2.6. others...

You might encounter **reiserfs** on Linux systems, but it is not common on Red Hat. Maybe you will see a **zfs**, or one of the dozen other file systems available.

14.3.3. Putting a file system on a partition

We now have a fresh partition. The system binaries to make file systems can be found with ls.

```
[root@RHEL4b ~]# ls -lS /sbin/mk*
-rwxr-xr-x  3 root root 34832 Apr 24  2006 /sbin/mke2fs
-rwxr-xr-x  3 root root 34832 Apr 24  2006 /sbin/mkfs.ext2
-rwxr-xr-x  3 root root 34832 Apr 24  2006 /sbin/mkfs.ext3
-rwxr-xr-x  3 root root 28484 Oct 13  2004 /sbin/mkdosfs
-rwxr-xr-x  3 root root 28484 Oct 13  2004 /sbin/mkfs.msdos
-rwxr-xr-x  3 root root 28484 Oct 13  2004 /sbin/mkfs.vfat
-rwxr-xr-x  1 root root 20313 Apr 10  2006 /sbin/mkinitrd
-rwxr-x---  1 root root 15444 Oct  5  2004 /sbin/mkzonedb
-rwxr-xr-x  1 root root 15300 May 24  2006 /sbin/mkfs.cramfs
-rwxr-xr-x  1 root root 13036 May 24  2006 /sbin/mkswap
-rwxr-xr-x  1 root root  6912 May 24  2006 /sbin/mkfs
-rwxr-xr-x  1 root root  5905 Aug  3  2004 /sbin/mkbootdisk
[root@RHEL4b ~]#
```

It is time for you to read the manual pages of **mkfs** and **mke2fs**. In the example below, you see the creation of an **ext2 file system** on /dev/sdb1. In real life, you might want to use options like -m0 and -j.

```
root@RHELv4u2:~# mke2fs /dev/sdb1
mke2fs 1.35 (28-Feb-2004)
Filesystem label=
OS type: Linux
Block size=1024 (log=0)
Fragment size=1024 (log=0)
28112 inodes, 112420 blocks
5621 blocks (5.00%) reserved for the super user
```

```
First data block=1
Maximum filesystem blocks=67371008
14 block groups
8192 blocks per group, 8192 fragments per group
2008 inodes per group
Superblock backups stored on blocks:
8193, 24577, 40961, 57345, 73729

Writing inode tables: done
Writing superblocks and filesystem accounting information: done

This filesystem will be automatically checked every 37 mounts or
180 days, whichever comes first.  Use tune2fs -c or -i to override.
```

14.3.4. Tuning a file system

You can use **tune2fs** to list and set file system settings. The first screenshot lists the reserved space for root (which is set at five percent).

```
[root@rhel4 ~]# tune2fs -l /dev/sda1 | grep -i "block count"
Block count:              104388
Reserved block count:     5219
[root@rhel4 ~]#
```

This example changes this value to ten percent. You can use tune2fs while the file system is active, even if it is the root file system (as in this example).

```
[root@rhel4 ~]# tune2fs -m10 /dev/sda1
tune2fs 1.35 (28-Feb-2004)
Setting reserved blocks percentage to 10 (10430 blocks)
[root@rhel4 ~]# tune2fs -l /dev/sda1 | grep -i "block count"
Block count:              104388
Reserved block count:     10430
[root@rhel4 ~]#
```

14.3.5. Checking a file system

The **fsck** command is a front end tool used to check a file system for errors.

```
[root@RHEL4b ~]# ls /sbin/*fsck*
/sbin/dosfsck  /sbin/fsck         /sbin/fsck.ext2  /sbin/fsck.msdos
/sbin/e2fsck   /sbin/fsck.cramfs  /sbin/fsck.ext3  /sbin/fsck.vfat
[root@RHEL4b ~]#
```

The last column in **/etc/fstab** is used to determine whether a file system should be checked at bootup.

```
[paul@RHEL4b ~]$ grep ext /etc/fstab
/dev/VolGroup00/LogVol00   /             ext3    defaults        1 1
LABEL=/boot                /boot         ext3    defaults        1 2
```

```
[paul@RHEL4b ~]$
```

Manually checking a mounted file system results in a warning from fsck.

```
[root@RHEL4b ~]# fsck /boot
fsck 1.35 (28-Feb-2004)
e2fsck 1.35 (28-Feb-2004)
/dev/sda1 is mounted.

WARNING!!!  Running e2fsck on a mounted filesystem may cause
SEVERE filesystem damage.

Do you really want to continue (y/n)? no

check aborted.
[root@RHEL4b ~]#
```

But after unmounting fsck and **e2fsck** can be used to check an ext2 file system.

```
[root@RHEL4b ~]# fsck  /boot
fsck 1.35 (28-Feb-2004)
e2fsck 1.35 (28-Feb-2004)
/boot: clean, 44/26104 files, 17598/104388 blocks
[root@RHEL4b ~]# fsck -p /boot
fsck 1.35 (28-Feb-2004)
/boot: clean, 44/26104 files, 17598/104388 blocks
[root@RHEL4b ~]# e2fsck -p /dev/sda1
/boot: clean, 44/26104 files, 17598/104388 blocks
[root@RHEL4b ~]#
```

14.3.6. Practice File Systems

1. List the filesystems that are known by your system.

2. Create an ext2 filesystem on the 50MB partition.

3. Create an ext3 filesystem on the 4GB primary and one of the 1GB logical drives.

4. Set the reserved space for root on the logical drive to 0 percent.

5. Verify your work with the usual commands.

6. Put a reiserfs on one of the logical drives.

14.4. Mounting

Once you've put a file system on a partition, you can **mount** it. Mounting a file system makes it available for use, usually as a directory. We say **mounting a file system** instead of mounting a partition because we will see later that we can also mount file systems that do not exists on partitions.

14.4.1. Mounting local disks

On all Unix systems, every file and every directory is part of one big file tree. To access a file, you need to know the full path starting from the root directory. When adding a file system to your computer, you need to make it available somewhere in the file tree. The directory where you make a file system available is called a **mount point**. Once mounted, the new file system is accessible to users. The screenshot below shows the creation of a mount point, and the mounting of an ext2 partition on a newly added SCSI disk.

```
root@RHELv4u2:~# mkdir /home/project55
root@RHELv4u2:~# mount -t ext2 /dev/sdb1 /home/project55/
root@RHELv4u2:~# ls /home/project55/
lost+found
root@RHELv4u2:~#
```

Actually the explicit -t ext2 option to set the file system is not always necesarry. The mount command is able to automatically detect a lot of file systems on partitions.

14.4.2. Displaying mounted file systems

To view all mounted file systems, look at the files **/proc/mounts** or **/etc/mtab**. The kernel provides the info in /proc/mount in file form, but /proc/mount does not exist as a file on any hard disk. Looking at /proc/mount is the best way to be sure, since the information comes directly from the kernel. The /etc/mtab file on the other hand is updated by the mount command. Do not edit /etc/mtab manually!

Another way to view all mounts is by issuing the **mount** command without any arguments. The screenshot below pipes the output of these three through grep, to only show our added SCSI disk.

```
root@RHELv4u2:~# cat /proc/mounts | grep /dev/sdb
/dev/sdb1 /home/project55 ext2 rw 0 0
root@RHELv4u2:~# cat /etc/mtab | grep /dev/sdb
/dev/sdb1 /home/project55 ext2 rw 0 0
root@RHELv4u2:~# mount | grep /dev/sdb
/dev/sdb1 on /home/project55 type ext2 (rw)
```

A more user friendly way to look at mounted hard disks is **df**. The df(diskfree) command has the added benefit of showing you the free space on each mounted disk. Like a lot of Linux commands, df supports the **-h** switch to make the output more **human readable**.

```
root@RHELv4u2:~# df
Filesystem           1K-blocks      Used Available Use% Mounted on
/dev/mapper/VolGroup00-LogVol00
11707972   6366996   4746240  58% /
/dev/sda1               101086      9300     86567  10% /boot
none                    127988         0    127988   0% /dev/shm
```

```
/dev/sdb1             108865    1550   101694   2% /home/project55
root@RHELv4u2:~# df -h
Filesystem            Size  Used Avail Use% Mounted on
/dev/mapper/VolGroup00-LogVol00
12G  6.1G  4.6G  58% /
/dev/sda1              99M  9.1M   85M  10% /boot
none                  125M     0  125M   0% /dev/shm
/dev/sdb1             107M  1.6M  100M   2% /home/project55
```

14.4.3. Permanent mounts

Until now, we performed all mounts manually. This works nice, until the next reboot. Luckily there is a way to tell your computer to automatically mount certain file systems during boot. This is done using the file system table located in the **/etc/fstab** file. Below is a sample /etc/fstab file.

```
root@RHELv4u2:~# cat /etc/fstab
/dev/VolGroup00/LogVol00 /                ext3    defaults        1 1
LABEL=/boot             /boot             ext3    defaults        1 2
none                    /dev/pts          devpts  gid=5,mode=620  0 0
none                    /dev/shm          tmpfs   defaults        0 0
none                    /proc             proc    defaults        0 0
none                    /sys              sysfs   defaults        0 0
/dev/VolGroup00/LogVol01 swap             swap    defaults        0 0
```

By adding the following two lines, we can automate the mounting of these file systems. The first line is for our freshly added SCSI disk, the second line mounts an NFS share.

```
/dev/sdb1               /home/project55     ext2    defaults    0 0
server12:/mnt/data/iso   /home/iso           nfs     defaults    0 0
```

14.4.4. Disk Usage (du)

The **du** command can summarize disk usage for files and directories. Preventing du to go into subdirectories with the -s option will give you a total for that directory. This option is often used together with -h, so **du -sh** on a mount point gives the total amount used in that partition.

```
root@pasha:~# du -sh /home/reet
881G     /home/reet
```

14.4.5. Disk Free (df)

In the **df -h** example below you can see the size, free space, used gigabytes and percentage and mount point of a partition.

```
root@laika:~# df -h | egrep -e "(sdb2|File)"
Filesystem            Size Used Avail Use% Mounted on
/dev/sdb2              92G   83G  8.6G  91% /media/sdb2
root@laika:~#
```

14.4.6. Practice Mounting File Systems

1. Mount the small 50MB partition on /home/project22.

2. Mount the big primary partition on /mnt, the copy some files to it (everything in /etc). Then mount the partition as read only on /srv/nfs/salesnumbers.

3. Verify your work with fdisk, df, mount. Also look in /etc and /proc to interesting files.

4. Make both mounts permanent, test that it works.

5. What happens when you mount a partition on a directory that contains some files ?

6. What happens when you mount two partitions on the same mountpoint ?

7. Describe the difference between these file searching commands: find, locate, updatedb, whereis, apropos and which.

8. Perform a file system check on the partition mounted at /srv/nfs/salesnumbers.

14.5. UUID and filesystems

14.5.1. About Unique Objects

A **UUID** or **Universally Unique Identifier**is used to uniquely identify objects. This 128bit standard allows anyone to create a unique UUID. Below we use the **vol_id** utility to display the UUID of an ext3 partition on a hard disk.

```
root@laika:~# vol_id --uuid /dev/sda1
825d4b79-ec40-4390-8a71-9261df8d4c82
```

Red Hat Enterprise Linux 5 does not put the **vol_id** command in the PATH. But you can use **tune2fs** or type the path to the vol_id command to display the UUID of a volume.

```
[root@RHEL5 ~]# tune2fs -l /dev/sda1 | grep UUID
Filesystem UUID:          11cfc8bc-07c0-4c3f-9f64-78422ef1dd5c
[root@RHEL5 ~]# /lib/udev/vol_id -u /dev/sda1
```

```
11cfc8bc-07c0-4c3f-9f64-78422ef1dd5c
```

14.5.2. Using UUID in /etc/fstab

You can use the UUID to make sure that a volume is universally uniquely identified in **/etc/fstab**. The device name can change depending on the disk devices that are present at boot time, but a UUID never changes.

First we use **tune2fs** to find the UUID.

```
[root@RHEL5 ~]# tune2fs -l /dev/sdc1 | grep UUID
Filesystem UUID:          7626d73a-2bb6-4937-90ca-e451025d64e8
```

Then we check that it is properly added to **/etc/fstab**, the UUID replaces the variable devicename /dev/sdc1.

```
[root@RHEL5 ~]# grep UUID /etc/fstab
UUID=7626d73a-2bb6-4937-90ca-e451025d64e8 /home/pro42 ext3 defaults 0 0
```

Now we can mount the volume using the mountpoint defined in /etc/fstab.

```
[root@RHEL5 ~]# mount /home/pro42
[root@RHEL5 ~]# df -h | grep 42
/dev/sdc1             397M   11M  366M   3% /home/pro42
```

The real test now, is to remove /dev/sdb from the system, reboot the machine and see what happens. After the reboot, the disk previously known as /dev/sdc is now /dev/sdb.

```
[root@RHEL5 ~]# tune2fs -l /dev/sdb1 | grep UUID
Filesystem UUID:          7626d73a-2bb6-4937-90ca-e451025d64e8
```

And thanks to the UUID in /etc/fstab, the mountpoint is mounted on the same disk as before.

```
[root@RHEL5 ~]# df -h | grep sdb
/dev/sdb1             397M   11M  366M   3% /home/pro42
```

14.5.3. Practice UUID

1. Find the UUID of one of your ext3 partitions with tune2fs and vol_id.

2. USe this UUID in /etc/fstab and test that it works when a disk is removed (and the device name is changed). (You can edit settings in vmware to remove a hard disk.)

14.6. RAID

14.6.1. Hardware or software

Redundant Array of Independent Disks or **RAID** can be set up using hardware or software. Hardware RAID is more expensive, but offers better performance. Software RAID is cheaper and easier to manage, but it uses your CPU and your memory.

14.6.2. RAID levels

14.6.2.1. RAID 0

RAID 0 uses two or more disks, and is often called **striping** (or stripe set, or striped volume). Data is divided in **chunks**, those chunks are evenly spread across every disk in the array. The main advantage of RAID 0 is that you can create **larger drives**. RAID 0 is the only RAID without redundancy.

14.6.2.2. JBOD

JBOD uses two or more disks, and is often called **concatenating** (spanning, spanned set, or spanned volume). Data is written to the first disk, until it is full. Then data is written to the second disk... The main advantage of JBOD (Just a Bunch of Disks) is that you can create **larger drives**. JBOD offers no redundancy.

14.6.2.3. RAID 1

RAID 1 uses exactly two disks, and is often called **mirroring** (or mirror set, or mirrored volume). All data written to the array is written on each disk. The main advantage of RAID 1 is **redundancy**. The main disadvantage is that you lose at least half of your available disk space (in other words, you at least double the cost).

14.6.2.4. RAID 2, 3 and 4 ?

RAID 2 uses bit level striping, RAID 3 byte level, and RAID 4 is the same as RAID 5, but with a dedicated parity disk. This is actually slower than RAID 5, because every write would have to write parity to this one (bottleneck) disk. It is unlikely that you will ever see these RAID levels in production.

14.6.2.5. RAID 5

RAID 5 uses **three** or more disks, each divided into chunks. Every time chunks are written to the array, one of the disks will receive a **parity** chunk. Unlike RAID 4, the parity chunk will alternate between all disks. The main advantage of this is that RAID 5 will allow for full data recovery in case of **one** hard disk failure.

14.6.2.6. RAID 6

RAID 6 is very similar to RAID 5, but uses two parity chunks. RAID 6 protects against two hard disk failures.

14.6.2.7. RAID 0+1

RAID 0+1 is a mirror(1) of stripes(0). This means you first create two RAID 0 stripe sets, and then you set them up as a mirror set. For example, when you have six 100GB disks, then the stripe sets are each 300GB. Combined in a mirror, this makes 300GB total. RAID 0+1 will survive one disk failure. It will only survive the second disk failure if this disk is in the same stripe set as the previous failed disk.

14.6.2.8. RAID 1+0

RAID 1+0 is a stripe(0) of mirrors(1). For example, when you have six 100GB disks, then you first create three mirrors of 100GB each. You then stripe them together into a 300GB drive. In this example, as long as not all disks in the same mirror fail, it can survive up to three hard disk failures.

14.6.2.9. RAID 50

RAID 5+0 is a stripe(0) of RAID 5 arrays. Suppose you have nine disks of 100GB, then you can create three RAID 5 arrays of 200GB each. You can then combine them into one large stripe set.

14.6.2.10. many others

There are many other nested RAID combinations, like RAID 30, 51, 60, 100, 150, ...

14.6.3. Building a software RAID array

You can do this during the installation with Disk Druid (easy), or afterwards using the commandline (not so easy).

First, you have to attach some disks to your computer. In this scenario, three brand new disks of one gigabyte each are added. Check with **fdisk -l** that they are connected.

```
root@RHELv4u2:~# fdisk -l

Disk /dev/sda: 12.8 GB, 12884901888 bytes
```

```
255 heads, 63 sectors/track, 1566 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

Device Boot      Start         End      Blocks   Id  System
/dev/sda1   *           1          13      104391   83  Linux
/dev/sda2              14        1566    12474472+  8e  Linux LVM

Disk /dev/sdb: 1073 MB, 1073741824 bytes
255 heads, 63 sectors/track, 130 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

Disk /dev/sdb doesn't contain a valid partition table

Disk /dev/sdc: 1073 MB, 1073741824 bytes
255 heads, 63 sectors/track, 130 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

Disk /dev/sdc doesn't contain a valid partition table

Disk /dev/sdd: 1073 MB, 1073741824 bytes
255 heads, 63 sectors/track, 130 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

Disk /dev/sdd doesn't contain a valid partition table
```

So far so good! Next step is to create a partition of type **fd** on every disk. The fd type is to set the partition as **Linux RAID auto**. Like this screenshot shows.

```
root@RHELv4u2:~# fdisk /dev/sdc
Device contains neither a valid DOS partition table, nor Sun, SGI or \
OSF disklabel
Building a new DOS disklabel. Changes will remain in memory only,
until you decide to write them. After that, of course, the previous
content won't be recoverable.

Warning: invalid flag 0x0000 of partition table 4 will be corrected b\
y w(rite)

Command (m for help): n
Command action
e   extended
p   primary partition (1-4)
p
Partition number (1-4): 1
First cylinder (1-130, default 1):
Using default value 1
Last cylinder or +size or +sizeM or +sizeK (1-130, default 130):
Using default value 130

Command (m for help): t
Selected partition 1
Hex code (type L to list codes): fd
Changed system type of partition 1 to fd (Linux raid autodetect)

Command (m for help): p

Disk /dev/sdc: 1073 MB, 1073741824 bytes
255 heads, 63 sectors/track, 130 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

Device Boot      Start     End      Blocks   Id  System
/dev/sdc1          1       130     1044193+  fd  Linux raid autodetect
```

```
Command (m for help): w
The partition table has been altered!

Calling ioctl() to re-read partition table.
Syncing disks.
root@RHELv4u2:~#
```

Now all three disks are ready for RAID, so we have to tell the system what to do with these disks.

```
root@RHELv4u2:~# fdisk -l

Disk /dev/sda: 12.8 GB, 12884901888 bytes
255 heads, 63 sectors/track, 1566 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

Device Boot      Start         End      Blocks   Id  System
/dev/sda1   *           1          13      104391   83  Linux
/dev/sda2              14        1566    12474472+  8e  Linux LVM

Disk /dev/sdb: 1073 MB, 1073741824 bytes
255 heads, 63 sectors/track, 130 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

Device Boot      Start      End      Blocks   Id  System
/dev/sdb1          1        130     1044193+  fd  Linux raid autodetect

Disk /dev/sdc: 1073 MB, 1073741824 bytes
255 heads, 63 sectors/track, 130 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

Device Boot      Start      End      Blocks   Id  System
/dev/sdc1          1        130     1044193+  fd  Linux raid autodetect

Disk /dev/sdd: 1073 MB, 1073741824 bytes
255 heads, 63 sectors/track, 130 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

Device Boot      Start      End      Blocks   Id  System
/dev/sdd1          1        130     1044193+  fd  Linux raid autodetect
```

The next step used to be *create the RAID table in **/etc/raidtab***. Nowadays, you can just issue the command **mdadm** with the correct parameters. The command below is split on two lines to fit this print, but you should type it on one line, without the backslash (\).

```
root@RHELv4u2:~# mdadm --create /dev/md0 --chunk=64 --level=5 --raid-d\
evices=3 /dev/sdb1 /dev/sdc1 /dev/sdd1
mdadm: array /dev/md0 started.
```

Below a partial screenshot how fdisk -l sees the RAID5

```
root@RHELv4u2:~# fdisk -l

<cut>
```

```
Disk /dev/md0: 2138 MB, 2138308608 bytes
2 heads, 4 sectors/track, 522048 cylinders
Units = cylinders of 8 * 512 = 4096 bytes

Disk /dev/md0 doesn't contain a valid partition table
```

We will use this software RAID 5 array in the next topic, LVM.

14.6.4. /proc/mdstat

The status of the raid devices can be seen in **/proc/mdstat**. This example shows a RAID 5 in the process of rebuilding.

```
[root@RHEL5 ~]# cat /proc/mdstat
Personalities : [raid6] [raid5] [raid4]
md0 : active raid5 sdg1[3] sdf1[1] sde1[0]
      1677056 blocks level 5, 64k chunk, algorithm 2 [3/2] [UU_]
      [=================>...]  recovery = 89.1% (747952/838528) finish\
=0.0min speed=25791K/sec

unused devices: >none<
```

This example shows an active software RAID 5.

```
[root@RHEL5 ~]# cat /proc/mdstat
Personalities : [raid6] [raid5] [raid4]
md0 : active raid5 sdg1[2] sdf1[1] sde1[0]
      1677056 blocks level 5, 64k chunk, algorithm 2 [3/3] [UUU]

unused devices: >none<
```

14.6.5. Removing a software RAID

The software raid is visible in /proc/mdstat when active. To remove the raid completely so you can use the disks for other purposes, you first have to stop (de-activate) it with mdadm.

```
mdadm --stop /dev/mdadm
```

When stopped, you can remove the raid with mdadm.

```
mdadm --remove /dev/mdadm
```

The disks can now be repartitioned.

14.6.6. Practice RAID

1. Add three virtual disks of 200MB each to the virtual Red Hat machine.

2. Create a software RAID 5 on the three disks. (It is not necessary to put a filesystem on it)

3. Verify with fdisk and in /proc/ that the RAID exists.

4. (optional) Stop and remove the RAID, unless you want to use it in the next chapter LVM.

Chapter 15. Logical Volume Management

15.1. Introduction to lvm

15.1.1. Problems with standard partitions

There are some problems when working with hard disks and standard partitions. Consider a system with a small and a large hard disk device, partitioned like this. The first disk (/dev/sda) is partitioned in two, the second disk (/dev/sdb) has three partitions.

Table 15.1. Disk Partitioning Example

/dev/sda		**/dev/sdb**			
/dev/sda1	/dev/sda2	/dev/sdb1	/dev/sdb2	/dev/sdb3	unused
/boot	/	/var	/home	/project42	
ext2	ext3	ext2	reiserfs	ext3	

In the example above, consider the options when you want to enlarge the space available for project42. What can you do ? The solution will always force you to unmount the filesystem, take a backup of the data, remove and recreate partitions, and then restore the data and remount the file system.

15.1.2. Solution with lvm

Using **lvm** will create a virtual layer between the mounted file systems and the hardware devices. This virtual layer will allow for an administrator to enlarge a mounted file system in use. When lvm is properly used, then there is no need to unmount the file system to enlarge it.

Table 15.2. LVM Example

/dev/sda		**/dev/sdb**			
Volume Group					
/boot	/	/var	/home	/project42	
ext2	ext3	ext2	reiserfs	ext3	

15.1.3. About lvm

Most lvm implementations support **physical storage grouping**, **logical volume resizing** and **data migration**.

Physical storage grouping is a fancy name for grouping multiple physical devices (hard disks) into a logical mass storage device. To enlarge this physical group, hard disks or even single partitions can be added at a later time. The size of LVM volumes on this physical group is independent of the individual size of the components. The total size of the group is the limit.

One of the nicest features of LVM is the logical volume resizing. You can increase the size of an LVM volume, sometimes even without any downtime. Additionally, you can migrate data away from a failing hard disk device.

15.2. LVM Terminology

15.2.1. Physical Volume (pv)

A **physical volume** is a disk, a partition or a (hardware or software) RAID device. All these devices can become a member of a **Volume Group**.

15.2.2. Volume Group (vg)

A **Volume Group** is an abstraction layer between **Physical Devices** and **Logical Volumes**.

15.2.3. Logical Volume (lv)

A **Logical Volume** is created in a **Volume Group**. Logical Volumes that contain a file system can be mounted. The use of logical volumes is similar to the use of partitions (both are standard block devices) and is accomplished with the same standard commands (mkfs, mount, fsck, df, ...).

15.3. Verifying existing Physical Volumes

15.3.1. lvmdiskscan

To get a list of block devices that can be used with LVM, use **lvmdiskscan**. The example below uses grep to limit the result to SCSI devices.

```
[root@RHEL5 ~]# lvmdiskscan | grep sd
  /dev/sda1                [      101.94 MB]
  /dev/sda2                [       15.90 GB] LVM physical volume
  /dev/sdb                 [      409.60 MB]
  /dev/sdc                 [      409.60 MB]
  /dev/sdd                 [      409.60 MB] LVM physical volume
  /dev/sde1                [       95.98 MB]
  /dev/sde5                [      191.98 MB]
  /dev/sdf                 [      819.20 MB] LVM physical volume
  /dev/sdg1                [      818.98 MB]
[root@RHEL5 ~]#
```

15.3.2. pvs

The easiest way to verify whether devices are known to lvm is with the **pvs** command. The screenshot below shows that only /dev/sda2 is currently known for use with LVM. It shows that /dev/sda2 is part of Volgroup00 and is almost 16GB in size. It also shows /dev/sdc and /dev/sdd as part of vg33. The device /dev/sdb is knwon to lvm, but not linked to any Volume Group.

```
[root@RHEL5 ~]# pvs
  PV         VG         Fmt  Attr PSize   PFree
  /dev/sda2  VolGroup00 lvm2 a-    15.88G      0
  /dev/sdb              lvm2 --   409.60M 409.60M
  /dev/sdc   vg33       lvm2 a-   408.00M 408.00M
  /dev/sdd   vg33       lvm2 a-   408.00M 408.00M
[root@RHEL5 ~]#
```

15.3.3. pvscan

The **pvscan** command will scan all disks for existing Physical Volumes. The information is similar to pvs, plus you get a line with total sizes.

```
[root@RHEL5 ~]# pvscan
  PV /dev/sdc    VG vg33         lvm2 [408.00 MB / 408.00 MB free]
  PV /dev/sdd    VG vg33         lvm2 [408.00 MB / 408.00 MB free]
  PV /dev/sda2   VG VolGroup00   lvm2 [15.88 GB / 0    free]
  PV /dev/sdb                    lvm2 [409.60 MB]
  Total: 4 [17.07 GB] / in use: 3 [16.67 GB] / in no VG: 1 [409.60 MB]
[root@RHEL5 ~]#
```

15.3.4. pvdisplay

Use **pvdisplay** to get more information about physical volumes. You can also use **pvdisplay** without an argument to display information about all physical (lvm) volumes.

```
[root@RHEL5 ~]# pvdisplay /dev/sda2
  --- Physical volume ---
  PV Name               /dev/sda2
  VG Name               VolGroup00
  PV Size               15.90 GB / not usable 20.79 MB
  Allocatable           yes (but full)
  PE Size (KByte)       32768
  Total PE              508
  Free PE               0
  Allocated PE          508
  PV UUID               TobYfp-Ggg0-Rf8r-xtLd-5XgN-RSPc-8vkTHD

[root@RHEL5 ~]#
```

15.4. Verifying existing Volume Groups

15.4.1. vgs

Similar to **pvs** is the use of **vgs** to display a quick overview of all volume groups. There is only one volume group in the screenshot below, it is named VolGroup00 and is almost 16GB in size.

```
[root@RHEL5 ~]# vgs
  VG         #PV #LV #SN Attr   VSize  VFree
  VolGroup00   1   2   0 wz--n- 15.88G    0
[root@RHEL5 ~]#
```

15.4.2. vgscan

The **vgscan** command will scan all disks for existing Volume Groups. It will also update the **/etc/lvm/.cache** file. This file contains a list of all current lvm devices.

```
[root@RHEL5 ~]# vgscan
  Reading all physical volumes.  This may take a while...
  Found volume group "VolGroup00" using metadata type lvm2
[root@RHEL5 ~]#
```

LVM will run the vgscan automatically at bootup, so if you add hotswap devices, then you will need to run vgscan to update /etc/lvm/.cache with the new devices.

15.4.3. vgdisplay

The **vgdisplay** command will give you more detailed information about a volume group (or about all volume groups if you omit the argument).

```
[root@RHEL5 ~]# vgdisplay VolGroup00
  --- Volume group ---
  VG Name               VolGroup00
  System ID
  Format                lvm2
  Metadata Areas        1
  Metadata Sequence No  3
  VG Access             read/write
  VG Status             resizable
  MAX LV                0
  Cur LV                2
  Open LV               2
  Max PV                0
  Cur PV                1
  Act PV                1
  VG Size               15.88 GB
  PE Size               32.00 MB
  Total PE              508
  Alloc PE / Size       508 / 15.88 GB
  Free  PE / Size       0 / 0
  VG UUID               qsXvJb-71qV-9l7U-ishX-FobM-qptE-VXmKIg

[root@RHEL5 ~]#
```

15.5. Verifying existing Logical Volumes

15.5.1. lvs

Use **lvs** for a quick look at all existing logical volumes. Below you can see two logical volumes named LogVol00 and LogVol01.

```
[root@RHEL5 ~]# lvs
  LV       VG         Attr   LSize  Origin Snap%  Move Log Copy%
  LogVol00 VolGroup00 -wi-ao 14.88G
  LogVol01 VolGroup00 -wi-ao  1.00G
[root@RHEL5 ~]#
```

15.5.2. lvscan

The **lvscan** command will scan all disks for existing Logical Volumes.

```
[root@RHEL5 ~]# lvscan
  ACTIVE            '/dev/VolGroup00/LogVol00' [14.88 GB] inherit
  ACTIVE            '/dev/VolGroup00/LogVol01' [1.00 GB] inherit
[root@RHEL5 ~]#
```

15.5.3. lvdisplay

More detailed information about logical volumes is available through the **lvdisplay(1)** command.

```
[root@RHEL5 ~]# lvdisplay VolGroup00/LogVol01
  --- Logical volume ---
  LV Name                /dev/VolGroup00/LogVol01
  VG Name                VolGroup00
  LV UUID                RnTGK6-xWsi-t530-ksJx-7cax-co5c-A1KlDp
  LV Write Access        read/write
  LV Status              available
  # open                 1
  LV Size                1.00 GB
  Current LE             32
  Segments               1
  Allocation             inherit
  Read ahead sectors     0
  Block device           253:1

[root@RHEL5 ~]#
```

15.6. Manage Physical Volumes

15.6.1. pvcreate

Use the **pvcreate** command to add devices to lvm. This example shows how to add a disk (or hardware RAID device) to lvm.

```
[root@RHEL5 ~]# pvcreate /dev/sdb
  Physical volume "/dev/sdb" successfully created
[root@RHEL5 ~]#
```

This example shows how to add a partition to lvm.

```
[root@RHEL5 ~]# pvcreate /dev/sdc1
  Physical volume "/dev/sdc1" successfully created
[root@RHEL5 ~]#
```

You can also add multiple disks or partitions as target to pvcreate. This example adds three disks to lvm.

```
[root@RHEL5 ~]# pvcreate /dev/sde /dev/sdf /dev/sdg
  Physical volume "/dev/sde" successfully created
  Physical volume "/dev/sdf" successfully created
  Physical volume "/dev/sdg" successfully created
[root@RHEL5 ~]#
```

15.6.2. pvremove

Use the **pvremove** command to remove physical volumes from lvm. The devices may not be in use.

```
[root@RHEL5 ~]# pvremove /dev/sde /dev/sdf /dev/sdg
  Labels on physical volume "/dev/sde" successfully wiped
  Labels on physical volume "/dev/sdf" successfully wiped
  Labels on physical volume "/dev/sdg" successfully wiped
[root@RHEL5 ~]#
```

15.6.3. pvresize

When you used fdisk to resize a partition on a disk, then you must use **pvresize** to make lvm recognize the new size of the physical volume that represents this partition.

```
[root@RHEL5 ~]# pvresize /dev/sde1
  Physical volume "/dev/sde1" changed
  1 physical volume(s) resized / 0 physical volume(s) not resized
```

15.6.4. pvchange

With **pvchange** you can prevent the allocation of a Physical Volume in a new Volume Group or Logical Volume. This can be useful if you plan to remove a Physical Volume.

```
[root@RHEL5 ~]# pvchange -xn /dev/sdd
  Physical volume "/dev/sdd" changed
  1 physical volume changed / 0 physical volumes not changed
[root@RHEL5 ~]#
```

To revert your previous decision, this example shows you how te re-enable the Physical Volume to allow allocation.

```
[root@RHEL5 ~]# pvchange -xy /dev/sdd
  Physical volume "/dev/sdd" changed
  1 physical volume changed / 0 physical volumes not changed
[root@RHEL5 ~]#
```

15.6.5. pvmove

With **pvmove** you can move Logical Volumes from within a Volume Group to another Physical Volume. This must be done before removing a Physical Volume.

```
[root@RHEL5 ~]# pvs | grep vg1
  /dev/sdf   vg1        lvm2 a-   816.00M      0
  /dev/sdg   vg1        lvm2 a-   816.00M 816.00M
[root@RHEL5 ~]# pvmove /dev/sdf
  /dev/sdf: Moved: 70.1%
  /dev/sdf: Moved: 100.0%
[root@RHEL5 ~]# pvs | grep vg1
  /dev/sdf   vg1        lvm2 a-   816.00M 816.00M
  /dev/sdg   vg1        lvm2 a-   816.00M      0
```

15.7. Manage Volume Groups

15.7.1. vgcreate

Use the **vgcreate** command to create a volume group. You can immediately name all the physical volumes that span the volume group.

```
[root@RHEL5 ~]# vgcreate vg42 /dev/sde /dev/sdf
  Volume group "vg42" successfully created
[root@RHEL5 ~]#
```

15.7.2. vgextend

Use the **vgextend** command to extend an existing volume group with a physical volume.

```
[root@RHEL5 ~]# vgextend vg42 /dev/sdg
  Volume group "vg42" successfully extended
[root@RHEL5 ~]#
```

15.7.3. vgremove

Use the **vgremove** command to remove volume groups from lvm. The volume groups may not be in use.

```
[root@RHEL5 ~]# vgremove vg42
  Volume group "vg42" successfully removed
[root@RHEL5 ~]#
```

15.7.4. vgreduce

Use the **vgreduce** command to remove a Physical Volume from the Volume Group.

The following example adds Physical Volume /dev/sdg to the vg1 Volume Group using vgextend. And then removes it again using vgreduce.

```
[root@RHEL5 ~]# pvs | grep sdg
  /dev/sdg              lvm2 --   819.20M 819.20M
[root@RHEL5 ~]# vgextend vg1 /dev/sdg
  Volume group "vg1" successfully extended
[root@RHEL5 ~]# pvs | grep sdg
  /dev/sdg   vg1        lvm2 a-   816.00M 816.00M
[root@RHEL5 ~]# vgreduce vg1 /dev/sdg
  Removed "/dev/sdg" from volume group "vg1"
[root@RHEL5 ~]# pvs | grep sdg
  /dev/sdg              lvm2 --   819.20M 819.20M
```

15.7.5. vgchange

Use the **vgchange** command to change parameters of a Volume Group.

This example shows how to prevent Physical Volumes from being added or removed to the Volume Group vg1.

```
[root@RHEL5 ~]# vgchange -xn vg1
  Volume group "vg1" successfully changed
[root@RHEL5 ~]# vgextend vg1 /dev/sdg
  Volume group vg1 is not resizeable.
```

You can also use vgchange to change most other properties of a Volume Group. This example changes the maximum nhumber of Logical Volumes and maximum number of Physical Volumes that vg1 can serve.

```
[root@RHEL5 ~]# vgdisplay vg1 | grep -i max
  MAX LV                0
  Max PV                0
[root@RHEL5 ~]# vgchange -l16 vg1
  Volume group "vg1" successfully changed
[root@RHEL5 ~]# vgchange -p8 vg1
  Volume group "vg1" successfully changed
[root@RHEL5 ~]# vgdisplay vg1 | grep -i max
  MAX LV                16
  Max PV                8
```

15.7.6. vgmerge

Merging two Volume Groups into one is done with **vgmerge**. The following example merges vg2 into vg1, keeping all the properties of vg1.

```
[root@RHEL5 ~]# vgmerge vg1 vg2
  Volume group "vg2" successfully merged into "vg1"
[root@RHEL5 ~]#
```

15.8. Manage Logical Volumes

15.8.1. lvcreate

Use the **lvcreate** command to create Logical Volumes in a Volume Group. This example creates an 8GB Logical Volume in Volume Group vg42.

```
[root@RHEL5 ~]# lvcreate -L5G vg42
```

```
  Logical volume "lvol0" created
[root@RHEL5 ~]#
```

As you can see, lvm automatically names the Logical Volume **lvol0**. The next example creates a 200MB Logical Volume named MyLV in Volume Group vg42.

```
[root@RHEL5 ~]# lvcreate -L200M -nMyLV vg42
  Logical volume "MyLV" created
[root@RHEL5 ~]#
```

The next example does the same thing, but with different syntax.

```
[root@RHEL5 ~]# lvcreate --size 200M -n MyLV vg42
  Logical volume "MyLV" created
[root@RHEL5 ~]#
```

This example creates a Logical Volume that occupies 10 percent of the Volume Group.

```
[root@RHEL5 ~]# lvcreate -l 10%VG -n MyLV2 vg42
  Logical volume "MyLV2" created
[root@RHEL5 ~]#
```

This example creates a Logical Volume that occupies 30 percent of the remaining free space in the Volume Group.

```
[root@RHEL5 ~]# lvcreate -l 30%FREE -n MyLV3 vg42
  Logical volume "MyLV3" created
[root@RHEL5 ~]#
```

15.8.2. lvremove

Use the **lvremove** command to remove Logical Volumes from a Volume Group. Removing a Logical Volume requires the name of the Volume Group.

```
[root@RHEL5 ~]# lvremove vg42/MyLV
Do you really want to remove active logical volume "MyLV"? [y/n]: y
  Logical volume "MyLV" successfully removed
[root@RHEL5 ~]#
```

Removing multiple Logical Volumes will request confirmation for each individual volume.

```
[root@RHEL5 ~]# lvremove vg42/MyLV vg42/MyLV2 vg42/MyLV3
Do you really want to remove active logical volume "MyLV"? [y/n]: y
  Logical volume "MyLV" successfully removed
Do you really want to remove active logical volume "MyLV2"? [y/n]: y
  Logical volume "MyLV2" successfully removed
```

```
Do you really want to remove active logical volume "MyLV3"? [y/n]: y
  Logical volume "MyLV3" successfully removed
[root@RHEL5 ~]#
```

15.8.3. lvextend

Extending the volume is easy with **lvextend**. This example extends a 200MB Logical Volume with 100 MB.

```
[root@RHEL5 ~]# lvdisplay /dev/vg2/lvol0 | grep Size
  LV Size                200.00 MB
[root@RHEL5 ~]# lvextend -L +100 /dev/vg2/lvol0
  Extending logical volume lvol0 to 300.00 MB
  Logical volume lvol0 successfully resized
[root@RHEL5 ~]# lvdisplay /dev/vg2/lvol0 | grep Size
  LV Size                300.00 MB
```

The next example creates a 100MB Logical Volume, and then extends it to 500MB.

```
[root@RHEL5 ~]# lvcreate --size 100M -n extLV vg42
  Logical volume "extLV" created
[root@RHEL5 ~]# lvextend -L 500M vg42/extLV
  Extending logical volume extLV to 500.00 MB
  Logical volume extLV successfully resized
[root@RHEL5 ~]#
```

This example doubles the size of a Logical Volume.

```
[root@RHEL5 ~]# lvextend -l+100%LV vg42/extLV
  Extending logical volume extLV to 1000.00 MB
  Logical volume extLV successfully resized
[root@RHEL5 ~]#
```

15.8.4. lvrename

Renaming a Logical Volume is done with **lvrename**. This example renames extLV to bigLV in the vg42 Volume Group.

```
[root@RHEL5 ~]# lvrename vg42/extLV vg42/bigLV
  Renamed "extLV" to "bigLV" in volume group "vg42"
[root@RHEL5 ~]#
```

15.9. Example: Using lvm

This example shows how you can use a device (in this case /dev/sdc, but it could have been /dev/sdb or any other disk or partition) with lvm, how to create a volume group (vg) and how to create and use a logical volume (vg/lvol0).

First thing to do, is create physical volumes that can join the volume group with **pvcreate**. This command makes a disk or partition available for use in Volume Groups. The screenshot shows how to present the SCSI Disk device to LVM.

```
root@RHEL4:~# pvcreate /dev/sdc
Physical volume "/dev/sdc" successfully created
```

Note for home users: lvm will work fine when using the complete disk, but another operating system on the same computer will not recognize lvm and will mark the disk as being empty! You can avoid this by creating a partition that spans the whole disk, then run pvcreate on the partition instead of the disk.

Then **vgcreate** creates a volume group using one device. Note that more devices could be added to the volume group.

```
root@RHEL4:~# vgcreate vg /dev/sdc
Volume group "vg" successfully created
```

The last step **lvcreate** creates a logical volume.

```
root@RHEL4:~# lvcreate --size 500m vg
Logical volume "lvol0" created
```

The logical volume /dev/vg/lvol0 can now be formatted with ext2, and mounted for normal use.

```
root@RHELv4u2:~# mke2fs -m0 -j /dev/vg/lvol0
mke2fs 1.35 (28-Feb-2004)
Filesystem label=
OS type: Linux
Block size=1024 (log=0)
Fragment size=1024 (log=0)
128016 inodes, 512000 blocks
0 blocks (0.00%) reserved for the super user
First data block=1
Maximum filesystem blocks=67633152
63 block groups
8192 blocks per group, 8192 fragments per group
2032 inodes per group
Superblock backups stored on blocks:
8193, 24577, 40961, 57345, 73729, 204801, 221185, 401409

Writing inode tables: done
Creating journal (8192 blocks): done
Writing superblocks and filesystem accounting information: done

This filesystem will be automatically checked every 37 mounts or
180 days, whichever comes first.  Use tune2fs -c or -i to override.
root@RHELv4u2:~# mkdir /home/project10
root@RHELv4u2:~# mount /dev/vg/lvol0 /home/project10/
root@RHELv4u2:~# df -h | grep proj
/dev/mapper/vg-lvol0  485M   11M  474M   3% /home/project10
```

A logical volume is very similar to a partition, it can be formatted with a file system, and can be mounted so users can access it.

15.10. Example: Extend a Logical Volume

A logical volume can be extended without unmounting the file system. Whether or not a volume can be extended depends on the file system it uses. Volumes that are mounted as vfat or ext2 cannot be extended, so in the example here we use the ext3 file system.

The fdisk command shows us newly added scsi-disks that will serve our lvm volume. This volume will then be extended. First, take a look at these disks.

```
[root@RHEL5 ~]# fdisk -l | grep sd[bc]
Disk /dev/sdb doesn't contain a valid partition table
Disk /dev/sdc doesn't contain a valid partition table
Disk /dev/sdb: 1181 MB, 1181115904 bytes
Disk /dev/sdc: 429 MB, 429496320 bytes
```

You already know how to partition a disk, below the first disk is partitioned (in one big primary partition), the second disk is left untouched.

```
[root@RHEL5 ~]# fdisk -l | grep sd[bc]
Disk /dev/sdc doesn't contain a valid partition table
Disk /dev/sdb: 1181 MB, 1181115904 bytes
/dev/sdb1               1         143     1148616   83  Linux
Disk /dev/sdc: 429 MB, 429496320 bytes
```

You also know how to prepare disks for lvm with **pvcreate**, and how to create a volume group with **vgcreate**. This example adds both the partitiond disk and the untouched disk to the volume group named **vg2**.

```
[root@RHEL5 ~]# pvcreate /dev/sdb1
  Physical volume "/dev/sdb1" successfully created
[root@RHEL5 ~]# pvcreate /dev/sdc
  Physical volume "/dev/sdc" successfully created
[root@RHEL5 ~]# vgcreate vg2 /dev/sdb1 /dev/sdc
  Volume group "vg2" successfully created
```

You can use **pvdisplay** to verify that both the disk and the partition belong to the volume group.

```
[root@RHEL5 ~]# pvdisplay | grep -B1 vg2
  PV Name               /dev/sdb1
  VG Name               vg2
--
  PV Name               /dev/sdc
  VG Name               vg2
```

And you are familiar both with the **lvcreate** command to create a small logical volume and the **mke2fs** command to put ext2 on it.

```
[root@RHEL5 ~]# lvcreate --size 200m vg2
  Logical volume "lvol0" created
[root@RHEL5 ~]# mke2fs -m20 -j /dev/vg2/lvol0
...
```

As you see, we end up with a mounted logical volume that according to **df** is almost 200 megabyte in size.

```
[root@RHEL5 ~]# mkdir /home/resizetest
[root@RHEL5 ~]# mount /dev/vg2/lvol0 /home/resizetest/
[root@RHEL5 ~]# df -h | grep resizetest
                      194M  5.6M  149M   4% /home/resizetest
```

Extending the volume is easy with **lvextend**.

```
[root@RHEL5 ~]# lvextend -L +100 /dev/vg2/lvol0
  Extending logical volume lvol0 to 300.00 MB
  Logical volume lvol0 successfully resized
```

But as you can see, there is a small problem: it appears that df is not able to display the extended volume in its full size. This is because the filesystem is only set for the size of the volume before the extension was added.

```
[root@RHEL5 ~]# df -h | grep resizetest
                      194M  5.6M  149M   4% /home/resizetest
```

With **lvdisplay** however we can see that the volume is indeed extended.

```
[root@RHEL5 ~]# lvdisplay /dev/vg2/lvol0 | grep Size
  LV Size                300.00 MB
```

To finish the extension, you need **resize2fs** to span the filesystem over the full size of the logical volume.

```
[root@RHEL5 ~]# resize2fs /dev/vg2/lvol0
resize2fs 1.39 (29-May-2006)
Filesystem at /dev/vg2/lvol0 is mounted on /home/resizetest; on-line re\
sizing required
Performing an on-line resize of /dev/vg2/lvol0 to 307200 (1k) blocks.
The filesystem on /dev/vg2/lvol0 is now 307200 blocks long.
```

Congratulations, you just successfully expanded a logical volume.

```
[root@RHEL5 ~]# df -h | grep resizetest
                      291M  6.1M  225M   3% /home/resizetest
[root@RHEL5 ~]#
```

15.11. Example: Resize a Physical Volume

This is a humble demonstration of how to resize a physical Volume with lvm (after you resize it with fdisk). The demonstration starts with a 100MB partition named /dev/sde1. We used fdisk to create it, and to verify the size.

```
[root@RHEL5 ~]# fdisk -l 2>/dev/null | grep sde1
/dev/sde1               1         100      102384   83  Linux
[root@RHEL5 ~]#
```

Now we can use pvcreate to create the Physical Volume, followed by pvs to verify the creation.

```
[root@RHEL5 ~]# pvcreate /dev/sde1
  Physical volume "/dev/sde1" successfully created
[root@RHEL5 ~]# pvs | grep sde1
  /dev/sde1             lvm2 --    99.98M  99.98M
[root@RHEL5 ~]#
```

The next step is ti use fdisk to enlarge the partition (actually deleting it and then recreating /dev/sde1 with more cylinders).

```
[root@RHEL5 ~]# fdisk /dev/sde

Command (m for help): p

Disk /dev/sde: 858 MB, 858993152 bytes
64 heads, 32 sectors/track, 819 cylinders
Units = cylinders of 2048 * 512 = 1048576 bytes

   Device Boot      Start         End      Blocks   Id  System
/dev/sde1               1         100      102384   83  Linux

Command (m for help): d
Selected partition 1

Command (m for help): n
Command action
   e   extended
   p   primary partition (1-4)
p
Partition number (1-4):
Value out of range.
Partition number (1-4): 1
First cylinder (1-819, default 1):
Using default value 1
Last cylinder or +size or +sizeM or +sizeK (1-819, default 819): 200

Command (m for help): w
The partition table has been altered!

Calling ioctl() to re-read partition table.
Syncing disks.
[root@RHEL5 ~]#
```

When we now use fdisk and pvs to verify the size of the partition and the Physical Volume, then there is a size difference. LVM is still using the old size.

```
[root@RHEL5 ~]# fdisk -l 2>/dev/null | grep sde1
/dev/sde1               1         200      204784   83  Linux
[root@RHEL5 ~]# pvs | grep sde1
  /dev/sde1             lvm2 --    99.98M  99.98M
```

```
[root@RHEL5 ~]#
```

Executing pvresize on the Physical Volume will make lvm aware of the size change of the partition. The correct size can be displayed with pvs.

```
[root@RHEL5 ~]# pvresize /dev/sde1
  Physical volume "/dev/sde1" changed
  1 physical volume(s) resized / 0 physical volume(s) not resized
[root@RHEL5 ~]# pvs | grep sde1
  /dev/sde1             lvm2 --   199.98M 199.98M
[root@RHEL5 ~]#
```

15.12. Example: Mirror a Logical Volume

We start by creating three physical volumes for lvm. Then we verify the creation and the size with pvs. Three physical disks because lvm uses two disks for the mirror and a third disk for the mirror log!

```
[root@RHEL5 ~]# pvcreate /dev/sdb /dev/sdc /dev/sdd
  Physical volume "/dev/sdb" successfully created
  Physical volume "/dev/sdc" successfully created
  Physical volume "/dev/sdd" successfully created
[root@RHEL5 ~]# pvs
  PV         VG         Fmt  Attr PSize   PFree
  /dev/sdb              lvm2 --   409.60M 409.60M
  /dev/sdc              lvm2 --   409.60M 409.60M
  /dev/sdd              lvm2 --   409.60M 409.60M
```

Then we create the Volume Group and verify again with pvs. Notice how the three physical volumes now belong to vg33, and how the size is rounded down (in steps of the extent size, here 4MB).

```
[root@RHEL5 ~]# vgcreate vg33 /dev/sdb /dev/sdc /dev/sdd
  Volume group "vg33" successfully created
[root@RHEL5 ~]# pvs
  PV         VG         Fmt  Attr PSize   PFree
  /dev/sda2  VolGroup00 lvm2 a-    15.88G      0
  /dev/sdb   vg33       lvm2 a-   408.00M 408.00M
  /dev/sdc   vg33       lvm2 a-   408.00M 408.00M
  /dev/sdd   vg33       lvm2 a-   408.00M 408.00M
[root@RHEL5 ~]#
```

The last step is to create the Logical Volume with **lvcreate**. Notice the **-m 1** switch to create one mirror. Notice also the change in free space in all three Physical Volumes!

```
[root@RHEL5 ~]# lvcreate --size 300m -n lvmir -m 1 vg33
  Logical volume "lvmir" created
[root@RHEL5 ~]# pvs
  PV         VG         Fmt  Attr PSize   PFree
  /dev/sda2  VolGroup00 lvm2 a-    15.88G      0
  /dev/sdb   vg33       lvm2 a-   408.00M 108.00M
  /dev/sdc   vg33       lvm2 a-   408.00M 108.00M
  /dev/sdd   vg33       lvm2 a-   408.00M 404.00M
```

You can see the copy status of the mirror with lvs. It currently shows a 100 percent copy.

```
[root@RHEL5 ~]# lvs vg33/lvmir
  LV    VG   Attr   LSize   Origin Snap%  Move Log        Copy%
  lvmir vg33 mwi-ao 300.00M                    lvmir_mlog 100.00
```

15.13. Example: Snapshot a Logical Volume

A snapshot is a virtual copy of all the data at a point in time on a volume. A snapshot Logical Volume will retain a copy of all changed files of the snapshotted Logical Volume.

The example below creates a snapshot of the bigLV Logical Volume.

```
[root@RHEL5 ~]# lvcreate -L100M -s -n snapLV vg42/bigLV
  Logical volume "snapLV" created
[root@RHEL5 ~]#
```

You can see with lvs that the snapshot snapLV is indeed a snapshot of bigLV. Moments after taking the snapshot, there are few changes to bigLV (0.02 percent).

```
[root@RHEL5 ~]# lvs
  LV       VG         Attr   LSize   Origin Snap%  Move Log Copy%
  bigLV    vg42       owi-a- 200.00M
  snapLV   vg42       swi-a- 100.00M bigLV    0.02
[root@RHEL5 ~]#
```

But after using bigLV for a while, more changes are done. This means the snapshot volume has to keep more original data (10.22 percent).

```
[root@RHEL5 ~]# lvs | grep vg42
  bigLV    vg42       owi-ao 200.00M
  snapLV   vg42       swi-a- 100.00M bigLV   10.22
[root@RHEL5 ~]#
```

You can now use regular backup tools (dump, tar, cpio, ...) to take a backup of the snapshot Logical Volume. This backup will contain all data as it existed on bigLV at the time the snapshot was taken. When the backup is done, you can remove the snapshot.

```
[root@RHEL5 ~]# lvremove vg42/snapLV
Do you really want to remove active logical volume "snapLV"? [y/n]: y
  Logical volume "snapLV" successfully removed
[root@RHEL5 ~]#
```

15.14. Practice LVM

1. Create a volume group that contains a complete disk and a partition on another disk.

2. Create two logical volumes (a small one and a bigger one) in this volumegroup. Format them wih ext3, mount them and copy some files to them.

3. Verify usage with fdisk, mount, pvs, vgs, lvs, pvdisplay, vgdisplay, lvdisplay and df. Does fdisk give you any information about lvm?

4. Enlarge the small logical volume by 50 percent, and verify your work!

5. Take a look at other commands that start with vg* , pv* or lv*.

6. Create a mirror and a striped Logical Volume.

7. Convert a linear logical volume to a mirror.

8. Convert a mirror logical volume to a linear.

9. Create a snapshot of a Logical Volume, take a backup of the snapshot. Then delete some files on the Logical Volume, then restore your backup.

10. Move your volume group to another disk (keep the Logical Volumes mounted).

11. If time permits, split a Volume Group with vgsplit, then merge it again with vgmerge.

Chapter 16. Booting Linux

16.1. Booting the system

Booting the system starts (once the hardware is powered on and configured) with a bootloader.

There are a variety of boot loaders available, most common on intel architecture is **GRUB**, which is replacing **Lilo** in many places. When installing Linux on SPARC architecture, you can choose **Silo**, Itanium systems can use **ELILO**, IBM S/390 and zSeries use **z/IPL** and PowerPC architectures use **YABOOT** (which means Yet Another boot loader).

Once the Linux kernel is loaded, the bootloader turns control over to it. From that moment on, the kernel is in control of the system. After discussing bootloaders, we continue with the init system that starts all the daemons.

16.2. GRUB

16.2.1. GRand Unified Bootloader

The most common bootloader on linux systems today is **grub**. On almost all intel based systems grub is replacing **lilo** (LInux LOader). Even Solaris switched to grub on x86 architecture.

One of the big advantages of rub over lilo is the capability to change the configuration during boot (by pressing e to edit the boot commandline).

16.2.2. menu.lst

Grub's configuration file is called **menu.lst** (old versions used **grub.conf**) and is located in /boot/grub. The screenshot below shows (part of) a typical grub configuration file.

```
paul@barry:~$ cat /boot/grub/menu.lst
default  0
timeout  5
color cyan/blue white/blue

title  Debian GNU/Linux, kernel 2.6.17-2-686
root  (hd0,0)
kernel  /boot/vmlinuz-2.6.17-2-686 root=/dev/hda1 ro
initrd  /boot/initrd.img-2.6.17-2-686
savedefault
boot

title  Debian GNU/Linux, kernel 2.6.15-1-686 (recovery mode)
root  (hd0,0)
```

```
kernel  /boot/vmlinuz-2.6.15-1-686 root=/dev/hda1 ro single
initrd  /boot/initrd.img-2.6.15-1-686
savedefault
boot

title  Debian GNU/Linux, kernel 2.6.15-1-686
root  (hd0,0)
kernel  /boot/vmlinuz-2.6.15-1-686 root=/dev/hda1 ro
initrd  /boot/initrd.img-2.6.15-1-686
savedefault
...
```

In the screenshot, you see some parameters at the top like **default** and **timeout**, followed by three **stanzas**. Each stanza is shown as a seperate choice on the bootmenu, and can be booted by grub by selecting it and pressing enter. The stanzas are automatically numbered, starting with 0. So when the default switch marks 0, then the first stanza will be booted, unless another choice is manually selected at boot time. Setting the default switch to 1 will boot the second stanza.

The **timeout** switch defines the amount of seconds a user has to make a choice in the bootmenu. In this example, the timeout is set to five seconds, so grub will wait five seconds before booting the first stanza.

Another interesting switch is **fallback**. This parameter allows you to set a backup stanza in case the first one fails.

16.2.3. Stanza commands

The **title** command serves as a description of the stanza that is visible in the bootmenu. It can be anything you like. In this example it describes the distro and kernel version.

```
title  Debian GNU/Linux, kernel 2.6.17-2-686
root  (hd0,0)
kernel  /boot/vmlinuz-2.6.17-2-686 root=/dev/hda1 ro
initrd  /boot/initrd.img-2.6.17-2-686
```

The **root** command will point to the hard disk to use. (hd0) is the first hard disk device, (hd1) is the second hard disk device. (hd0,0) is the first partition on the first disk, (hd0,1) is the second partition on that disk.

The **kernel** command point to the kernel(file) that grub needs to load. And **initrd** points to an initial RAM disk to accompany the kernel at system boot before mounting / .

16.2.4. Chainloading

Chainloading refers to grub loading another operating systems bootloader. The **chainloader** switch receives one opetion: the number of sectors to read and boot. For DOS one sector is enough, so a grub entry for a DOS or OS/2 partition might look like this. Note that MS-DOS requires the boot/root partition to be active!

```
title MS-DOS 6.22
root  (hd0,1)
makeactive
chainloader +1
```

16.2.5. Installing grub

Run the **grub-install** to install grub. The command requires a destination for overwriting the boot sector or mbr.

```
# grub-install /dev/hda
```

16.3. Lilo

16.3.1. Linux Loader

Lilo used to be the most used Linux bootloader, but is steadily being replaced in x86 with grub.

16.3.2. lilo.conf

Here is an example of a typical **lilo.conf** file. The **delay** switch receives a number in tenths of a second. So the delay below is three seconds, not thirty!

```
boot = /dev/hda
delay = 30

image = /boot/vmlinuz
  root = /dev/hda1
  label = Red Hat 5.2

image = /boot/vmlinuz
  root = /dev/hda2
  label = S.U.S.E. 8.0

other = /dev/hda4
  table = /dev/hda
  label = MS-DOS 6.22
```

The configration file shows three example stanzas. The first one boots Red Hat from the first partition on the first disk (hda1). The second stanza boots Suse 8.0 from the next partition. The last one loads MS-DOS.

16.4. Booting

The kernel receives system control from the bootloader. After a while the kernel starts the **init daemon**. The init daemon has **PID 1**. Many unix and linux systems use(d) init scripts to start daemons in the **System V release 4** style (explained in detail below).

But this synchronous (one after the other) method of starting daemons is slow, and although slow booting is not a problem on servers where uptime is measured in years, the recent uptake of linux on the desktop results in user complaints. To improve linux (and Solaris) startup speed, **Canonical** has developed **upstart** (first used in Ubuntu) and **Sun** has developed **Service Management Facility** for Solaris 10. Both systems are asynchronous and can replace the SysV init scripts. There is also an ongoing effort to create **initng** (init next generation).

16.5. Daemons

A **daemon** is a process that runs in background, without a link to a GUI or terminal. Daemons are usually started at system boot, and stay alive until the system shuts down. In more recent technical writings, daemons are often refered to as **services**.

Unix **daemons** are not to be confused with demons. Evi Nemeth, co-author of the UNIX System Administration Handbook has the following to say about daemons:

Many people equate the word "daemon" with the word "demon", implying some kind of satanic connection between UNIX and the underworld. This is an egregious misunderstanding. "Daemon" is actually a much older form of "demon"; daemons have no particular bias towards good or evil, but rather serve to help define a person's character or personality. The ancient Greeks' concept of a "personal daemon" was similar to the modern concept of a "guardian angel" ....

16.6. Init

16.6.1. /etc/inittab

After the kernel, **/sbin/init** is started with PID 1. Init will read its configuration file **/etc/inittab**. In that file, it will look for the value of initdefault (3 in the screenshot below).

```
[paul@rhel4 ~]$ grep ^id /etc/inittab
id:3:initdefault:
```

16.6.2. Runlevel

This number indicates the default **runlevel**. Some linuxes have a brief description of runlevels in /etc/inittab, like here on Red Hat Enterprise Linux 4.

```
# Default runlevel. The runlevels used by RHS are:
#   0 - halt (Do NOT set initdefault to this)
#   1 - Single user mode
#   2 - Multiuser, without NFS (The same as 3, if you don't have network)
#   3 - Full multiuser mode
#   4 - unused
#   5 - X11
#   6 - reboot (Do NOT set initdefault to this)
#
```

Runlevel 0 means the system is shutting down. Runlevel 1 is used for troubleshooting, only the root user can log on, and only at the console. Runlevel 3 is typical for servers, whereas runlevel 5 is typical for desktops (graphical logon). Besides runlevels 0, 1 and 6, the use may vary depending on the distribution. Some Debian and derived linux systems have full network and GUI logon on runlevels 2 to 5. So always verify the proper meaning of runlevels on your system.

16.6.3. sysinit

Independent of the runlevel, init will run the **/etc/rc.d/rc.sysinit** script (**/etc/init.d/rcS** on debian). This script does a lot of things : setting environment, populating /etc/mtab, mounting file systems, starting swap and more.

```
[paul ~]$ egrep -e"^# Ini" -e"^# Sta" -e"^# Che" /etc/rc.d/rc.sysinit
# Check SELinux status
# Initialize hardware
# Start the graphical boot, if necessary; /usr may not be mounted yet...
# Initialiaze ACPI bits
# Check filesystems
# Start the graphical boot, if necessary and not done yet.
# Check to see if SELinux requires a relabel
# Initialize pseudo-random number generator
# Start up swapping.
# Initialize the serial ports.
[paul ~]$
```

That **egrep** command could also have been written with **grep** like this : grep "^# \(Ini\|Sta\|Che\)". The screenshot above was made on Red Hat Enterprise Linux 4.

16.6.4. rc scripts

Init will continue to read /etc/inittab and meets this section on debian linux.

```
l0:0:wait:/etc/init.d/rc 0
l1:1:wait:/etc/init.d/rc 1
l2:2:wait:/etc/init.d/rc 2
l3:3:wait:/etc/init.d/rc 3
l4:4:wait:/etc/init.d/rc 4
l5:5:wait:/etc/init.d/rc 5
l6:6:wait:/etc/init.d/rc 6
```

(on Red Hat Enterprise Linux it is identical except init.d is rc.d.

```
l0:0:wait:/etc/rc.d/rc 0
l1:1:wait:/etc/rc.d/rc 1
```

```
l2:2:wait:/etc/rc.d/rc 2
l3:3:wait:/etc/rc.d/rc 3
l4:4:wait:/etc/rc.d/rc 4
l5:5:wait:/etc/rc.d/rc 5
l6:6:wait:/etc/rc.d/rc 6
```

In both cases, this means that init will start the rc script with as only parameter the runlevel. Actually /etc/inittab has fields seperated by colons. The second field determines the runlevel in which this line should be executed. So in both cases, only one line of the seven will be executed, depending on the runlevel set by initdefault.

When you take a look in the relevant **/etc/rc3.d** directory, which is real on debian and a symbolic link to **/etc/rc.d/rc3.d** on Red Hat, then you will see a lot of (links to) scripts who's name start with either uppercase K or uppercase S. When entering a runlevel, scripts with uppercase S are started in alphabetical order with "start" as the only parameter. When leaving a runlevel, the same happens for scripts starting with K. All this is done by the rc script.

16.6.5. Power and Ctrl-Alt-Del

When rc is finished starting all those scripts, init will continue to read /etc/inittab. It will read commands on what to execute in case of **powerfailure**, powerok and **Ctrl-Alt-Delete**. The init process never stops keeping an eye on power failures and that triple key combo.

The relevant part on Red Hat Enterprise Linux.

```
[paul@RHEL4b ~]$ grep "\(^c\|^p\)" /etc/inittab
ca::ctrlaltdel:/sbin/shutdown -t3 -r now
pf::powerfail:/sbin/shutdown -f -h +2 "PowerFailure;System Shutting Down"
pr:12345:powerokwait:/sbin/shutdown -c "PowerRestored;Shutdown Cancelled"
```

And very similar on Debian Etch.

```
paul@barry:~$ grep "\(^c\|^p\)" /etc/inittab
ca:12345:ctrlaltdel:/sbin/shutdown -t1 -a -r now
pf::powerwait:/etc/init.d/powerfail start
pn::powerfailnow:/etc/init.d/powerfail now
po::powerokwait:/etc/init.d/powerfail stop
```

16.6.6. getty

Almost at the end of /etc/inittab, there is a section to start and **respawn** several mingetty's.

```
[root@RHEL4b ~]# grep getty /etc/inittab
# Run gettys in standard runlevels
1:2345:respawn:/sbin/mingetty tty1
2:2345:respawn:/sbin/mingetty tty2
```

```
3:2345:respawn:/sbin/mingetty tty3
4:2345:respawn:/sbin/mingetty tty4
5:2345:respawn:/sbin/mingetty tty5
6:2345:respawn:/sbin/mingetty tty6
[root@RHEL4b ~]#
```

A mingetty will display a message on a virtual console and allow you to type a userid and sends that info to the login program. The login program will verify whether that user exists in /etc/passwd and prompt for (and verify) a password. If the password is correct, login passes control to the shell listed in /etc/passwd.

So the getty's are started by init, and watched until they die (user exit's the shell and is logged out). When this happens, the init daemon will respawn a new mingetty. So even if you kill the mingetty's, they will be reborn automatically.

```
[root@RHEL4b ~]# ps fax |grep mingetty
 3038 tty1     Ss+    0:00 /sbin/mingetty tty1
 3039 tty2     Ss+    0:00 /sbin/mingetty tty2
 3040 tty3     Ss+    0:00 /sbin/mingetty tty3
 3041 tty4     Ss+    0:00 /sbin/mingetty tty4
 3042 tty5     Ss+    0:00 /sbin/mingetty tty5
 3043 tty6     Ss+    0:00 /sbin/mingetty tty6
[root@RHEL4b ~]# kill 3038 3039 3040 3041 3042 3043
[root@RHEL4b ~]# ps fax |grep mingetty
 4774 tty1     Ss+    0:00 /sbin/mingetty tty1
 4884 tty2     Ss+    0:00 /sbin/mingetty tty2
 4974 tty3     Ss+    0:00 /sbin/mingetty tty3
 5026 tty4     Ss+    0:00 /sbin/mingetty tty4
 5073 tty5     Ss+    0:00 /sbin/mingetty tty5
 5098 tty6     Ss+    0:00 /sbin/mingetty tty6
[root@RHEL4b ~]#
```

You can disable a mingetty for a certain tty by removing the runlevel from the second field in its line in /etc/inittab. Don't forget to tell init about the change of its configuration file with **kill -1 1**.

16.7. Starting and stopping daemons

The K and S scripts usually are links to the real scripts in **/etc/init.d** or **/etc/rc.d/init.d**. These can also be used when the system is running to start and stop daemons (or services). Most of them accept the following parameters: start, stop, restart, status.

```
root@laika:~# /etc/init.d/samba restart
 * Stopping Samba daemons...                              [ OK ]
 * Starting Samba daemons...                              [ OK ]
root@laika:~#
```

You can achieve the same result on Red Hat and derived linuxes with the **service** command.

```
[root@RHEL4b ~]# service smb restart
```

```
Shutting down SMB services:                                [  OK  ]
Shutting down NMB services:                                [  OK  ]
Starting SMB services:                                     [  OK  ]
Starting NMB services:                                     [  OK  ]
[root@RHEL4b ~]#
```

16.8. Display the runlevel

You can see your current runlevel with the **runlevel** or **who -r** commands.

The runlevel command is typical linux and will output the previous and the current runlevel. If there was no previous runlevel, then it will mark it with the letter N.

```
[root@RHEL4b ~]# runlevel
N 3
```

The history of who -r dates back to older unixes, and it still works on linux.

```
[root@RHEL4b ~]# who -r
         run-level 3  Jul 28 09:15                   last=S
```

16.9. Changing the runlevel

You can switch to another runlevel with the **telinit** command. On Linux **/sbin/telinit** is usually a hard link to /sbin/init.

16.10. more info

You might also want to take a look at **chkconfig**, **update-rc.d**, **shutdown**, **poweroff** and passing **init=/bin/bash** to the kernel.

16.11. Practice

1. Take a copy of the kernel, initrd and System.map files in /boot in /boot, naming them 3.0 instead of 2.6. Then add a stanza in grub for this 3.0 files.

2. Change /etc/inittab so that only two mingetty's are respawned. Kill the other mingetty's and verify that they don't come back.

3. Use the Red Hat Enterprise Linux virtual machine. Go to runlevel 5, display the current and previous runlevel, then go back to runlevel 3.

4. Is the sysinit script on your computers setting or changing the PATH environment variable ?

5. Write a script that acts like a daemon script in /etc/init.d/. It should have a case statement to act on start/stop/restart and status. Test the script!

6. Have your script started automatically in runlevel 3, test that it works. If it works, also try stopping it in a runlevel.

7. If time permits, use chkconfig to setup your script in runlevels 2 and 3.

16.12. Solutions

1. The files should look like in this screenshot, verify the names in /boot. You could do a reboot and test your stanza.

```
[root@RHEL5 ~]# grep 3.0.18 /boot/grub/menu.lst
title Red Hat Enterprise Linux Server (3.0.18)
 kernel /vmlinuz-3.0.18 ro root=/dev/VolGroup00/LogVol00 rhgb quiet
 initrd /initrd-3.0.18.img
[root@RHEL5 ~]#
```

2. Killing the mingetty's will result in init respawning them. You can edit /etc/inittab so it looks like the screenshot below. Don't forget to also run kill -1 1.

```
[root@RHEL5 ~]# grep tty /etc/inittab
# Run gettys in standard runlevels
1:2345:respawn:/sbin/mingetty tty1
2:2345:respawn:/sbin/mingetty tty2
3:2:respawn:/sbin/mingetty tty3
4:2:respawn:/sbin/mingetty tty4
5:2:respawn:/sbin/mingetty tty5
6:2:respawn:/sbin/mingetty tty6
[root@RHEL5 ~]#
```

3. use the telinit (or init) and runlevel commands

4. On Red Hat, grep for PATH in /etc/rc.sysinit, on Debian/Ubuntu check /etc/rc.local. The answer is probably no, but on RHEL5 the rc.sysinit script does set the HOSTNAME variable.

```
[root@RHEL5 etc]# grep HOSTNAME rc.sysinit
```

5. The script could look something like this.

```
#!/bin/bash
# /etc/init.d/pold
#

# always runs
touch /var/lock/pold
```

```
case "$1" in
  start)
    echo -n "Starting pold..."
    sleep 1;
    echo "done."
    ;;
  stop)
    echo -n "Stopping pold..."
    sleep 1;
    echo "done."
    ;;
  *)
    echo "Usage: /etc/init.d/pold {start|stop}"
    exit 1
    ;;
esac

exit 0
```

Chapter 17. Linux Kernel

17.1. about the Linux kernel

17.1.1. kernel versions

In 1991 Linux Torvalds wrote (the first version of) the Linux kernel. He put it online, and other people started contributing code. Over 4000 individuals contributed source code to the latest kernel release (version 2.6.27 in November 2008).

Major Linux kernel versions used to come in even and odd numbers. Versions **2.0**, **2.2**, **2.4** and **2.6** are considered stable kernel versions. Whereas **2.1**, **2.3** and **2.5** were unstable (read development) versions. Since the release of 2.6.0 in January 2004, all development has been done in the 2.6 tree. There is currently no v2.7.x and according to Linus the even/stable vs odd/development scheme is abandoned forever.

17.1.2. uname -r

To see your current Linux kernel version, issue the **uname -r** command as shown below.

This first example shows Linux major version **2.6** and minor version **24**. The rest **-22-generic** is specific to the distribution (Ubuntu in this case).

```
paul@laika:~$ uname -r
2.6.24-22-generic
```

The same command on Red Hat Enterprise Linux shows an older kernel (2.6.18) with **-92.1.17.el5** being specific to the distribution.

```
[paul@RHEL52 ~]$ uname -r
2.6.18-92.1.17.el5
```

17.2. Linux kernel source

17.2.1. ftp.kernel.org

The home of the Linux kernel source is **ftp.kernel.org**. It contains all official releases of the Linux kernel source code from 1991. It provides free downloads over http, ftp and rsync of all these releases, as well as changelogs and patches. More information can be otained on the website **www.kernel.org**.

Anyone can anonymously use an ftp client to access ftp.kernel.org

```
paul@laika:~$ ftp ftp.kernel.org
Connected to pub3.kernel.org.
220 Welcome to ftp.kernel.org.
Name (ftp.kernel.org:paul): anonymous
331 Please specify the password.
Password:
230-       Welcome to the
230-
230-   LINUX KERNEL ARCHIVES
230-       ftp.kernel.org
```

All the Linux kernel versions are located in the pub/linux/kernel/ directory.

```
ftp> ls pub/linux/kernel/v*
200 PORT command successful. Consider using PASV.
150 Here comes the directory listing.
drwxrwsr-x    2 536      536          4096 Mar 20  2003 v1.0
drwxrwsr-x    2 536      536         20480 Mar 20  2003 v1.1
drwxrwsr-x    2 536      536          8192 Mar 20  2003 v1.2
drwxrwsr-x    2 536      536         40960 Mar 20  2003 v1.3
drwxrwsr-x    3 536      536         16384 Feb 08  2004 v2.0
drwxrwsr-x    2 536      536         53248 Mar 20  2003 v2.1
drwxrwsr-x    3 536      536         12288 Mar 24  2004 v2.2
drwxrwsr-x    2 536      536         24576 Mar 20  2003 v2.3
drwxrwsr-x    5 536      536         28672 Dec 02 08:14 v2.4
drwxrwsr-x    4 536      536         32768 Jul 14  2003 v2.5
drwxrwsr-x    7 536      536        110592 Dec 05 22:36 v2.6
226 Directory send OK.
ftp>
```

17.2.2. /usr/src

On your local computer, the kernel source is located in **/usr/src**. Note though that the structure inside /usr/src might be different depending on the distribution that you are using.

First let's take a look at **/usr/src on Debian**. There appear to be two versions of the complete Linux source code there. Looking for a specific file (e1000_main.c) with find reveals it's exact location.

```
paul@barry:~$ ls -l /usr/src/
drwxr-xr-x 20 root root     4096 2006-04-04 22:12 linux-source-2.6.15
drwxr-xr-x 19 root root     4096 2006-07-15 17:32 linux-source-2.6.16
paul@barry:~$ find /usr/src -name e1000_main.c
/usr/src/linux-source-2.6.15/drivers/net/e1000/e1000_main.c
/usr/src/linux-source-2.6.16/drivers/net/e1000/e1000_main.c
```

This is very similar to **/usr/src on Ubuntu**, except there is only one kernel here (and it is newer).

```
paul@laika:~$ ls -l /usr/src/
drwxr-xr-x 23 root root     4096 2008-11-24 23:28 linux-source-2.6.24
paul@laika:~$ find /usr/src -name "e1000_main.c"
```

```
/usr/src/linux-source-2.6.24/drivers/net/e1000/e1000_main.c
```

Now take a look at **/usr/src on Red Hat Enterprise Linux**.

```
[paul@RHEL52 ~]$ ls -l /usr/src/
drwxr-xr-x 5 root root 4096 Dec  5 19:23 kernels
drwxr-xr-x 7 root root 4096 Oct 11 13:22 redhat
```

We will have to dig a little deeper to find the kernel source on Red Hat!

```
[paul@RHEL52 ~]$ cd /usr/src/redhat/BUILD/
[paul@RHEL52 BUILD]$ find . -name "e1000_main.c"
./kernel-2.6.18/linux-2.6.18.i686/drivers/net/e1000/e1000_main.c
```

17.2.3. downloading the kernel source

17.2.3.1. Debian

Installing the kernel source on Debian is really simple with **aptitude install linux-source**. You can do a search for all linux-source packeges first, like in this screenshot.

```
root@barry:~# aptitude search linux-source
v   linux-source          -
v   linux-source-2.6      -
id  linux-source-2.6.15    - Linux kernel source for version 2.6.15
i   linux-source-2.6.16    - Linux kernel source for version 2.6.16
p   linux-source-2.6.18    - Linux kernel source for version 2.6.18
p   linux-source-2.6.24    - Linux kernel source for version 2.6.24
```

And then use **aptitude install** to download and install the Debian Linux kernel source code.

```
root@barry:~# aptitude install linux-source-2.6.24
```

When the aptitude is finished, you will see a new file named **/usr/src/linux-source-<version>.tar.bz2**

```
root@barry:/usr/src# ls -lh
drwxr-xr-x 20 root root 4.0K 2006-04-04 22:12 linux-source-2.6.15
drwxr-xr-x 19 root root 4.0K 2006-07-15 17:32 linux-source-2.6.16
-rw-r--r--  1 root root  45M 2008-12-02 10:56 linux-source-2.6.24.tar.bz2
```

17.2.3.2. Ubuntu

Ubuntu is based on Debian and also uses **aptitude**, so the task is very similar.

```
root@laika:~# aptitude search linux-source
i   linux-source           - Linux kernel source with Ubuntu patches
v   linux-source-2.6       -
i A linux-source-2.6.24    - Linux kernel source for version 2.6.24
root@laika:~# aptitude install linux-source
```

And when aptitude finishes, we end up with a **/usr/src/linux-source-<version>.tar.bz** file.

```
oot@laika:~# ll /usr/src
total 45M
-rw-r--r--  1 root root  45M 2008-11-24 23:30 linux-source-2.6.24.tar.bz2
```

17.2.3.3. Red Hat Enterprise Linux

The Red Hat kernel source is located on the fourth source cdrom. The file is called **kernel-2.6.9-42.EL.src.rpm** (example for RHELv4u4). It is also available online at ftp://ftp.redhat.com/pub/redhat/linux/enterprise/5Server/en/os/SRPMS/ (example for RHEL5).

To download the kernel source on RHEL, use this long wget command (on one line, without the trailing \).

```
wget ftp://ftp.redhat.com/pub/redhat/linux/enterprise/5Server/en/os/\
SRPMS/kernel-`uname -r`.src.rpm
```

When the wget download is finished, you end up with a 60M .rpm file.

```
[root@RHEL52 src]# ll
total 60M
-rw-r--r-- 1 root root  60M Dec  5 20:54 kernel-2.6.18-92.1.17.el5.src.rpm
drwxr-xr-x 5 root root 4.0K Dec  5 19:23 kernels
drwxr-xr-x 7 root root 4.0K Oct 11 13:22 redhat
```

We will need to perform some more steps before this can be used as kernel source code.

First, we issue the **rpm -i kernel-2.6.9-42.EL.src.rpm** command to install this Red Hat package.

```
[root@RHEL52 src]# ll
total 60M
-rw-r--r-- 1 root root  60M Dec  5 20:54 kernel-2.6.18-92.1.17.el5.src.rpm
drwxr-xr-x 5 root root 4.0K Dec  5 19:23 kernels
drwxr-xr-x 7 root root 4.0K Oct 11 13:22 redhat
[root@RHEL52 src]# rpm -i kernel-2.6.18-92.1.17.el5.src.rpm
```

The we move to the SPECS directory and perform an **rpmbuild**.

```
[root@RHEL52 ~]# cd /usr/src/redhat/SPECS
[root@RHEL52 SPECS]# rpmbuild -bp -vv --target=i686 kernel-2.6.spec
```

The rpmbuild command put the RHEL Linux kernel source code in **/usr/src/redhat/BUILD/kernel-<version>/**.

```
[root@RHEL52 kernel-2.6.18]# pwd
/usr/src/redhat/BUILD/kernel-2.6.18
[root@RHEL52 kernel-2.6.18]# ll
total 20K
drwxr-xr-x  2 root root 4.0K Dec  6  2007 config
-rw-r--r--  1 root root 3.1K Dec  5 20:58 Config.mk
drwxr-xr-x 20 root root 4.0K Dec  5 20:58 linux-2.6.18.i686
drwxr-xr-x 19 root root 4.0K Sep 20  2006 vanilla
drwxr-xr-x  8 root root 4.0K Dec  6  2007 xen
```

17.3. kernel boot files

17.3.1. vmlinuz

The **vmlinuz** file in /boot is the compressed kernel.

```
paul@barry:~$ ls -lh /boot | grep vmlinuz
-rw-r--r-- 1 root root 1.2M 2006-03-06 16:22 vmlinuz-2.6.15-1-486
-rw-r--r-- 1 root root 1.1M 2006-03-06 16:30 vmlinuz-2.6.15-1-686
-rw-r--r-- 1 root root 1.3M 2008-02-11 00:00 vmlinuz-2.6.18-6-686
paul@barry:~$
```

17.3.2. initrd

The kernel uses **initrd** (an initial RAM disk) at boot time. The initrd is mounted before the kernel loads, and can contain additional drivers and modules. It is a **compressed cpio archive**, so you can look at the contents in this way.

```
root@RHELv4u4:/boot# mkdir /mnt/initrd
root@RHELv4u4:/boot# cp initrd-2.6.9-42.0.3.EL.img TMPinitrd.gz
root@RHELv4u4:/boot# gunzip TMPinitrd.gz
root@RHELv4u4:/boot# file TMPinitrd
TMPinitrd: ASCII cpio archive (SVR4 with no CRC)
root@RHELv4u4:/boot# cd /mnt/initrd/
root@RHELv4u4:/mnt/initrd# cpio -i | /boot/TMPinitrd
4985 blocks
root@RHELv4u4:/mnt/initrd# ls -l
total 76
drwxr-xr-x  2 root root 4096 Feb  5 08:36 bin
drwxr-xr-x  2 root root 4096 Feb  5 08:36 dev
drwxr-xr-x  4 root root 4096 Feb  5 08:36 etc
-rwxr-xr-x  1 root root 1607 Feb  5 08:36 init
drwxr-xr-x  2 root root 4096 Feb  5 08:36 lib
```

```
drwxr-xr-x  2 root root 4096 Feb  5 08:36 loopfs
drwxr-xr-x  2 root root 4096 Feb  5 08:36 proc
lrwxrwxrwx  1 root root    3 Feb  5 08:36 sbin -> bin
drwxr-xr-x  2 root root 4096 Feb  5 08:36 sys
drwxr-xr-x  2 root root 4096 Feb  5 08:36 sysroot
root@RHELv4u4:/mnt/initrd#
```

17.3.3. System.map

The **System.map** contains the symbol table and changes with every kernel compile. The symbol table is also present in **/proc/kallsyms** (pre 2.6 kernels name this file / proc/ksyms).

```
root@RHELv4u4:/boot# head System.map-`uname -r`
00000400 A __kernel_vsyscall
0000041a A SYSENTER_RETURN_OFFSET
00000420 A __kernel_sigreturn
00000440 A __kernel_rt_sigreturn
c0100000 A _text
c0100000 T startup_32
c01000c6 t checkCPUtype
c0100147 t is486
c010014e t is386
c010019f t L6
root@RHELv4u4:/boot# head /proc/kallsyms
c0100228 t _stext
c0100228 t calibrate_delay_direct
c0100228 t stext
c0100337 t calibrate_delay
c01004db t rest_init
c0100580 t do_pre_smp_initcalls
c0100585 t run_init_process
c01005ac t init
c0100789 t early_param_test
c01007ad t early_setup_test
root@RHELv4u4:/boot#
```

17.3.4. .config

The last file copied to the /boot directory is the kernel configuration used for compilation. This file is not necessary in the /boot directory, but it is common practice to put a copy there. It allows you to recompile a kernel, starting from the same configuration as an existing working one.

17.4. Linux kernel modules

17.4.1. about kernel modules

The Linux kernel is a monolithic kernel with loadable modules. These modules contain parts of the kernel used typically for device drivers, file systems and network protocols. Most of the time the necesarry kernel modules are loaded automatically and dynamically without administrator interaction.

17.4.2. /lib/modules

The modules are stored in the **/lib/modules/<kernel-version>** directory. There is a seperate directory for each kernel that was compiled for your system.

```
paul@laika:~$ ll /lib/modules/
total 12K
drwxr-xr-x 7 root root 4.0K 2008-11-10 14:32 2.6.24-16-generic
drwxr-xr-x 8 root root 4.0K 2008-12-06 15:39 2.6.24-21-generic
drwxr-xr-x 8 root root 4.0K 2008-12-05 12:58 2.6.24-22-generic
```

17.4.3. <module>.ko

The file containing the modules usually ends in **.ko**. This screenshot shows the location of the isdn module files.

```
paul@laika:~$ find /lib/modules -name isdn.ko
/lib/modules/2.6.24-21-generic/kernel/drivers/isdn/i4l/isdn.ko
/lib/modules/2.6.24-22-generic/kernel/drivers/isdn/i4l/isdn.ko
/lib/modules/2.6.24-16-generic/kernel/drivers/isdn/i4l/isdn.ko
```

17.4.4. lsmod

To see a list of currently loaded modules, use **lsmod**. You see the name of each loaded module, the size, the use count, and the names of other modules using this one.

```
[root@RHEL52 ~]# lsmod | head -5
Module                  Size  Used by
autofs4                24517  2
hidp                   23105  2
rfcomm                 42457  0
l2cap                  29505  10 hidp,rfcomm
```

17.4.5. /proc/modules

Naturally, the same information is present in **/proc/modules**. Actually **lsmod** only reads and reformats the output of /proc/modules.

```
[root@RHEL52 ~]# head -5 /proc/modules
autofs4 24517 2 - Live 0xe09bd000
hidp 23105 2 - Live 0xe09d3000
rfcomm 42457 0 - Live 0xe0ac1000
```

```
l2cap 29505 10 hidp,rfcomm, Live 0xe0ab8000
bluetooth 53797 5 hidp,rfcomm,l2cap, Live 0xe09e4000
```

17.4.6. insmod

Kernel modules can be manually loaded with the **insmod** command. This is a very simple (and obsolete) way of loading modules. The screenshot shows **insmod** loading the fat module (for fat file system support).

```
root@barry:/lib/modules/2.6.17-2-686# pwd
/lib/modules/2.6.17-2-686
root@barry:/lib/modules/2.6.17-2-686# lsmod | grep fat
root@barry:/lib/modules/2.6.17-2-686# insmod kernel/fs/fat/fat.ko
root@barry:/lib/modules/2.6.17-2-686# lsmod | grep fat
fat                    46588  0
```

insmod is not detecting dependencies, so it fails to load the isdn module (because the isdn module depends on the slhc module).

```
[root@RHEL52 drivers]# pwd
/lib/modules/2.6.18-92.1.18.el5/kernel/drivers
[root@RHEL52 kernel]# insmod isdn/i4l/isdn.ko
insmod: error inserting 'isdn/i4l/isdn.ko': -1 Unknown symbol in module
```

17.4.7. modinfo

As you can see in the screenshot of **modinfo** below, the isdn module depends in the slhc module.

```
[root@RHEL52 drivers]# modinfo isdn/i4l/isdn.ko | head -6
filename:       isdn/i4l/isdn.ko
license:        GPL
author:         Fritz Elfert
description:    ISDN4Linux: link layer
srcversion:     99650346E708173496F6739
depends:        slhc
```

17.4.8. modprobe

The big advantage of **modprobe** over **insmod** is that modprobe will load all necessary modules, whereas insmod requires manual loading of depedencies. Another advantage is that you don't need to point to the filename with full path.

This screenshot shows how modprobe loads the isdn module, automatically loading slhc in background.

```
[root@RHEL52 kernel]# lsmod | grep isdn
[root@RHEL52 kernel]# modprobe isdn
[root@RHEL52 kernel]# lsmod | grep isdn
isdn                  122433  0
slhc                   10561  1 isdn
[root@RHEL52 kernel]#
```

17.4.9. /lib/modules/<kernel>/modules.dep

Module dependencies are stored in **modules.dep**.

```
[root@RHEL52 2.6.18-92.1.18.el5]# pwd
/lib/modules/2.6.18-92.1.18.el5
[root@RHEL52 2.6.18-92.1.18.el5]# head -3 modules.dep
/lib/modules/2.6.18-92.1.18.el5/kernel/drivers/net/tokenring/3c359.ko:
/lib/modules/2.6.18-92.1.18.el5/kernel/drivers/net/pcmcia/3c574_cs.ko:
/lib/modules/2.6.18-92.1.18.el5/kernel/drivers/net/pcmcia/3c589_cs.ko:
```

17.4.10. depmod

The **modules.dep** file can be updated (recreated) with the **depmod** command. In this screenshot no modules were added, so **depmod** generates the same file.

```
root@barry:/lib/modules/2.6.17-2-686# ls -l modules.dep
-rw-r--r-- 1 root root 310676 2008-03-01 16:32 modules.dep
root@barry:/lib/modules/2.6.17-2-686# depmod
root@barry:/lib/modules/2.6.17-2-686# ls -l modules.dep
-rw-r--r-- 1 root root 310676 2008-12-07 13:54 modules.dep
```

17.4.11. rmmod

Similar to insmod, the **rmmod** command is rarely used anymore.

```
[root@RHELv4u3 ~]# modprobe isdn
[root@RHELv4u3 ~]# rmmod slhc
ERROR: Module slhc is in use by isdn
[root@RHELv4u3 ~]# rmmod isdn
[root@RHELv4u3 ~]# rmmod slhc
[root@RHELv4u3 ~]# lsmod | grep isdn
[root@RHELv4u3 ~]#
```

17.4.12. modprobe -r

Contrary to rmmod, **modprobe** will automatically remove unneeded modules.

```
[root@RHELv4u3 ~]# modprobe isdn
```

```
[root@RHELv4u3 ~]# lsmod | grep isdn
isdn                  133537  0
slhc                    7233  1 isdn
[root@RHELv4u3 ~]# modprobe -r isdn
[root@RHELv4u3 ~]# lsmod | grep isdn
[root@RHELv4u3 ~]# lsmod | grep slhc
[root@RHELv4u3 ~]#
```

17.4.13. /etc/modprobe.conf

The **/etc/modprobe.conf** file and the **/etc/modprobe.d** directory can contain aliases (used by humans) and options (for dependent modules) for modprobe.

```
[root@RHEL52 ~]# cat /etc/modprobe.conf
alias scsi_hostadapter mptbase
alias scsi_hostadapter1 mptspi
alias scsi_hostadapter2 ata_piix
alias eth0 pcnet32
alias eth2 pcnet32
alias eth1 pcnet32
```

17.5. compiling a kernel

17.5.1. extraversion

Enter into **/usr/src/redhat/BUILD/kernel-2.6.9/linux-2.6.9/** and change the **extraversion** in the Makefile.

```
[root@RHEL52 linux-2.6.18.i686]# pwd
/usr/src/redhat/BUILD/kernel-2.6.18/linux-2.6.18.i686
[root@RHEL52 linux-2.6.18.i686]# vi Makefile
[root@RHEL52 linux-2.6.18.i686]# head -4 Makefile
VERSION = 2
PATCHLEVEL = 6
SUBLEVEL = 18
EXTRAVERSION = -paul2008
```

17.5.2. make mrproper

Now clean up the source from any previous installs with **make mrproper**. If this is your first after downloading the source code, then this is not needed.

```
[root@RHEL52 linux-2.6.18.i686]# make mrproper
  CLEAN   scripts/basic
  CLEAN   scripts/kconfig
  CLEAN   include/config
  CLEAN   .config .config.old
```

17.5.3. .config

Now copy a working **.config** from /boot to our kernel directory. This file contains the configuration that was used for your current working kernel. It determines whether modules are included in compilation or not.

```
[root@RHEL52 linux-2.6.18.i686]# cp /boot/config-2.6.18-92.1.18.el5 .config
```

17.5.4. make menuconfig

Now run **make menuconfig** (or the graphical **make xconfig**). This tool allows you to select whether to compile stuff as a module (m), as part of the kernel (*), or not at all (smaller kernel size). If you remove too much, your kernel will not work. The configuration will be stored in the hidden **.config** file.

```
[root@RHEL52 linux-2.6.18.i686]# make menuconfig
```

17.5.5. make clean

Issue a **make clean** to prepare the kernel for compile. **make clean** will remove most generated files, but keeps your kernel configuration. Running a **make mrproper** at this point would destroy the .config file that you built with **make menuconfig**.

```
[root@RHEL52 linux-2.6.18.i686]# make clean
```

17.5.6. make bzImage

And then run **make bzImage**, sit back and relax while the kernel compiles. You can use **time make bzImage** to know how long it takes to compile, so next time you can go for a short walk.

```
[root@RHEL52 linux-2.6.18.i686]# time make bzImage
  HOSTCC  scripts/basic/fixdep
  HOSTCC  scripts/basic/docproc
  HOSTCC  scripts/kconfig/conf.o
  HOSTCC  scripts/kconfig/kxgettext.o
...
```

This command will end with telling you the location of the **bzImage** file (and with time info if you also specified the time command.

```
Kernel: arch/i386/boot/bzImage is ready  (#1)
```

```
real 13m59.573s
user 1m22.631s
sys 11m51.034s
[root@RHEL52 linux-2.6.18.i686]#
```

You can already copy this image to /boot with **cp arch/i386/boot/bzImage /boot/vmlinuz-<kernel-version>**.

17.5.7. make modules

Now run **make modules**. It can take 20 to 50 minutes to compile all the modules.

```
[root@RHEL52 linux-2.6.18.i686]# time make modules
  CHK     include/linux/version.h
  CHK     include/linux/utsrelease.h
  CC [M]  arch/i386/kernel/msr.o
  CC [M]  arch/i386/kernel/cpuid.o
  CC [M]  arch/i386/kernel/microcode.o
```

17.5.8. make modules_install

To copy all the compiled modules to **/lib/modules** just run **make modules_install** (takes about 20 seconds). Here's a screenshot from before the command.

```
[root@RHEL52 linux-2.6.18.i686]# ls -l /lib/modules/
total 20
drwxr-xr-x 6 root root 4096 Oct 15 13:09 2.6.18-92.1.13.el5
drwxr-xr-x 6 root root 4096 Nov 11 08:51 2.6.18-92.1.17.el5
drwxr-xr-x 6 root root 4096 Dec  6 07:11 2.6.18-92.1.18.el5
[root@RHEL52 linux-2.6.18.i686]# make modules_install
```

And here is the same directory after. Notice that **make modules_install** created a new directory for the new kernel.

```
[root@RHEL52 linux-2.6.18.i686]# ls -l /lib/modules/
total 24
drwxr-xr-x 6 root root 4096 Oct 15 13:09 2.6.18-92.1.13.el5
drwxr-xr-x 6 root root 4096 Nov 11 08:51 2.6.18-92.1.17.el5
drwxr-xr-x 6 root root 4096 Dec  6 07:11 2.6.18-92.1.18.el5
drwxr-xr-x 3 root root 4096 Dec  6 08:50 2.6.18-paul2008
```

17.5.9. /boot

We still need to copy the kernel, the System.map and our configuration file to /boot. Strictly speaking the .config file is not obligatory, but it might help you in future compilations of the kernel.

```
[root@RHEL52 ]# pwd
/usr/src/redhat/BUILD/kernel-2.6.18/linux-2.6.18.i686
[root@RHEL52 ]# cp System.map /boot/System.map-2.6.18-paul2008
[root@RHEL52 ]# cp .config /boot/config-2.6.18-paul2008
[root@RHEL52 ]# cp arch/i386/boot/bzImage /boot/vmlinuz-2.6.18-paul2008
```

17.5.10. mkinitrd

The kernel often uses an initrd file at bootup. We can use **mkinitrd** to generate this file. Make sure you use the correct kernel name!

```
[root@RHEL52 ]# pwd
/usr/src/redhat/BUILD/kernel-2.6.18/linux-2.6.18.i686
[root@RHEL52 ]# mkinitrd /boot/initrd-2.6.18-paul2008 2.6.18-paul2008
```

17.5.11. bootloader

Compilation is now finished, don't forget to create an additional stanza in grub or lilo.

17.6. compiling one module

17.6.1. hello.c

A little C program that will be our module.

```
[root@rhel4a kernel_module]# cat hello.c
#include <linux/module.h>
#include <section>

int init_module(void)
{
 printk(KERN_INFO "Start Hello World...\n");
 return 0;
}

void cleanup_module(void)
{
 printk(KERN_INFO "End Hello World... \n");
}
```

17.6.2. Makefile

The make file for this module.

```
[root@rhel4a kernel_module]# cat Makefile
obj-m += hello.o
```

```
all:
make -C /lib/modules/$(shell uname -r)/build M=$(PWD) modules
clean:
make -C /lib/modules/$(shell uname -r)/build M=$(PWD) clean
```

These are the only two files needed.

```
[root@rhel4a kernel_module]# ll
total 16
-rw-rw-r--  1 paul paul 250 Feb 15 19:14 hello.c
-rw-rw-r--  1 paul paul 153 Feb 15 19:15 Makefile
```

17.6.3. make

The running of the **make** command.

```
[root@rhel4a kernel_module]# make
make -C /lib/modules/2.6.9-paul-2/build M=~/kernel_module modules
make[1]: Entering dir... `/usr/src/redhat/BUILD/kernel-2.6.9/linux-2.6.9'
CC [M]  /home/paul/kernel_module/hello.o
Building modules, stage 2.
MODPOST
CC      /home/paul/kernel_module/hello.mod.o
LD [M]  /home/paul/kernel_module/hello.ko
make[1]: Leaving dir... `/usr/src/redhat/BUILD/kernel-2.6.9/linux-2.6.9'
[root@rhel4a kernel_module]#
```

Now we have more files.

```
[root@rhel4a kernel_module]# ll
total 172
-rw-rw-r--  1 paul paul   250 Feb 15 19:14 hello.c
-rw-r--r--  1 root root 64475 Feb 15 19:15 hello.ko
-rw-r--r--  1 root root   632 Feb 15 19:15 hello.mod.c
-rw-r--r--  1 root root 37036 Feb 15 19:15 hello.mod.o
-rw-r--r--  1 root root 28396 Feb 15 19:15 hello.o
-rw-rw-r--  1 paul paul   153 Feb 15 19:15 Makefile
[root@rhel4a kernel_module]#
```

17.6.4. hello.ko

Use **modinfo** to verify that it is really a module.

```
[root@rhel4a kernel_module]# modinfo hello.ko
filename:       hello.ko
vermagic:       2.6.9-paul-2 SMP 686 REGPARM 4KSTACKS gcc-3.4
depends:
[root@rhel4a kernel_module]#
```

Good, so now we can load our hello module.

```
[root@rhel4a kernel_module]# lsmod | grep hello
[root@rhel4a kernel_module]# insmod ./hello.ko
[root@rhel4a kernel_module]# lsmod | grep hello
hello                   5504  0
[root@rhel4a kernel_module]# tail -1 /var/log/messages
Feb 15 19:16:07 rhel4a kernel: Start Hello World...
[root@rhel4a kernel_module]# rmmod hello
[root@rhel4a kernel_module]#
```

Finally **/var/log/messages** has a little surprise.

```
[root@rhel4a kernel_module]# tail -2 /var/log/messages
Feb 15 19:16:07 rhel4a kernel: Start Hello World...
Feb 15 19:16:35 rhel4a kernel: End Hello World...
[root@rhel4a kernel_module]#
```

Chapter 18. Introduction to Networking

18.1. About TCP/IP

18.1.1. Overview of tcp/ip v4

The unicast **Internet Protocol** is one of the oldest network protocols, commonly used today for LAN and WAN networks. Every **host** gets a unique 32-bit **ip-address**, this is either static or received from a **DHCP** server. Internet networks contain several **subnets**. Those subnets used to be **classful** (A,B,C,D or E), but this wasted a lot of address space. Today we work with **CIDR** notation to determine **network id** and **host id**.

In a couple of years we will all be using IPv6! *At least, that is what people say since 1995...*

18.1.2. Internet and routers

The internet is a collection of **routers** that act as gateways between different **segments**. Routers use their **routing table** to determine the route of tcp/ip **packets**. Routers are **layer 3** devices, layer 2 contains **bridges** and **switches**, layer 1 is cabling with **repeaters** and **hubs**. Layer 2 devices know your 48-bit unique in the world **MAC** address.

18.1.3. many protocols

For reliable connections, you use **tcp**, whereas **udp** is connectionless but faster. The **icmp** error messages are used by **ping**, multicast groups are managed by **igmp** and the ip to mac resolution is done by the **broadcast** protocol **arp**.

These protocols are visible in the protocol field of the ip header, and are listed in the **/etc/protocols** file.

```
paul@laika:~$ grep tcp /etc/protocols
tcp     6       TCP             # transmission control protocol
paul@laika:~$
```

Every host receives a **hostname**, usually placed in a **DNS name space** forming the **FQDN** or Fully Qualified Domain Name. Common application level protocols like SMTP, HTTP, SSH, telnet and FTP have fixed **port numbers**.

To find a port number, look in **/etc/services**.

```
paul@laika:~$ grep tftp /etc/services
tftp            69/udp
paul@laika:~$
```

18.1.4. Practice TCP/IP

1. Which ports are used by http, pop3, ssh, telnet, nntp and ftp ?

2. Explain why e-mail and websites are sent over tcp, whereas internet streaming radio and live broadcasts are using udp.

18.2. Using TCP/IP

18.2.1. to GUI or not to GUI

If you can, setup your tcp/ip configuration at install time, otherwise use the graphical tool from your distribution. In the case of RHEL, this is the **Network Administration Tool**, Novell and OpenSUSE users can use YaST. Avoid mixed use of the GUI tool with command line or direct editing of network configuration files. You should choose only one method to manage these files, because many GUI tools will override your manually edited settings. Also, on Red Hat Servers avoid editing the files in **/etc/sysconfig/networking** manually!

Now that we settled this, let's take a look at the files and script that configure your network.

18.2.2. /sbin/ifconfig

You can use the **ifconfig** command to see the tcp/ip configuration of a network interface. The first ethernet network card on linux is eth0.

```
[root@RHEL4b ~]# ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 00:0C:29:3B:15:80
          inet addr:192.168.1.191  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fe3b:1580/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:84 errors:0 dropped:0 overruns:0 frame:0
          TX packets:80 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:9216 (9.0 KiB)  TX bytes:8895 (8.6 KiB)
          Interrupt:185 Base address:0x1400

[root@RHEL4b ~]#
```

You can also disable a network interface with **ifconfig eth0 down**, or enable it with **ifconfig eth0 up**.

Every user has access to /sbin/ifconfig, providing the path is set. Normal users cannot use it to disable or enable interfaces, or set the ip address.

```
[root@RHEL4b ~]# ifconfig eth0 192.168.1.199
[root@RHEL4b ~]#
```

The ip address change will be valid until the next change, or until reboot. You can also supply the **subnet mask** with ifconfig.

```
root@laika:~# ifconfig eth0 192.168.1.40 netmask 255.255.255.0
root@laika:~#
```

Careful, if you try this via an ssh connection, then you might lose your ssh connection.

18.2.3. /etc/init.d/network(ing)

If you have a problem with network interfaces, you can try to restart the network init script, as shown here on Ubuntu 7.04. The script stops and starts the interfaces, and renews an ip configuration with the DHCP server.

```
root@laika:~# /etc/init.d/networking restart
 * Reconfiguring network interfaces...
There is already a pid file /var/run/dhclient.eth0.pid with pid 14570
killed old client process, removed PID file
Internet Systems Consortium DHCP Client V3.0.4
Copyright 2004-2006 Internet Systems Consortium.
All rights reserved.
For info, please visit http://www.isc.org/sw/dhcp/

Listening on LPF/eth0/00:90:f5:4e:ae:17
Sending on   LPF/eth0/00:90:f5:4e:ae:17
Sending on   Socket/fallback
DHCPRELEASE on eth0 to 192.168.1.1 port 67
There is already a pid file /var/run/dhclient.eth0.pid with pid 134993416
Internet Systems Consortium DHCP Client V3.0.4
Copyright 2004-2006 Internet Systems Consortium.
All rights reserved.
For info, please visit http://www.isc.org/sw/dhcp/

Listening on LPF/eth0/00:90:f5:4e:ae:17
Sending on   LPF/eth0/00:90:f5:4e:ae:17
Sending on   Socket/fallback
DHCPDISCOVER on eth0 to 255.255.255.255 port 67 interval 5
DHCPOFFER from 192.168.1.1
DHCPREQUEST on eth0 to 255.255.255.255 port 67
DHCPACK from 192.168.1.1
bound to 192.168.1.40 -- renewal in 249143 seconds.
root@laika:~#
```

18.2.4. /etc/sysconfig

Red Hat derived Linux systems store their network configuration files in the **/etc/sysconfig/** directory. Debian derived systems do not have this directory.

18.2.4.1. /etc/sysconfig/network

Routing and host information for all network interfaces is specified in the **/etc/sysconfig/network** file. Below an example, setting 192.168.1.1 as the router (default gateway), and leaving the default hostname of localhost.localdomain. Common options not shown in this screenshot are **GATEWAYDEV** to set one of your network cards as the gateway device, and **NISDOMAIN** to specify the NIS domain name.

```
paul@RHELv4u2:~$ cat /etc/sysconfig/network
NETWORKING=yes
HOSTNAME=localhost.localdomain
GATEWAY=192.168.1.1
```

The same file, but here the hostname of the machine is not set to the default as above.

```
[paul@RHEL4b ~]$ cat /etc/sysconfig/network
NETWORKING=yes
HOSTNAME=RHEL4b
[paul@RHEL4b ~]$
```

18.2.4.2. /etc/sysconfig/network-scripts

For every network card in your computer, you should have an interface configuration file named **/etc/sysconfig/network-scripts/ifcfg-$IFNAME**. Be careful when editing these files, your edits will work, until you start the **system-config-network** (might soon be renamed to redhat-config-network) tool. This tool can and will overwrite your manual edits.

The first ethernet NIC will get **ifcfg-eth0**, the next one ifcfg-eth1 and so on. Below is an example.

```
paul@RHELv4u2:~$ cat /etc/sysconfig/network-scripts/ifcfg-eth0
DEVICE=eth0
BOOTPROTO=static
BROADCAST=192.168.1.255
HWADDR=00:0C:29:5A:86:D7
IPADDR=192.168.1.222
NETMASK=255.255.255.0
NETWORK=192.168.1.0
ONBOOT=yes
TYPE=Ethernet
```

When the second nic is configured for dhcp, then this is the ifcfg-eth1.

```
paul@RHELv4u2:~$ cat /etc/sysconfig/network-scripts/ifcfg-eth1
DEVICE=eth1
BOOTPROTO=dhcp
HWADDR=00:0C:29:6A:34:D8
ONBOOT=yes
TYPE=Ethernet
```

Besides **dhcp** and **bootp** the BOOTPROTO variable can be **static** or **none**, both meaning there should be no protocol used at boottime to set the interface values. The BROADCAST variable is no longer needed, it will be calculated.

The HWADDR can be used to make sure that the nic's get the correct name when multiple nic's are present in the computer. It can not be used to set the MAC address of a nic. For this, you need to specify the MACADDR variable. Do not use HWADDR and MACADDR in the same ifcfg file.

18.2.5. /sbin/ifup and /sbin/ifdown

The **ifup** and **ifdown** commands take an interface as argument and bring it up or down. The screenshot below deactivates the eth0 network interface.

```
root@laika:~# ifdown eth0
There is already a pid file /var/run/dhclient.eth0.pid with pid 14925
killed old client process, removed PID file
Internet Systems Consortium DHCP Client V3.0.4
Copyright 2004-2006 Internet Systems Consortium.
All rights reserved.
For info, please visit http://www.isc.org/sw/dhcp/

Listening on LPF/eth0/00:90:f5:4e:ae:17
Sending on   LPF/eth0/00:90:f5:4e:ae:17
Sending on   Socket/fallback
DHCPRELEASE on eth0 to 192.168.1.1 port 67
```

On debian derived systems, these commands will look at **/etc/network/interfaces**, whereas on Red Hat derived systems they will look at /etc/sysconfig/network-scripts/ifcfg- files. In the screenshot below ifup is used to bring up the eth0 interface. Because the /etc/network/interfaces file says eth0 uses DHCP, the ifup tool will (try to) start the dhclient daemon.

```
root@laika:~# ifup eth0
There is already a pid file /var/run/dhclient.eth0.pid with pid 134993416
Internet Systems Consortium DHCP Client V3.0.4
Copyright 2004-2006 Internet Systems Consortium.
All rights reserved.
For info, please visit http://www.isc.org/sw/dhcp/

Listening on LPF/eth0/00:90:f5:4e:ae:17
Sending on   LPF/eth0/00:90:f5:4e:ae:17
Sending on   Socket/fallback
DHCPDISCOVER on eth0 to 255.255.255.255 port 67 interval 8
DHCPOFFER from 192.168.1.1
DHCPREQUEST on eth0 to 255.255.255.255 port 67
DHCPACK from 192.168.1.1
bound to 192.168.1.40 -- renewal in 231552 seconds.
root@laika:~#
```

18.2.6. /sbin/dhclient

Home and client Linux desktops often have **dhclient** running. This is a daemon that enables a network interface to lease an ip configuration from a DHCP server. When your adapter is configured for DHCP or BOOTP, then /sbin/ifup will start the dhclient daemon.

18.2.7. /sbin/route

You can see the computer's local routing table with the **route** command (and also with **netstat -r**).

```
root@RHEL4b ~]# netstat -r
Kernel IP routing table
Destination     Gateway   Genmask         Flags   MSS Window  irtt Iface
192.168.1.0     *         255.255.255.0   U         0 0          0 eth0
[root@RHEL4b ~]# route
Kernel IP routing table
Destination     Gateway   Genmask         Flags Metric Ref    Use Iface
192.168.1.0     *         255.255.255.0   U     0      0        0 eth0
[root@RHEL4b ~]#
```

It appears this computer does not have a **gateway** configured, so we use **route add default gw** to add a **default gateway**.

```
[root@RHEL4b ~]# route add default gw 192.168.1.1
[root@RHEL4b ~]# route
Kernel IP routing table
Destination     Gateway      Genmask        Flags Metric Ref  Use Iface
192.168.1.0     *            255.255.255.0  U     0      0      0 eth0
default         192.168.1.1  0.0.0.0        UG    0      0      0 eth0
[root@RHEL4b ~]#
```

18.2.8. arp

Mac to IP resolution is handled by the **arp** protocol. The arp table can be displayed with the arp tool.

```
root@barry:~# arp -a
? (192.168.1.191) at 00:0C:29:3B:15:80 [ether] on eth1
agapi (192.168.1.73) at 00:03:BA:09:7F:D2 [ether] on eth1
anya (192.168.1.1) at 00:12:01:E2:87:FB [ether] on eth1
faith (192.168.1.41) at 00:0E:7F:41:0D:EB [ether] on eth1
kiss (192.168.1.49) at 00:D0:E0:91:79:95 [ether] on eth1
laika (192.168.1.40) at 00:90:F5:4E:AE:17 [ether] on eth1
pasha (192.168.1.71) at 00:03:BA:02:C3:82 [ether] on eth1
shaka (192.168.1.72) at 00:03:BA:09:7C:F9 [ether] on eth1
root@barry:~#
```

Anya is a Cisco Firewall, Faith is an HP Color printer, Kiss is a Kiss DP600, laika is a Clevo laptop and Agapi, Shaka and Pasha are SPARC servers. The question mark is a Red Hat Enterprise Linux server running in vmware.

18.2.9. ping

If you can ping to another host, then ip is configured.

```
[root@RHEL4b ~]# ping 192.168.1.5
PING 192.168.1.5 (192.168.1.5) 56(84) bytes of data.
64 bytes from 192.168.1.5: icmp_seq=0 ttl=64 time=1004 ms
64 bytes from 192.168.1.5: icmp_seq=1 ttl=64 time=1.19 ms
64 bytes from 192.168.1.5: icmp_seq=2 ttl=64 time=0.494 ms
64 bytes from 192.168.1.5: icmp_seq=3 ttl=64 time=0.419 ms

--- 192.168.1.5 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3009ms
rtt min/avg/max/mdev = 0.419/251.574/1004.186/434.520 ms, pipe 2
[root@RHEL4b ~]#
```

18.2.10. Red Hat network settings backup

It is always a good idea to have a backup of current network settings. The **system-config-network-cmd** can do this for you.

```
root ~# system-config-network-cmd -e > NetworkSettings20070208.txt
```

And system-config-network-cmd can also be used to restore these settings.

```
root ~# system-config-network-cmd -i -c < NetworkSettings20070208.txt
```

For other Linux Systems, take a backup of the relevant portions in /etc.

18.2.11. Restarting the network

To stop, start or restart all network interfaces and services, use **service network stop|start|restart**. *Do not stop the network when connected through ssh.*

18.2.12. ethtool

To display or change network card settings, use **ethtool**. The results depend on the capabilities of your network card. The example shows a network that auto-negotiates it's bandwidth.

```
root@laika:~# ethtool eth0
Settings for eth0:
 Supported ports: [ TP ]
 Supported link modes:   10baseT/Half 10baseT/Full
                         100baseT/Half 100baseT/Full
                         1000baseT/Full
 Supports auto-negotiation: Yes
 Advertised link modes:  10baseT/Half 10baseT/Full
                         100baseT/Half 100baseT/Full
                         1000baseT/Full
 Advertised auto-negotiation: Yes
 Speed: 1000Mb/s
 Duplex: Full
 Port: Twisted Pair
 PHYAD: 0
 Transceiver: internal
 Auto-negotiation: on
 Supports Wake-on: pumbg
 Wake-on: g
 Current message level: 0x00000033 (51)
 Link detected: yes
```

This example shows how to use ethtool to switch the bandwidth from 1000Mbit to 100Mbit and back. Note that some time passes before the nic is back to 1000Mbit.

```
root@laika:~# ethtool eth0 | grep Speed
 Speed: 1000Mb/s
root@laika:~# ethtool -s eth0 speed 100
root@laika:~# ethtool eth0 | grep Speed
 Speed: 100Mb/s
root@laika:~# ethtool -s eth0 speed 1000
root@laika:~# ethtool eth0 | grep Speed
 Speed: 1000Mb/s
```

18.2.13. Practice IP Configuration

1. Use ifconfig to list all your network interfaces and their ip-addresses. Write down your ip-address and subnet mask.

2. Use the GUI tool of your distro to set a fix ip address (use the same address as the one you got from dhcp). Verify with ifconfig and ping to a neighbour that it works. Also look at the configuration files in /etc/network or /etc/sysconfig to see how the GUI tool sets a fixed address.

3. Use the GUI tool to enable dhcp again (and verify the changes in the config files).

4. Use ifdown or ifconfig to disable your eth0 network card.

5. Restart networking to enable your network card again.

6. Is the dhclient daemon running ?

7. Verify that you have a default gateway.

8. Ping the default gateway, then look at the MAC address of the default gateway.

18.3. multiple IP adresses

18.3.1. Binding multiple ip-addresses

To bind more than one ip-addres to the same interface, use **ifcfg-eth0:0**, where the last zero can be anything else. Only two directives are required in the file.

```
root@RHELv4u2:/etc/sysconfig/network-scripts# cat ifcfg-eth0:0
DEVICE=eth0:0
```

```
IPADDR=192.168.1.232
```

18.3.2. Enabling extra ip-addresses

To activate a virtual network interface, use **ifup**, to deactivate it, use **ifdown**.

```
root@RHELv4u2:~# ifdown eth0:0
root@RHELv4u2:~# ifup eth0:0
```

18.3.3. Practice multiple IP addresses

1. Add an extra ip address to your server. Test that it works (have your neighbour ssh to it)!

2. Use ifdown and ifup to disable and enable the second ip address.

18.4. multihomed hosts

18.4.1. bonding

You can combine (bond) two physical network interfaces as one logical interface. Having two network cards serve the same IP-address doubles the bandwidth, and provides hardware redundancy. For **bonding** to work, you have to load the kernel module for bonding. You can do this manually with **modprobe**.

```
root@RHELv4u2:~# modprobe bonding
root@RHELv4u2:~# lsmod | grep bon
bonding                58984  0
```

Or automatically, by adding the alias to **/etc/modprobe.conf** (used to be called /etc/modules.conf).

```
root@RHELv4u2:~# echo alias bond0 bonding >> /etc/modprobe.conf
```

You need two network cards to enable bonding, and add the **MASTER** and **SLAVE** variables. In this case we used eth0 and eth1, configured like this.

```
root@RHELv4u2:~# cat /etc/sysconfig/network-scripts/ifcfg-eth0
DEVICE=eth0
BROADCAST=192.168.1.255
HWADDR=00:0C:29:5A:86:D7
IPADDR=192.168.1.222
NETMASK=255.255.255.0
NETWORK=192.168.1.0
ONBOOT=yes
```

```
TYPE=Ethernet
GATEWAY=192.168.1.1
MASTER=bond0
SLAVE=yes
USERCTL=no
root@RHELv4u2:~# cat /etc/sysconfig/network-scripts/ifcfg-eth1
DEVICE=eth1
BROADCAST=192.168.1.255
HWADDR=00:0C:29:5A:86:E1
IPADDR=192.168.1.232
NETMASK=255.255.255.0
NETWORK=192.168.1.0
ONBOOT=yes
TYPE=Ethernet
GATEWAY=192.168.1.1
MASTER=bond0
SLAVE=yes
USERCTL=no
root@RHELv4u2:~#
```

And you need to set up a bonding interface. In this case, we call it bond0.

```
root@RHELv4u2:~# cat /etc/sysconfig/network-scripts/ifcfg-bond0
DEVICE=bond0
BOOTPROTO=none
ONBOOT=no
NETWORK=192.168.1.0
NETMASK=255.255.255.0
IPADDR=192.168.1.229
USERCTL=no
root@RHELv4u2:~#
```

To bring up the interface, just use the **ifup bond0** command.

```
root@RHELv4u2:/etc/sysconfig/network-scripts# ifup bond0
Enslaving eth0 to bond0
Enslaving eth1 to bond0
root@RHELv4u2:~#
```

The **ifconfig** command will show you all activated interfaces.

```
root@RHELv4u2:~# ifconfig
bond0     Link encap:Ethernet  HWaddr 00:0C:29:5A:86:D7
inet addr:192.168.1.229  Bcast:192.168.1.255  Mask:255.255.255.0
inet6 addr: fe80::200:ff:fe00:0/64 Scope:Link
UP BROADCAST RUNNING MASTER MULTICAST  MTU:1500  Metric:1
RX packets:3835 errors:0 dropped:0 overruns:0 frame:0
TX packets:1001 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:0
RX bytes:469645 (458.6 KiB)  TX bytes:139816 (136.5 KiB)

eth0      Link encap:Ethernet  HWaddr 00:0C:29:5A:86:D7
inet6 addr: fe80::20c:29ff:fe5a:86d7/64 Scope:Link
UP BROADCAST RUNNING SLAVE MULTICAST  MTU:1500  Metric:1
RX packets:3452 errors:0 dropped:0 overruns:0 frame:0
TX packets:837 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1000
RX bytes:412155 (402.4 KiB)  TX bytes:117844 (115.0 KiB)
```

```
Interrupt:11 Base address:0x1400

eth1      Link encap:Ethernet  HWaddr 00:0C:29:5A:86:D7
inet6 addr: fe80::20c:29ff:fe5a:86d7/64 Scope:Link
UP BROADCAST RUNNING SLAVE MULTICAST  MTU:1500  Metric:1
RX packets:392 errors:0 dropped:0 overruns:0 frame:0
TX packets:177 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1000
RX bytes:58084 (56.7 KiB)  TX bytes:24078 (23.5 KiB)
Interrupt:10 Base address:0x1480
```

18.4.2. /proc/net/bond*

You can verify the proper working of the bonding interfaces by looking at **/proc/net/bonding/**. Below is a screenshot of a Red Hat Enterprise 5 server, with eth1 and eth2 in bonding.

```
[root@RHEL5 ~]# cat /proc/net/bonding/bond0
Ethernet Channel Bonding Driver: v3.1.2 (January 20, 2007)

Bonding Mode: load balancing (round-robin)
MII Status: up
MII Polling Interval (ms): 0
Up Delay (ms): 0
Down Delay (ms): 0

Slave Interface: eth1
MII Status: up
Link Failure Count: 0
Permanent HW addr: 00:0c:29:a0:9d:e3

Slave Interface: eth2
MII Status: up
Link Failure Count: 0
Permanent HW addr: 00:0c:29:a0:9d:ed
[root@RHEL5 ~]#
```

18.4.3. Practice multihomed hosts

1. Add a network card to the vmware machine, and bond the two cards as one virtual (double bandwidth and failover) card.

18.5. Introduction to iptables

18.5.1. Introducing iptables

The Linux kernel has a built-in stateful firewall named iptables. To stop the **iptables** firewall on Red Hat, use the service command.

```
root@RHELv4u4:~# service iptables stop
```

```
Flushing firewall rules:                                  [  OK  ]
Setting chains to policy ACCEPT: filter                   [  OK  ]
Unloading iptables modules:                               [  OK  ]
root@RHELv4u4:~#
```

The easy way to configure iptables, is to use a graphical tool like KDE's **kmyfirewall** or **Security Level Configuration Tool**. You can find the latter in the GUI menu, somewhere in System Tools - Security, or you can start it by typing **system-config-securitylevel** in bash. These tools allow for some basic firewall configuration. You can decide whether to enable or disable the firewall, and what typical standard ports are allowed when the firewall is active. You can even add some custom ports. When you are done, the configuration is written to **/etc/sysconfig/iptables** on Red Hat.

```
root@RHELv4u4:~# cat /etc/sysconfig/iptables
# Firewall configuration written by system-config-securitylevel
# Manual customization of this file is not recommended.
*filter
:INPUT ACCEPT [0:0]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
:RH-Firewall-1-INPUT - [0:0]
-A INPUT -j RH-Firewall-1-INPUT
-A FORWARD -j RH-Firewall-1-INPUT
-A RH-Firewall-1-INPUT -i lo -j ACCEPT
-A RH-Firewall-1-INPUT -p icmp --icmp-type any -j ACCEPT
-A RH-Firewall-1-INPUT -p 50 -j ACCEPT
-A RH-Firewall-1-INPUT -p 51 -j ACCEPT
-A RH-Firewall-1-INPUT -p udp --dport 5353 -d 224.0.0.251 -j ACCEPT
-A RH-Firewall-1-INPUT -p udp -m udp --dport 631 -j ACCEPT
-A RH-Firewall-1-INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
-A RH-F...NPUT -m state --state NEW -m tcp -p tcp --dport 22 -j ACCEPT
-A RH-F...NPUT -m state --state NEW -m tcp -p tcp --dport 80 -j ACCEPT
-A RH-F...NPUT -m state --state NEW -m tcp -p tcp --dport 21 -j ACCEPT
-A RH-F...NPUT -m state --state NEW -m tcp -p tcp --dport 25 -j ACCEPT
-A RH-Firewall-1-INPUT -j REJECT --reject-with icmp-host-prohibited
COMMIT
root@RHELv4u4:~#
```

To start the service, issue the **service iptables start** command. You can configure iptables to start at boot time with chkconfig.

```
root@RHELv4u4:~# service iptables start
Applying iptables firewall rules:                         [  OK  ]
root@RHELv4u4:~# chkconfig iptables on
root@RHELv4u4:~#
```

One of the nice features of iptables is that it displays extensive **status** information when queried with the **service iptables status** command.

```
root@RHELv4u4:~# service iptables status
Table: filter
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
RH-Firewall-1-INPUT  all  --  0.0.0.0/0            0.0.0.0/0
```

```
Chain FORWARD (policy ACCEPT)
target     prot opt source               destination
RH-Firewall-1-INPUT  all  --  0.0.0.0/0            0.0.0.0/0

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination

Chain RH-Firewall-1-INPUT (2 references)
target  prot opt source      destination
ACCEPT  all  --  0.0.0.0/0   0.0.0.0/0
ACCEPT  icmp --  0.0.0.0/0   0.0.0.0/0   icmp type 255
ACCEPT  esp  --  0.0.0.0/0   0.0.0.0/0
ACCEPT  ah   --  0.0.0.0/0   0.0.0.0/0
ACCEPT  udp  --  0.0.0.0/0   224.0.0.251 udp dpt:5353
ACCEPT  udp  --  0.0.0.0/0   0.0.0.0/0   udp dpt:631
ACCEPT  all  --  0.0.0.0/0   0.0.0.0/0   state RELATED,ESTABLISHED
ACCEPT  tcp  --  0.0.0.0/0   0.0.0.0/0   state NEW tcp dpt:22
ACCEPT  tcp  --  0.0.0.0/0   0.0.0.0/0   state NEW tcp dpt:80
ACCEPT  tcp  --  0.0.0.0/0   0.0.0.0/0   state NEW tcp dpt:21
ACCEPT  tcp  --  0.0.0.0/0   0.0.0.0/0   state NEW tcp dpt:25
REJECT  all  --  0.0.0.0/0   0.0.0.0/0   reject-with icmp-host-prohibited

root@RHELv4u4:~#
```

Mastering firewall configuration requires a decent knowledge of tcp/ip. Good iptables tutorials can be found online here http://iptables-tutorial.frozentux.net/iptables-tutorial.html and here http://tldp.org/HOWTO/IP-Masquerade-HOWTO/.

18.5.2. Practice iptables

1. Verify whether the firewall is running.

2. Disable the firewall.

18.6. xinetd and inetd

18.6.1. About the superdaemon

Back when resources like RAM memory were limited, a super-server was deviced to listen to all sockets and start the appropriate daemon only when needed. Services like swat, telnet and vmware are typically served by such a super-server. The **xinetd** superserver is more recent than **inetd**. We will discuss the configuration both daemons.

Recent Linux distributions like RHEL5 and Ubuntu8.04 do not install inetd or xinetd by default.

18.6.2. inetd or xinetd

First verify whether your computer is running inetd or xinetd. This Debian 4.0 Etch is running inetd.

```
root@barry:~# ps fax | grep inet
 3870 ?        Ss     0:00 /usr/sbin/inetd
```

This Red Hat Enterprise Linux 4 update 4 is running xinetd.

```
[root@RHEL4b ~]# ps fax | grep inet
 3003 ?        Ss     0:00 xinetd -stayalive -pidfile /var/run/xinetd.pid
```

Both daemons have the same functionality (listening to many ports, starting other daemons when they are needed), but they have different configuration files.

18.6.3. The superdaemon xinetd

The **xinetd** daemon is often called a superdaemon because it listens to a lot of incoming connections, and starts other daemons when they are needed. When a connection request is received, xinetd will first check TCP wrappers (/etc/hosts.allow and /etc/hosts.deny) and then give control of the connection to the other daemon. This superdaemon is configured through **/etc/xinetd.conf** and the files in the directory **/etc/xinetd.d**. Let's first take a look at /etc/xinetd.conf.

```
paul@RHELv4u2:~$ cat /etc/xinetd.conf
#
# Simple configuration file for xinetd
#
# Some defaults, and include /etc/xinetd.d/

defaults
{
instances               = 60
log_type                = SYSLOG authpriv
log_on_success          = HOST PID
log_on_failure          = HOST
cps                     = 25 30
}

includedir /etc/xinetd.d

paul@RHELv4u2:~$
```

According to the settings in this file, xinetd can handle 60 client requests at once. It uses the **authpriv** facility to log the host ip-address and pid of successful daemon spawns. When a service (aka protocol linked to daemon) gets more than 25 cps (connections per second), it holds subsequent requests for 30 seconds.

The directory **/etc/xinetd.d** contains more specific configuration files. Let's also take a look at one of them.

```
paul@RHELv4u2:~$ ls /etc/xinetd.d
amanda     chargen-udp  echo      klogin      rexec   talk
```

```
amandaidx  cups-lpd     echo-udp  krb5-telnet  rlogin  telnet
amidxtape  daytime      eklogin   kshell       rsh     tftp
auth       daytime-udp  finger    ktalk        rsync   time
chargen    dbskkd-cdb   gssftp    ntalk        swat    time-udp
paul@RHELv4u2:~$ cat /etc/xinetd.d/swat
# default: off
# description: SWAT is the Samba Web Admin Tool. Use swat \
#              to configure your Samba server. To use SWAT, \
#              connect to port 901 with your favorite web browser.
service swat
{
port            = 901
socket_type     = stream
wait            = no
only_from       = 127.0.0.1
user            = root
server          = /usr/sbin/swat
log_on_failure  += USERID
disable         = yes
}
paul@RHELv4u2:~$
```

The services should be listed in the **/etc/services** file. Port determines the service port, and must be the same as the port specified in /etc/services. The **socket_type** should be set to **stream** for tcp services (and to dgram for udp). The **log_on_failure +=** concats the userid to the log message formatted in /etc/xinetd.conf. The last setting **disable** can be set to yes or no. Setting this to **no** means the service is enabled!

Check the xinetd and xinetd.conf manual pages for many more configuration options.

18.6.4. The superdaemon inetd

This superdaemon has only one configuration file **/etc/inetd.conf**. Every protocol or daemon that it is listening for, gets one line in this file.

```
root@barry:~# grep ftp /etc/inetd.conf
tftp dgram udp wait nobody /usr/sbin/tcpd /usr/sbin/in.tftpd /boot/tftp
root@barry:~#
```

You can disable a service in inetd.conf above by putting a # at the start of that line. Here an example of the disabled vmware web interface (listening on tcp port 902).

```
paul@laika:~$ grep vmware /etc/inetd.conf
#902 stream tcp nowait root /usr/sbin/vmware-authd vmware-authd
```

18.6.5. Practice

1. Verify on all systems whether they are using xinetd or inetd.

2. Look at the configuration files.

3. (If telnet is installable, then replace swat in these questions with telnet) Is swat installed ? If not, then install swat and look at the changes in the (x)inetd configuration. Is swat enabled or disabled ?

4. Disable swat, test it. Enable swat, test it.

18.7. OpenSSH

18.7.1. Secure Shell

Avoid using **telnet**, **rlogin** and **rsh** to remotely connect to your servers. These older protocols do not encrypt the login session, which means your user id and password can be sniffed by tools like **ethereal** aka wireshark. To securely connect to your servers, use **OpenSSH**. An ssh connection always starts with a cryptographic handshake, followed by encryption of the transport layer using a symmetric cypher. Then authentication takes place (using user id/password or public/private keys) and communication can take place over the encrypted connection. In other words, the tunnel is encrypted before you start typing anything.

The OpenSSH package is maintained by the **OpenBSD** people and is distributed with a lot of operating systems (it may even be the most popular package in the world). Below sample use of **ssh** to connect from one server (RHELv4u2) to another one (RHELv4u4).

```
paul@RHELv4u2:~$ ssh 192.168.1.220
The authenticity of host '192.168.1.220' can't be established.
RSA key fingerprint is c4:3c:52:e6:d8:8b:ce:17:8b:c9:78:5a:f3:51:06:4f.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.1.220' (RSA) to the list of known...
paul@192.168.1.220's password:
Last login: Sun Jan 21 07:16:26 2007 from 192.168.1.40
paul@RHELv4u4:~$
```

The second time ssh remembers the connection. It added an entry to the ~/.ssh/known_hosts file.

```
paul@RHELv4u2:~$ ssh 192.168.1.220
paul@192.168.1.220's password:
Last login: Sun Jan 21 08:49:19 2007 from 192.168.1.222
paul@RHELv4u4:~$
```

18.7.2. SSH Protocol versions

The ssh protocol has two versions (1 and 2). Avoid using version 1 anywhere, since it contains some known vulnerabilities. You can control the protocol version via **/etc/ssh/ssh_config** for the client side and **/etc/ssh/sshd_config** for the openssh-server daemon.

```
root@laika:/etc/ssh# grep Protocol ssh_config
#   Protocol 2,1
root@laika:/etc/ssh# grep Protocol sshd_config
Protocol 2
root@laika:/etc/ssh#
```

Configuration of ssh is done in the **/etc/ssh** directory and is pretty straightforward.

18.7.3. About Public and Private keys

Imagine Alice and Bob, two people that like to communicate with eachother. Using public and private keys they can communicate with encryption and with authentication.

When Alice wants to send an encrypted message to Bob, she uses the public key of Bob. Bob shares his Public Key with Alice, but keeps his Private Key private! Since Bob is the only one to have Bob's Private Key, Alice is sure that Bob is the only one that can read the encrypted message.

When Bob wants to verify that the message came from Alice, Bob uses the Public Key of Alice to verify that Alice signed the message with her Private Key. Since Alice is the only one to have Alice's Private Key, Bob is sure the message came from Alice.

18.7.4. Setting up passwordless ssh

To set up passwordless ssh authentication through public/private keys, use **ssh-keygen** to generate a key pair without a passphrase, and then copy your public key to the destination server. Let's do this step by step.

In the example that follows, we will set up ssh without password between Alice and Bob. Alice has an account on a Red Hat Enterprise Linux server, Bob is using Ubuntu on his laptop. Bob wants to give Alice access using ssh and the public and private key system. This means that even if Bob changes his password on his laptop, Alice will still have access.

18.7.4.1. ssh-keygen

The example below shows how Alice uses **ssh-keygen** to generate a key pair. Alice does not enter a passphrase.

```
[alice@RHEL5 ~]$ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/alice/.ssh/id_rsa):
Created directory '/home/alice/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/alice/.ssh/id_rsa.
Your public key has been saved in /home/alice/.ssh/id_rsa.pub.
The key fingerprint is:
9b:ac:ac:56:c2:98:e5:d9:18:c4:2a:51:72:bb:45:eb alice@RHEL5
[alice@RHEL5 ~]$
```

18.7.4.2. ~/.ssh

While ssh-keygen generates a public and a private key, it will also create a hidden .ssh directory with proper permissions. If you create the .ssh directory manually, then you need to chmod 700 it! Otherwise ssh will refuse to use the keys (world readable private keys are not secure!).

As you can see, the .ssh directory is secure in Alice's home directory.

```
[alice@RHEL5 ~]$ ls -ld .ssh
drwx------ 2 alice alice 4096 May  1 07:38 .ssh
[alice@RHEL5 ~]$
```

Bob is using Ubuntu at home. He decides to manually create the .ssh directory, so he needs to manually secure it.

```
bob@laika:~$ mkdir .ssh
bob@laika:~$ ls -ld .ssh
drwxr-xr-x 2 bob bob 4096 2008-05-14 16:53 .ssh
bob@laika:~$ chmod 700 .ssh/
bob@laika:~$
```

18.7.4.3. id_rsa and id_rsa.pub

The ssh-keygen command generate two keys in .ssh. The public key is named **~/.ssh/id_rsa.pub**. The private key is named **~/.ssh/id_rsa**.

```
[alice@RHEL5 ~]$ ls -l .ssh/
total 16
-rw------- 1 alice alice 1671 May  1 07:38 id_rsa
-rw-r--r-- 1 alice alice  393 May  1 07:38 id_rsa.pub
[alice@RHEL5 ~]$
```

18.7.4.4. scp

To copy the public key from Alice's server tot Bob's laptop, Alice decides to use **scp**.

```
[alice@RHEL5 .ssh]$ scp id_rsa.pub bob@192.168.48.92:~/.ssh/authorized_keys
bob@192.168.48.92's password:
id_rsa.pub                                    100%  393     0.4KB/s   00:00
[alice@RHEL5 .ssh]$
```

Be careful when copying a second key! Do not overwrite the first key, instead append the key to the same ~/.ssh/authorized_keys file!

18.7.4.5. authorized_keys

In your ~/.ssh directory, you can create a file called **authorized_keys**. This file can contain one or more public keys from people you trust. Those trusted people can use their private keys to prove their identity and gain access to your account via ssh (without password). The example shows Bob's authorized_keys file containing the public key of Alice.

```
bob@laika:~$ cat .ssh/authorized_keys
ssh-rsa AAAAB3NzaC1yc2EAAAABIwAAAQEApCQ9xzyLzJes1sR+hPyqW2vyzt1D4zTLqk\
MDWBR4mMFuUZD/O583I3Lg/Q+JIq0RSksNzaL/BNLDou1jMpBe2Dmf/u22u4KmqlJBfDhe\
yTmGSBzeNYCYRSMq78CT9l9a+y6x/shucwhaILsy8A2XfJ9VCggkVtu7XlWFDL2cum08/0\
mRFwVrfc/uPsAn5XkkTscl4g21mQbnp9wJC40pGSJXXMuFOk8MgCb5ieSnpKFniAKM+tEo\
/vjDGSi3F/bxu691jscrU0VUdIoOSo98HUfEf7jKBRikxGAC7I4HLa+/zX73OIvRFAb2hv\
tUhn6RHrBtUJUjbSGiYeFTLDfcTQ== alice@RHEL5
bob@laika:~$
```

18.7.4.6. passwordless ssh

Alice can now use ssh to connect passwordless to Bob's laptop. In combination with ssh's capability to execute commands on the remote host, this can be useful in pipes across different machines.

```
[alice@RHEL5 ~]$ ssh bob@192.168.48.92 "ls -l .ssh"
total 4
-rw-r--r-- 1 bob bob 393 2008-05-14 17:03 authorized_keys
[alice@RHEL5 ~]$
```

18.7.5. X forwarding via SSH

The **ssh protocol** will remember the servers it connected to (and warn you in case something suspicious happened), and will use strong 128-bit encryption. Another popular feature of ssh is called **X11 forwarding** and is implemented with **ssh -X**.

Below an example of X11 forwarding: user paul logs in as user greet on her computer to start the graphical application mozilla-thunderbird. Although the application will run on the remote computer from greet, it will be displayed on the screen attached locally to paul's computer.

```
paul@laika:~/PDF$ ssh -X greet@greet.dyndns.org -p 55555
Warning: Permanently added the RSA host key for IP address \
'81.240.174.161' to the list of known hosts.
Password:
Linux raika 2.6.8-2-686 #1 Tue Aug 16 13:22:48 UTC 2005 i686 GNU/Linux
```

```
Last login: Thu Jan 18 12:35:56 2007
greet@raika:~$ ps fax | grep thun
greet@raika:~$ mozilla-thunderbird &
[1] 30336
```

18.7.6. Troubleshooting ssh

Use **ssh -v** to get debug information about the ssh connection attempt.

```
paul@laika:~$ ssh -v bert@192.168.1.192
OpenSSH_4.3p2 Debian-8ubuntu1, OpenSSL 0.9.8c 05 Sep 2006
debug1: Reading configuration data /home/paul/.ssh/config
debug1: Reading configuration data /etc/ssh/ssh_config
debug1: Applying options for *
debug1: Connecting to 192.168.1.192 [192.168.1.192] port 22.
debug1: Connection established.
debug1: identity file /home/paul/.ssh/identity type -1
debug1: identity file /home/paul/.ssh/id_rsa type 1
debug1: identity file /home/paul/.ssh/id_dsa type -1
debug1: Remote protocol version 1.99, remote software version OpenSSH_3
debug1: match: OpenSSH_3.9p1 pat OpenSSH_3.*
debug1: Enabling compatibility mode for protocol 2.0
...
```

18.7.7. Practice SSH

1. Create a user for your neighbour, then test ssh to your neighbour (by ip-address or by hostname). (You might need to install the openssh-server with aptitude.)

2. Create a bookmark in Firefox, then close your firefox! Use ssh -X to run firefox on your screen, but on your neighbour's computer. Do you see your neighbour's bookmark ?

3. Verify in the ssh configuration files that only protocol version 2 is allowed.

4. Use ssh-keygen to create a keypair without passphrase. Setup passwordless ssh between you and your neighbour. (or between the ubuntu and the Red Hat)

18.8. Network File System

18.8.1. Network Attached Storage (NAS)

NAS means using separate servers with lots of storage, connected over a (hopefully very fast) network. NAS servers offer **file-based access** over the network with protocols like **NCP** (old Novell Netware), Sun's **NFS** (common on Unix) or **SMB** (implemented on Unix/Linux with Samba). NAS is not to be confused with **SAN**, which uses **block-based access** over proprietary protocols (Fiber Channel, iSCSI, ...).

A **NAS head** is a NAS without on-board storage, which connects to a SAN and acts as a translator between the file-level NAS protocols and the block-level SAN protocols.

18.8.2. NFS: the Network File System

18.8.2.1. protocol versions

The older **NFS** versions 2 and 3 are stateless (udp) by default, but they can use tcp. Clients connect to the server using **RPC** (on Linux this is controlled by the **portmap** daemon. Look at **rpcinfo** to verify that NFS and its related services are running.

```
root@RHELv4u2:~# /etc/init.d/portmap status
portmap (pid 1920) is running...
root@RHELv4u2:~# rpcinfo -p
program vers proto   port
100000    2   tcp    111  portmapper
100000    2   udp    111  portmapper
100024    1   udp  32768  status
100024    1   tcp  32769  status
root@RHELv4u2:~# service nfs start
Starting NFS services:                                     [  OK  ]
Starting NFS quotas:                                       [  OK  ]
Starting NFS daemon:                                       [  OK  ]
Starting NFS mountd:                                       [  OK  ]
```

The same rpcinfo command when NFS is started.

```
root@RHELv4u2:~# rpcinfo -p
program vers proto   port
100000    2   tcp    111  portmapper
100000    2   udp    111  portmapper
100024    1   udp  32768  status
100024    1   tcp  32769  status
100011    1   udp    985  rquotad
100011    2   udp    985  rquotad
100011    1   tcp    988  rquotad
100011    2   tcp    988  rquotad
100003    2   udp   2049  nfs
100003    3   udp   2049  nfs
100003    4   udp   2049  nfs
100003    2   tcp   2049  nfs
100003    3   tcp   2049  nfs
100003    4   tcp   2049  nfs
100021    1   udp  32770  nlockmgr
100021    3   udp  32770  nlockmgr
100021    4   udp  32770  nlockmgr
100021    1   tcp  32789  nlockmgr
100021    3   tcp  32789  nlockmgr
100021    4   tcp  32789  nlockmgr
100005    1   udp   1004  mountd
100005    1   tcp   1007  mountd
100005    2   udp   1004  mountd
100005    2   tcp   1007  mountd
100005    3   udp   1004  mountd
100005    3   tcp   1007  mountd
root@RHELv4u2:~#
```

NFS version 4 requires tcp (port 2049) and supports **Kerberos** user authentication as an option. NFS authentication only takes place when mounting the share. NFS versions 2 and 3 authenticate only the host.

18.8.2.2. server configuration

NFS is configured in **/etc/exports**. Here is a sample /etc/exports to explain the syntax. You need some way (NIS domain or LDAP) to synchronize userid's across computers when using NFS a lot. The **rootsquash** option will change UID 0 to the UID of the nfsnobody user account. The **sync** option will write writes to disk before completing the client request.

```
paul@laika:~$ cat /etc/exports
# Everyone can read this share
/mnt/data/iso  *(ro)

# Only the computers barry and pasha can readwrite this one
/var/www pasha(rw) barry(rw)

# same, but without root squashing for barry
/var/ftp pasha(rw) barry(rw,no_root_squash)

# everyone from the netsec.lan domain gets access
/var/backup       *.netsec.lan(rw)

# ro for one network, rw for the other
/var/upload   192.168.1.0/24(ro) 192.168.5.0/24(rw)
```

You don't need to restart the nfs server to start exporting your newly created exports. You can use the **exportfs -va** command to do this. It will write the exported directories to **/var/lib/nfs/etab**, where they are immediately applied.

18.8.2.3. client configuration

We have seen the **mount** command and the **/etc/fstab** file before.

```
root@RHELv4u2:~# mount -t nfs barry:/mnt/data/iso /home/project55/
root@RHELv4u2:~# cat /etc/fstab | grep nfs
barry:/mnt/data/iso   /home/iso               nfs     defaults    0 0
root@RHELv4u2:~#
```

18.8.2.4. Mounting NAS

Just a simple fictitious example. Suppose the project55 people tell you they only need a couple of CD-ROM images, and you already have them available on an NFS server. You could issue the following command to mount the network attached storage on their /home/project55 mount point.

```
root@RHELv4u2:~# mount -t nfs 192.168.1.40:/mnt/data/iso /home/project55/
root@RHELv4u2:~# ls -lh /home/project55/
total 3.6G
drwxr-xr-x  2 1000 1000 4.0K Jan 16 17:55 RHELv4u1
drwxr-xr-x  2 1000 1000 4.0K Jan 16 14:14 RHELv4u2
drwxr-xr-x  2 1000 1000 4.0K Jan 16 14:54 RHELv4u3
drwxr-xr-x  2 1000 1000 4.0K Jan 16 11:09 RHELv4u4
-rw-r--r--  1 root root 1.6G Oct 13 15:22 sled10-vmwarews5-vm.zip
root@RHELv4u2:~#
```

18.8.3. Practice NFS

1. Create two directories with some files. Use NFS to share one of them as read only, the other must be writable. Have your neighbour connect to them to test.

2. Investigate the user owner of the files created by your neighbour.

3. Protect a share by ip-address or hostname, so only your neighbour can connect.

Chapter 19. Scheduling

19.1. About scheduling

Linux administrators use the **at** to schedule one time jobs. Recurring jobs are better scheduled with **cron**. The next two sections will discuss both tools.

19.2. at

Simple scheduling can be done with the **at** command. This screenshot shows the scheduling of the date command at 22:01 and the sleep command at 22:03.

```
root@laika:~# at 22:01
at> date
at> <EOT>
job 1 at Wed Aug  1 22:01:00 2007
root@laika:~# at 22:03
at> sleep 10
at> <EOT>
job 2 at Wed Aug  1 22:03:00 2007
root@laika:~#
```

In real life you will hopefully be scheduling more useful commands ;-)

It is easy to check when jobs are scheduled with the **atq** or **at -l** commands.

```
root@laika:~# atq
1       Wed Aug  1 22:01:00 2007 a root
2       Wed Aug  1 22:03:00 2007 a root
root@laika:~# at -l
1       Wed Aug  1 22:01:00 2007 a root
2       Wed Aug  1 22:03:00 2007 a root
root@laika:~#
```

The at command understands English words like tomorrow and teatime to schedule commands the next day and at four in the afternoon.

```
root@laika:~# at 10:05 tomorrow
at> sleep 100
at> <EOT>
job 5 at Thu Aug  2 10:05:00 2007
root@laika:~# at teatime tomorrow
at> tea
at> <EOT>
job 6 at Thu Aug  2 16:00:00 2007
root@laika:~# atq
6       Thu Aug  2 16:00:00 2007 a root
5       Thu Aug  2 10:05:00 2007 a root
root@laika:~#
```

Jobs in the at queue can be removed with **atrm**.

```
root@laika:~# atq
6       Thu Aug  2 16:00:00 2007 a root
5       Thu Aug  2 10:05:00 2007 a root
root@laika:~# atrm 5
root@laika:~# atq
6       Thu Aug  2 16:00:00 2007 a root
root@laika:~#
```

You can also use the **/etc/at.allow** and **/etc/at.deny** files to manage who can schedule jobs with at.

19.3. cron

19.3.1. crontab

The **crontab(1)** command can be used to maintain the **crontab(5)** file. Each user can have their own crontab file to schedule jobs at a specific time. This time can be specified with five fields in this order: minute, hour, day of the month, month and day of the week. If a field contains an asterisk (*), then this means all values of that field.

The following example means : run script42 eight minutes after two, every day of the month, every month and every day of the week.

```
8 14 * * * script42
```

Run script8472 every month on the first of the month at 25 past midnight.

```
25 0 1 * * script8472
```

Run this script33 every two minutes on Sunday (both 0 and 7 refer to Sunday).

```
*/2 * * * 0
```

Instead of these five fields, you can also type one of these: @reboot, @yearly or @annually, @monthly, @weekly, @daily or @midnight, and @hourly.

Users should not edit the crontab file directly, instead they should type **crontab -e** which will use the editor defined in the EDITOR or VISUAL environment variable. Users can display their cron table with **crontab -l**. The **cron daemon** is reading the cron tables, taking into account the **/etc/cron.allow** and **/etc/cron.deny** files.

19.3.2. /etc/cron*

The **/etc/crontab** file contains entries for when to run hourly/daily/weekly/monthly tasks. It will look similar to this output.

```
SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

20 3 * * *        root    run-parts --report /etc/cron.daily
40 3 * * 7        root    run-parts --report /etc/cron.weekly
55 3 1 * *        root    run-parts --report /etc/cron.monthly
```

The directories shown in the next screenshot contain the tasks that are run at the times scheduled in /etc/crontab. The /etc/cron.d directory is for special cases, to schedule jobs that require finer control than hourly/daily/weekly/monthly.

```
paul@laika:~$ ls -ld /etc/cron.*
drwxr-xr-x 2 root root 4096 2008-04-11 09:14 /etc/cron.d
drwxr-xr-x 2 root root 4096 2008-04-19 15:04 /etc/cron.daily
drwxr-xr-x 2 root root 4096 2008-04-11 09:14 /etc/cron.hourly
drwxr-xr-x 2 root root 4096 2008-04-11 09:14 /etc/cron.monthly
drwxr-xr-x 2 root root 4096 2008-04-11 09:14 /etc/cron.weekly
```

19.4. Practice Scheduling

1. Schedule two jobs with at, display the at queue and remove a job.

2. As normal user, use crontab -e to schedule a script to run every four minutes.

3. As root, display the crontab file of your normal user.

4. Take a look at the cron files and directories in /etc and understand them. What is the run-parts command doing ?

Chapter 20. Logging

20.1. About logging

20.1.1. /var/log

The location for log files according to the FHS is **/var/log**. You will find a lot of log files and directories for common applications in /var/log.

```
[paul@RHEL4b ~]$ ls /var/log
acpid           cron.2    maillog.2   quagga           secure.4
amanda          cron.3    maillog.3   radius           spooler
anaconda.log    cron.4    maillog.4   rpmpkgs          spooler.1
anaconda.syslog cups      mailman     rpmpkgs.1        spooler.2
anaconda.xlog   dmesg     messages    rpmpkgs.2        spooler.3
audit           exim      messages.1  rpmpkgs.3        spooler.4
boot.log        gdm       messages.2  rpmpkgs.4        squid
boot.log.1      httpd     messages.3  sa               uucp
boot.log.2      iiim      messages.4  samba            vbox
boot.log.3      iptraf    mysqld.log  scrollkeeper.log vmware-tools-guestd
boot.log.4      lastlog   news        secure           wtmp
canna           mail      pgsql       secure.1         wtmp.1
cron            maillog   ppp         secure.2         Xorg.0.log
cron.1          maillog.1 prelink.log secure.3         Xorg.0.log.old
[paul@RHEL4b ~]$
```

20.1.2. /var/log/messages

A typical first file to check when troubleshooting is the **/var/log/messages** file. By default this file will contain information on what just happened to the system.

```
[root@RHEL4b ~]# tail /var/log/messages
Jul 30 05:13:56 anacron: anacron startup succeeded
Jul 30 05:13:56 atd: atd startup succeeded
Jul 30 05:13:57 messagebus: messagebus startup succeeded
Jul 30 05:13:57 cups-config-daemon: cups-config-daemon startup succeeded
Jul 30 05:13:58 haldaemon: haldaemon startup succeeded
Jul 30 05:14:00 fstab-sync[3560]: removed all generated mount points
Jul 30 05:14:01 fstab-sync[3628]: added mount point /media/cdrom for...
Jul 30 05:14:01 fstab-sync[3646]: added mount point /media/floppy for...
Jul 30 05:16:46 sshd(pam_unix)[3662]: session opened for user paul by...
Jul 30 06:06:37 su(pam_unix)[3904]: session opened for user root by paul
[root@RHEL4b ~]#
```

20.2. Login logging

To keep track of who is logging into the system, Linux can maintain the **/var/log/wtmp**, **/var/log/btmp**, **/var/run/utmp** and **/var/log/lastlog** files.

20.2.1. /var/run/utmp (who)

Use the **who** command to see the /var/run/utmp file. This command is showing you all the **currently** logged in users. Notice that the utmp file is in /var/run and not in /var/log .

```
[root@rhel4 ~]# who
paul     pts/1        Feb 14 18:21 (192.168.1.45)
sandra   pts/2        Feb 14 18:11 (192.168.1.42)
inge     pts/3        Feb 14 12:01 (192.168.1.33)
els      pts/4        Feb 14 14:33 (192.168.1.19)
```

20.2.2. /var/log/wtmp (last)

The /var/log/wtmp file is updated by the **login program**. Use **last** to see the /var/run/wtmp file.

```
[root@rhel4a ~]# last | head
paul     pts/1       192.168.1.45     Wed Feb 14 18:39   still logged in
reboot   system boot 2.6.9-42.0.8.ELs Wed Feb 14 18:21          (01:15)
nicolas  pts/5       pc-dss.telematic Wed Feb 14 12:32 - 13:06  (00:33)
stefaan  pts/3       pc-sde.telematic Wed Feb 14 12:28 - 12:40  (00:12)
nicolas  pts/3       pc-nae.telematic Wed Feb 14 11:36 - 12:21  (00:45)
nicolas  pts/3       pc-nae.telematic Wed Feb 14 11:34 - 11:36  (00:01)
dirk     pts/5       pc-dss.telematic Wed Feb 14 10:03 - 12:31  (02:28)
nicolas  pts/3       pc-nae.telematic Wed Feb 14 09:45 - 11:34  (01:48)
dimitri  pts/5       rhel4            Wed Feb 14 07:57 - 08:38  (00:40)
stefaan  pts/4       pc-sde.telematic Wed Feb 14 07:16 - down   (05:50)
[root@rhel4a ~]#
```

The last command can also be used to get a list of last reboots.

```
[paul@rekkie ~]$ last reboot
reboot   system boot  2.6.16-rekkie   Mon Jul 30 05:13     (370+08:42)

wtmp begins Tue May 30 23:11:45 2006
[paul@rekkie ~]$
```

20.2.3. /var/log/lastlog (lastlog)

Use **lastlog** to see the /var/log/lastlog file.

```
[root@rhel4a ~]# lastlog | tail
tim              pts/5  10.170.1.122     Tue Feb 13 09:36:54 +0100 2007
rm               pts/6  rhel4            Tue Feb 13 10:06:56 +0100 2007
henk                                     **Never logged in**
stefaan          pts/3  pc-sde.telematic Wed Feb 14 12:28:38 +0100 2007
dirk             pts/5  pc-dss.telematic Wed Feb 14 10:03:11 +0100 2007
```

```
arsene                                   **Never logged in**
nicolas          pts/5  pc-dss.telematic Wed Feb 14 12:32:18 +0100 2007
dimitri          pts/5  rhel4            Wed Feb 14 07:57:19 +0100 2007
bashuserrm       pts/7  rhel4            Tue Feb 13 10:35:40 +0100 2007
kornuserrm       pts/5  rhel4            Tue Feb 13 10:06:17 +0100 2007
[root@rhel4a ~]#
```

20.2.4. /var/log/btmp (lastb)

There is also the **lastb** command to display the **/var/log/btmp** file. This file is updated by the login program when entering the wrong password, so it contains failed login attempts. Many computers will not have this file, resulting in no logging of failed login attempts.

```
[root@RHEL4b ~]# lastb
lastb: /var/log/btmp: No such file or directory
Perhaps this file was removed by the operator to prevent logging lastb\
 info.
[root@RHEL4b ~]#
```

The reason given for this is that users sometimes type their password by mistake instead of their login, so this world readable file poses a security risk. You can enable bad login logging by simply creating the file. Doing a chmod o-r /var/log/btmp improves security.

```
[root@RHEL4b ~]# touch /var/log/btmp
[root@RHEL4b ~]# ll /var/log/btmp
-rw-r--r--  1 root root 0 Jul 30 06:12 /var/log/btmp
[root@RHEL4b ~]# chmod o-r /var/log/btmp
[root@RHEL4b ~]# lastb

btmp begins Mon Jul 30 06:12:19 2007
[root@RHEL4b ~]#
```

Failed logins via ssh, rlogin or su are not registered in /var/log/btmp. Failed logins via tty are.

```
[root@RHEL4b ~]# lastb
HalvarFl tty3                  Mon Jul 30 07:10 - 07:10  (00:00)
Maria    tty1                  Mon Jul 30 07:09 - 07:09  (00:00)
Roberto  tty1                  Mon Jul 30 07:09 - 07:09  (00:00)

btmp begins Mon Jul 30 07:09:32 2007
[root@RHEL4b ~]#
```

20.2.5. su and ssh logins

Depending on the distribution, you may also have the **/var/log/secure** file being filled with messages from the auth and/or authpriv syslog facilities. This log will include su and/or ssh failed login attempts. Some distributions put this in **/var/log/auth.log**, verify the syslog configuration.

```
[root@RHEL4b ~]# cat /var/log/secure
Jul 30 07:09:03 sshd[4387]: Accepted publickey for paul from ::ffff:19\
2.168.1.52 port 33188 ssh2
Jul 30 05:09:03 sshd[4388]: Accepted publickey for paul from ::ffff:19\
2.168.1.52 port 33188 ssh2
Jul 30 07:22:27 sshd[4655]: Failed password for Hermione from ::ffff:1\
92.168.1.52 port 38752 ssh2
Jul 30 05:22:27 sshd[4656]: Failed password for Hermione from ::ffff:1\
92.168.1.52 port 38752 ssh2
Jul 30 07:22:30 sshd[4655]: Failed password for Hermione from ::ffff:1\
92.168.1.52 port 38752 ssh2
Jul 30 05:22:30 sshd[4656]: Failed password for Hermione from ::ffff:1\
92.168.1.52 port 38752 ssh2
Jul 30 07:22:33 sshd[4655]: Failed password for Hermione from ::ffff:1\
92.168.1.52 port 38752 ssh2
Jul 30 05:22:33 sshd[4656]: Failed password for Hermione from ::ffff:1\
92.168.1.52 port 38752 ssh2
Jul 30 08:27:33 sshd[5018]: Invalid user roberto from ::ffff:192.168.1\
.52
Jul 30 06:27:33 sshd[5019]: input_userauth_request: invalid user rober\
to
Jul 30 06:27:33 sshd[5019]: Failed none for invalid user roberto from \
::ffff:192.168.1.52 port 41064 ssh2
Jul 30 06:27:33 sshd[5019]: Failed publickey for invalid user roberto \
from ::ffff:192.168.1.52 port 41064 ssh2
Jul 30 08:27:36 sshd[5018]: Failed password for invalid user roberto f\
rom ::ffff:192.168.1.52 port 41064 ssh2
Jul 30 06:27:36 sshd[5019]: Failed password for invalid user roberto f\
rom ::ffff:192.168.1.52 port 41064 ssh2
[root@RHEL4b ~]#
```

You can enable this yourself, with a custom logfile by adding the following line tot syslog.conf.

```
auth.*,authpriv.*                /var/log/customsec.log
```

20.3. Syslogd daemon

20.3.1. About syslog

The standard method of logging on Linux is through the **syslogd** daemon. Syslog was developed by **Eric Allman** for sendmail, but quickly became a standard among many Unix applications and was much later written as rfc 3164. The syslog daemon can receive messages on udp **port 514** from many applications (and appliances), and can append to logfiles, print, display messages on terminals and forward logs to other syslogd daemons on other machines. The syslogd daemon is configured in **/etc/syslog.conf**.

Each line in the configuration file uses a **facility** to determine where the message is coming from. It also contains a **level** for the severity of the message, and an **action** to decide on what to do with the message.

20.3.2. Facilities

The **man syslog.conf** will explain the different default facilities for certain daemons, such as mail, lpr, news and kern(el) messages. The local0 to local7 facility can be used for appliances (or any networked device that supports syslog). Here is a list of all facilities for syslog.conf version 1.3. The security keyword is deprecated.

```
auth (security)
authpriv
cron
daemon
ftp
kern
lpr mail
mark (internal use only)
news
syslog
user
uucp
local0-7
```

20.3.3. Levels

The worst severity a message can have is **emerg** followed by **alert** and **crit**. Lowest priority should go to **info** and **debug** messages. Specifying a severity will also log all messages with a higher severity. You can prefix the severity with = to obtain only messages that match that severity. You can also specify **.none** to prevent a specific action from any message from a certain facility.

Here is a list of all levels, in ascending order. The keywords warn, error and panic are deprecated.

```
debug
info
notice
warning (warn)
err (error)
crit
alert
emerg (panic)
```

20.3.4. Actions

The default action is to send a message to the username listed as action. When the action is prefixed with a **/** then syslog will send the message to the file (which can be a regular file, but also a printer or terminal). The **@** sign prefix will send the message on to another syslog server. Here is a list of all possible actions.

```
root,user1      list of users, seperated by comma's
*               message to all logged on users
/               file (can be a printer, a console, a tty, ...)
-/              file, but don't sync after every write
|               named pipe
@               other syslog hostname
```

In addition, you can prefix actions with a **-** to omit syncing the file after every logging.

20.3.5. Configuration

Below a sample configuration of custom local4 messages in **/etc/syslog.conf**.

```
local4.crit                             /var/log/critandabove
local4.=crit                            /var/log/onlycrit
local4.*                                /var/log/alllocal4
```

Don't forget to restart the server.

```
[root@rhel4a ~]# /etc/init.d/syslog restart
Shutting down kernel logger:                                [  OK  ]
Shutting down system logger:                                [  OK  ]
Starting system logger:                                     [  OK  ]
Starting kernel logger:                                     [  OK  ]
[root@rhel4a ~]#
```

20.4. logger

The logger command can be used to generate syslog test messages. You can aslo use it in scripts. An example of testing syslogd with the **logger** tool.

```
[root@rhel4a ~]# logger -p local4.debug "l4 debug"
[root@rhel4a ~]# logger -p local4.crit "l4 crit"
[root@rhel4a ~]# logger -p local4.emerg "l4 emerg"
[root@rhel4a ~]#
```

The results of the tests with logger.

```
[root@rhel4a ~]# cat /var/log/critandabove
Feb 14 19:55:19 rhel4a paul: l4 crit
Feb 14 19:55:28 rhel4a paul: l4 emerg
[root@rhel4a ~]# cat /var/log/onlycrit
Feb 14 19:55:19 rhel4a paul: l4 crit
[root@rhel4a ~]# cat /var/log/alllocal4
Feb 14 19:55:11 rhel4a paul: l4 debug
Feb 14 19:55:19 rhel4a paul: l4 crit
Feb 14 19:55:28 rhel4a paul: l4 emerg
[root@rhel4a ~]#
```

20.5. Watching logs

You might want to use the **tail -f** command to look at the last lines of a log file. The -f option will dynamically display lines that are appended to the log. You can do the same with other commands by preceding them with the **watch** command.

20.6. Rotating logs

A lot of log files are always growing in size. To keep this within bounds, you might want to use **logrotate** to rotate, compress, remove and mail logfiles. More info on the logrotate command in the scheduling chapter.

20.7. Practice Logging

1. Display the /var/run/utmp file.

2. Display the /var/log/wtmp file.

3. Use the lastlog and lastb commands, understand the difference.

4. Examine syslog to find the location of the log file containing ssh failed logins.

5. Configure syslog to put local4.error and above messages in /var/log/l4e.log and local4.info only .info in /var/log/l4i.log. Test that it works with the logger tool!

6. Configure /var/log/Mysu.log, all the su to root messages should go in that log. Test that it works!

7. Send the local5 messages to the syslog server of your neighbour. Test that it works.

8. Write a script that executes logger to local4 every 5 seconds (different message). Use tail -f and watch on your local4 log files.

Chapter 21. Memory Management

21.1. About Memory

You can display information about RAM memory with **free -om**, **top** and cat **/proc/ meminfo**. You should understand terms like **swapping**, **paging** and **virtual memory**.

21.2. Swap space

21.2.1. About swap space

When the operating system needs more memory than physically present in RAM, it will use **swap space**. Swap space is located on slower but cheaper memory. Notice that, although hard disks are commonly used for swap space, their access times are one hundred thousand times slower.

The swap space can be a file, a partition, or a combination of files and partitions. You can see the swap space with the **free** command, or with **cat /proc/swaps**.

```
paul@RHELv4u4:~$ free -om
           total     used     free    shared   buffers    cached
Mem:         249      245        4         0       125        55
Swap:       1023        0     1023
paul@RHELv4u4:~$ cat /proc/swaps
Filename                          Type       Size    Used    Priority
/dev/mapper/VolGroup00-LogVol01   partition  1048568 0       -1
paul@RHELv4u4:~$
```

The amount of swap space that you need depends heavily on the services that the computer provides.

21.2.2. Creating a swap partition

You can activate or deactivate swap space with the **swapon** and **swapoff** commands. New swap space can be created with the **mkswap** command. The screenshot below shows the creation and activation of a swap partition.

```
root@RHELv4u4:~# fdisk -l 2> /dev/null | grep hda
Disk /dev/hda: 536 MB, 536870912 bytes
/dev/hda1               1        1040      524128+  83  Linux
root@RHELv4u4:~# mkswap /dev/hda1
Setting up swapspace version 1, size = 536702 kB
root@RHELv4u4:~# swapon /dev/hda1
```

Now you can see that **/proc/swaps** displays all swap spaces separately, whereas the **free -om** command only makes a human readable summary.

```
root@RHELv4u4:~# cat /proc/swaps
Filename                          Type         Size    Used    Priority
/dev/mapper/VolGroup00-LogVol01   partition    1048568 0       -1
/dev/hda1                         partition    524120  0       -2
root@RHELv4u4:~# free -om
          total     used    free   shared    buffers    cached
Mem:        249      245       4        0        125        54
Swap:      1535        0    1535
root@RHELv4u4:~#
```

21.2.3. Creating a swap file

Here is one more example showing you how to create a **swap file**. On Solaris you can use **mkfile** instead of **dd**.

```
root@RHELv4u4:~# dd if=/dev/zero of=/smallswapfile bs=1024 count=4096
4096+0 records in
4096+0 records out
root@RHELv4u4:~# mkswap /smallswapfile
Setting up swapspace version 1, size = 4190 kB
root@RHELv4u4:~# swapon /smallswapfile
root@RHELv4u4:~# cat /proc/swaps
Filename                          Type        Size    Used   Priority
/dev/mapper/VolGroup00-LogVol01   partition   1048568 0      -1
/dev/hda1                         partition   524120  0      -2
/smallswapfile                    file        4088    0      -3
root@RHELv4u4:~#
```

21.2.4. Swap space in /etc/fstab

If you like these swaps to be permanent, then don't forget to add them to **/etc/fstab**. The lines in /etc/fstab will be similar to the following.

```
/dev/hda1         swap       swap     defaults      0 0
/smallswapfile    swap       swap     defaults      0 0
```

21.3. Practice Memory

1. Use **dmesg** to find the total amount of memory in your computer.

2. Use **free** to display memory usage in kilobytes (then in megabytes).

3. On the Red Hat, create a swap partition on one of your new disks, and a swap file on the other new disk.

4. Put all swap spaces in /etc/fstab and activate them. Use free again to verify that it works.

Chapter 22. Installing Linux

22.1. About

The past couple of years the installation of linux has become a lot easier then before, at least for end users installing a distro like Ubuntu, Fedora, Debian or Mandrake on their home computer. Servers usually come pre-installed, and if not pre-installed, then setup of a linux server today is very easy.

Linux can be installaed in many different ways. End users most commonly use cdrom's or dvd's for installation, most of the time with a working internet connection te receive updates. Administrators might prefer network installations using protocols like **tftp**, **bootp**, **rarp** and/or **nfs** or response file solutions like **Red Hat Kickstart** or **Solaris Jumpstart**.

22.2. Installation by cdrom

Installation of linux from cdrom is easy! Most distributions ask very few questions during install (keyboard type, language, username) and detect all the hardware themselves. There is usually no need to retrieve thrid-party drivers from the internet. The GUI installation gives options like Desktop (for end users), Workstation (for developers), Server or minimal (usually without graphical interface).

22.3. Installation with rarp and tftp

Installing over the network involves powering on the machine, have it find a rarpd server to get an ip-address, then let it find an tftps server to get an installation image copied to the machine. This image can then boot. The procedure below demonstrates how to setup three Sun SPARC servers with Ubuntu Linux, using a Debian Linux machine to host the tftp, bootp and nfs daemons.

First we need to configure the mac to ip resolution in the **/etc/ethers** configuration file. Each server will receive a unique ip-address during installation.

```
root@laika:~# cat /etc/ethers
00:03:ba:02:c3:82       192.168.1.71
00:03:ba:09:7c:f9       192.168.1.72
00:03:ba:09:7f:d2       192.168.1.73
```

We need to install the rarpd and tftpd daemons on the (Debian) machine that will be hosting the install image.

```
root@laika:~# aptitude install rarpd
root@laika:~# aptitude install tftpd
```

The tftp services must be activated in inetd or xinetd.

```
root@laika:~# cat /etc/inetd.conf | tail -1
tftp dgram udp wait nobody /usr/sbin/tcpd /usr/sbin/in.tftpd /srv/tftp
```

And finally the linux install image must be present in the tftp served directory. The filename of the image must be the hex ip-address, this is accomplished with symbolic links.

```
root@laika:~# ll /srv/tftp/
total 7.5M
lrwxrwxrwx 1 root root   13 2007-03-02 21:49 C0A80147 -> ubuntu610.img
lrwxrwxrwx 1 root root   13 2007-03-03 14:13 C0A80148 -> ubuntu610.img
lrwxrwxrwx 1 root root   13 2007-03-02 21:49 C0A80149 -> ubuntu610.img
-rw-r--r-- 1 paul paul 7.5M 2007-03-02 21:42 ubuntu610.img
```

Time to enter **boot net** now in the openboot prompt. Twenty minutes later the three servers where humming with linux.

22.4. About Red Hat Kickstart

Automating Linux installations with response files can be done with **Red Hat kickstart**. One way to set it up is by using the graphical tool **/usr/sbin/system-config-kickstart**. If you prefer to set it up manually, read on.

You can modify the sample kickstart file **RH-DOCS/sample.ks** (can be found on the documentation dvd). Put this file so **anaconda** can read it.

Anaconda is the Red Ha installer written in Python. The name is chose because anacondas are lizard-eating pythons. Lizard is the name of the Caldera Linux installation program.

Another option is to start with the **/root/anaconda-ks.cfg** file. This is a sample kickstart file that contains all the settings from your current installation.

Do not change the order of the sections inside your kickstart file! The Red Hat System Administration Guide contains about 25 pages describing all the options, most of them are easy ti understand if you already performed a couple of installations.

22.5. Using Kickstart

To use kickstart, name your kickstart file **ks.cfg** and put it in the root directory of your installation cdrom (or on a usb stick or a floppy). For network based installations, name the file **$ip-address-kickstart** and place the following in **dhcpd.conf**.

```
filename "/export/kickstart"
next-server remote.installation.server
```

Leaving out the **next-server** line will result in the client looking for the file on the dhcp server itself.

Booting from cdrom with kickstart reauires the following commnad at the **boot:** prompt.

```
linux ks=cdrom:/ks.cfg
```

When the kickstart file is on the network, use nfs or http like in these examples.

```
linux ks=nfs:servername:/path/to/ks.cfg
```

```
linux ks=http://servername/path/to/ks.cfg
```

Chapter 23. Package Management

23.1. About repositories

Software for your Linux distribution is not scattered all over the place, it is in general managed in a central distributed repository. This means that applications in the repository are tested for your distribution. Installing this software is very easy. The problem begins when you need software from outside of the central repository.

You can install software from the repository on Linux in different ways. Beginners should use the graphical software installation tool that is provided by the distribution (Synaptic on Debian, Add/Remove Software on Ubuntu, Yast on Suse, ...). More advanced people tend to use the command line (rpm, yum, dpkg, aptitude). A third option is to download vanilla source code and compile the software yourself, providing the application is open source.

23.2. Red Hat Package Manager

The **Red Hat Package Manager** is used by many distributions. Besides Red Hat, who originally created the **.rpm** format and Fedora, there is also Mandriva, Red Flag Linux, OpenSUSE and others. The most common front ends (besides **rpm**) using this tool are **yum**, **YaST** and **up2date**.

23.2.1. rpm

23.2.1.1. About rpm

The **Red Hat Package Manager** can be used on the command line with **rpm** or in a graphical way going to Applications--System Settings--Add/Remove Applications. Type **rpm --help** to see some of the options.

Software distributed in the rpm format will be named **foo-version.platform.rpm** .

23.2.1.2. rpm -qa

To obtain a list of all installed software, use the **rpm -qa** command.

```
[root@RHEL52 ~]# rpm -qa | grep samba
system-config-samba-1.2.39-1.el5
samba-3.0.28-1.el5_2.1
samba-client-3.0.28-1.el5_2.1
samba-common-3.0.28-1.el5_2.1
```

23.2.1.3. rpm -q

To verify whether one package is installed, use **rpm -q**.

```
root@RHELv4u4:~# rpm -q gcc
gcc-3.4.6-3
root@RHELv4u4:~# rpm -q laika
package laika is not installed
```

23.2.1.4. rpm -q --redhatprovides

To check whether a package is provided by Red Hat, use the **--redhatprovides** option.

```
root@RHELv4u4:~# rpm -q --redhatprovides bash
bash-3.0-19.3
root@RHELv4u4:~# rpm -q --redhatprovides gcc
gcc-3.4.6-3
root@RHELv4u4:~# rpm -q --redhatprovides laika
no package provides laika
```

23.2.1.5. rpm -Uvh

To install or upgrade a package, use the -Uvh switches. The -U switch is the same as -i for install, except that older versions of the software are removed. The -vh switches are for nicer output.

```
root@RHELv4u4:~# rpm -Uvh gcc-3.4.6-3
```

23.2.1.6. rpm -e

To remove a package, use the -e switch.

```
root@RHELv4u4:~# rpm -e gcc-3.4.6-3
```

rpm -e verifies dependencies, and thus will prevent you from accidentaly erasing packages that are needed by other packages.

```
[root@RHEL52 ~]# rpm -e gcc-4.1.2-42.el5
error: Failed dependencies:
gcc = 4.1.2-42.el5 is needed by (installed) gcc-c++-4.1.2-42.el5.i386
gcc = 4.1.2-42.el5 is needed by (installed) gcc-gfortran-4.1.2-42.el5.i386
gcc is needed by (installed) systemtap-0.6.2-1.el5_2.2.i386
```

23.2.1.7. /var/lib/rpm

The **rpm** database is located at **/var/lib/rpm**. This database contains all meta information about packages that are installed (via rpm). It keeps track of all files, which enables complete removes of software.

23.2.2. yum

The **Yellowdog Updater, Modified (yum)** is an easier command to work with rpm packages. It is installed by default on Fedora and is optional on Red Hat Enterprise Linux.

Issue **yum list available** to see a list of available packages.

```
yum list available
```

To search for a package containing a certain string in the description or name use **yum search $string**.

```
yum search $string
```

To install an application, use **yum install $package**. Naturally yum will install all the necesary dependencies.

```
yum install $package
```

To bring all applications up to date, by downloading and installing them, issue **yum update**. All software that was installed via yum will be updated to the latest version that is available in the repository.

```
yum update
```

The configuration of your yum repositories is done in **/etc/yum/yum.conf** and **/etc/yum/repos.d/**.

23.2.3. up2date

up2date is the Red Hat Update Agent. It is available on Red Hat Enterprise Linux and serves as a connection to RHN (Red Hat Network). It has simple switches like -d for download, -i for install and -l for list (of packages that can be updated).

23.3. Debian Package Management

23.3.1. About .deb

The Debian Package Management System is used by Debian, Ubuntu, Linux Mint and all derivatives from Debian and Ubuntu. It uses **.deb** packages.

23.3.2. dpkg -l

The low level tool to work with **.deb** packages is **dpkg**. Here you see how to obtain a list of all installed packages. The ii at the beginning means the package is installed.

```
root@laika:~# dpkg -l | grep gcc-4.2
ii gcc-4.2      4.2.4-1ubuntu3 The GNU C compiler
ii gcc-4.2-base 4.2.4-1ubuntu3 The GNU Compiler Collection (base package)
```

23.3.3. dpkg

You could use **dpkg -i** to install a package and **dpkg -r** to remove a package, but you'd have to manually keep track of dependencies.

23.3.4. aptitude

Most people use **aptitude** for package management on Debian and Ubuntu Systems.

To synchronize with the repositories.

```
aptitude update
```

To patch and upgrade all software to the latest version..

```
aptitude safe-upgrade
```

To install an application with all dependencies.

```
aptitude install $package
```

To search the repositories for applications that contain a certain string in their name or description.

```
aptitude search $string
```

To remove an application and all unused files.

```
aptitude remove $package
```

23.4. Downloading software

First and most important, whenever you download software, start by reading the README file!

Normally the readme will explain what to do after download. You will probably receive a .tar.gz or a .tgz file. Read the documentation, then put the compressed file in a directory. You can use the following to find out where the package wants to install.

```
tar tvzpf $downloadedFile.tgz
```

You unpack them like with **tar xzf**, it will create a directory called applicationName-1.2.3

```
tar xzf $applicationName.tgz
```

Replace the z with a j when the file ends in .tar.bz2. The **tar**, gzip and bzip2 commands are explained in detail later.

If you download a .deb file, then you'll have to use dpkg to install it, .rpm's can be installed with the rpm command. Sometimes people use the **alien** command to convert between package formats.

23.5. Compiling software

First and most important, whenever you download source code for installation, start by reading the README file!

Usually the steps are always the same three : running **./configure** followed by **make** (which is the actual compiling) and then by **make install** to copy the files to their proper location.

```
./configure
make
make install
```

23.6. Practice: Installing software

1. Find the Graphical application on all computers to add and remove applications.

2. Verify on both systems whether gcc is installed.

3. Use aptitude or yum to search for and install the 'dict', 'samba' and 'wesnoth' applications. Did you find all them all ?

4. Search the internet for 'webmin' and install it.

5. If time permits, uninstall Samba from the ubuntu machine, download the latest version from samba.org and install it.

23.7. Solution: Installing software

1. Find the Graphical application on all computers to add and remove applications.

2. Verify on both systems whether gcc is installed.

```
dpkg -l | grep gcc
```

```
rpm -qa | grep gcc
```

3. Use aptitude or yum to search for and install the 'dict', 'samba' and 'wesnoth' applications. Did you find all them all ?

```
aptitude search wesnoth (Debian, Ubuntu and family)
```

```
yum search wesnoth (Red Hat and family)
```

4. Search the internet for 'webmin' and install it.

```
Google should point you to webmin.com.
```

```
There are several formats available there choose .rpm, .deb or .tgz .
```

5. If time permits, uninstall Samba from the ubuntu machine, download the latest version from samba.org and install it.

Chapter 24. Backup

24.1. About tape devices

Don't forget that the name of a device strictly speaking has no meaning since the kernel will use the major and minor number to find the hardware! See the man page of **mknod** and the devices.txt file in the linux kernel source for more info.

24.1.1. SCSI tapes

On the official Linux device list (http://www.lanana.org/docs/device-list/) we find the names for SCSI tapes (major 9 char). SCSI tape devices are located underneath **/dev/st** and are numbered starting with 0 for the first tape device.

```
/dev/st0   First tape device
/dev/st1   Second tape device
/dev/st2   Third tape device
```

To prevent **automatic rewinding of tapes**, prefix them with the letter n.

```
/dev/nst0   First no rewind tape device
/dev/nst1   Second no rewind tape device
/dev/nst2   Third no rewind tape device
```

By default, SCSI tapes on linux will use the highest hardware compression that is supported by the tape device. To lower the compression level, append one of the letters l (low), m (medium) or a (auto) to the tape name.

```
/dev/st0l   First low compression tape device
/dev/st0m   First medium compression tape device
/dev/nst2m  Third no rewind medium compression tape device
```

24.1.2. IDE tapes

On the official Linux device list (http://www.lanana.org/docs/device-list/) we find the names for IDE tapes (major 37 char). IDE tape devices are located underneath **/dev/ht** and are numbered starting with 0 for the first tape device. No rewind and compression is similar to SCSI tapes.

```
/dev/ht0   First IDE tape device
/dev/nht0  Second no rewind IDE tape device
/dev/ht0m  First medium compression IDE tape device
```

24.1.3. mt

To manage your tapes, use **mt** (Magnetic Tape). Some examples.

To receive information about the status of the tape.

```
mt -f /dev/st0 status
```

To rewind a tape...

```
mt -f /dev/st0 rewind
```

To rewind and eject a tape...

```
mt -f /dev/st0 eject
```

To erase a tape...

```
mt -f /dev/st0 erase
```

24.2. Compression

It can be beneficial to compress files before backup. The two most popular tools for compression of regular files on linux are **gzip/gunzip** and **bzip2/bunzip2**. Below you can see gzip in action, notice that it adds the **.gz** extension to the file.

```
paul@RHELv4u4:~/test$ ls -l allfiles.tx*
-rw-rw-r--  1 paul paul 8813553 Feb 27 05:38 allfiles.txt
paul@RHELv4u4:~/test$ gzip allfiles.txt
paul@RHELv4u4:~/test$ ls -l allfiles.tx*
-rw-rw-r--  1 paul paul 931863 Feb 27 05:38 allfiles.txt.gz
paul@RHELv4u4:~/test$ gunzip allfiles.txt.gz
paul@RHELv4u4:~/test$ ls -l allfiles.tx*
-rw-rw-r--  1 paul paul 8813553 Feb 27 05:38 allfiles.txt
paul@RHELv4u4:~/test$
```

In general, gzip is much faster than bzip2, but the latter one compresses a lot better. Let us compare the two.

```
paul@RHELv4u4:~/test$ cp allfiles.txt bllfiles.txt
paul@RHELv4u4:~/test$ time gzip allfiles.txt

real    0m0.050s
user    0m0.041s
sys     0m0.009s
paul@RHELv4u4:~/test$ time bzip2 bllfiles.txt

real    0m5.968s
user    0m5.794s
sys     0m0.076s
paul@RHELv4u4:~/test$ ls -l ?llfiles.tx*
-rw-rw-r--  1 paul paul 931863 Feb 27 05:38 allfiles.txt.gz
-rw-rw-r--  1 paul paul 708871 May 12 10:52 bllfiles.txt.bz2
paul@RHELv4u4:~/test$
```

24.3. tar

The **tar** utility gets its name from **Tape ARchive**. This tool will receive and send files to a destination (typically a tape or a regular file). The c option is used to create a tar archive (or tarfile), the f option to name/create the **tarfile**. The example below takes a backup of /etc into the file /backup/etc.tar .

```
root@RHELv4u4:~# tar cf /backup/etc.tar /etc
root@RHELv4u4:~# ls -l /backup/etc.tar
-rw-r--r--  1 root root 47800320 May 12 11:47 /backup/etc.tar
root@RHELv4u4:~#
```

Compression can be achieved without pipes since tar uses the z flag to compress with gzip, and the j flag to compress with bzip2.

```
root@RHELv4u4:~# tar czf /backup/etc.tar.gz /etc
root@RHELv4u4:~# tar cjf /backup/etc.tar.bz2 /etc
root@RHELv4u4:~# ls -l /backup/etc.ta*
-rw-r--r--  1 root root 47800320 May 12 11:47 /backup/etc.tar
-rw-r--r--  1 root root  6077340 May 12 11:48 /backup/etc.tar.bz2
-rw-r--r--  1 root root  8496607 May 12 11:47 /backup/etc.tar.gz
root@RHELv4u4:~#
```

The t option is used to **list the contents of a tar file**. Verbose mode is enabled with v (also useful when you want to see the files being archived during archiving).

```
root@RHELv4u4:~# tar tvf /backup/etc.tar
drwxr-xr-x root/root         0 2007-05-12 09:38:21 etc/
-rw-r--r-- root/root      2657 2004-09-27 10:15:03 etc/warnquota.conf
-rw-r--r-- root/root     13136 2006-11-03 17:34:50 etc/mime.types
drwxr-xr-x root/root         0 2004-11-03 13:35:50 etc/sound/
...
```

To **list a specific file in a tar archive**, use the t option, added with the filename (without leading /).

```
root@RHELv4u4:~# tar tvf /backup/etc.tar etc/resolv.conf
-rw-r--r-- root/root        77 2007-05-12 08:31:32 etc/resolv.conf
root@RHELv4u4:~#
```

Use the x flag to **restore a tar archive**, or a single file from the archive. Remember that by default tar will restore the file in the current directory.

```
root@RHELv4u4:~# tar xvf /backup/etc.tar etc/resolv.conf
etc/resolv.conf
root@RHELv4u4:~# ls -l /etc/resolv.conf
-rw-r--r--  2 root root 40 May 12 12:05 /etc/resolv.conf
```

```
root@RHELv4u4:~# ls -l etc/resolv.conf
-rw-r--r--  1 root root 77 May 12 08:31 etc/resolv.conf
root@RHELv4u4:~#
```

You can **preserve file permissions** with the p flag. And you can exclude directories or file with **--exclude**.

```
root ~# tar cpzf /backup/etc_with_perms.tgz /etc
root ~# tar cpzf /backup/etc_no_sysconf.tgz /etc --exclude /etc/sysconfig
root ~# ls -l /backup/etc_*
-rw-r--r--  1 root root 8434293 May 12 12:48 /backup/etc_no_sysconf.tgz
-rw-r--r--  1 root root 8496591 May 12 12:48 /backup/etc_with_perms.tgz
root ~#
```

You can also create a text file with names of files and directories to archive, and then supply this file to tar with the -T flag.

```
root@RHELv4u4:~# find /etc -name *.conf > files_to_archive.txt
root@RHELv4u4:~# echo /home -iname *.pdf >> files_to_archive.txt
root@RHELv4u4:~# tar cpzf /backup/backup.tgz -T files_to_archive.txt
```

The tar utility can receive filenames from the find command, with the help of xargs.

```
find /etc -type f -name "*.conf" | xargs tar czf /backup/confs.tar.gz
```

You can also use tar to copy a directory, this is more efficient than using cp -r.

```
(cd /etc; tar -cf - . ) | (cd /backup/copy_of_etc/; tar -xpf - )
```

Another example of tar, this copies a directory securely over the network.

```
(cd /etc;tar -cf - . )|(ssh user@srv 'cd /backup/cp_of_etc/; tar -xf - ')
```

tar can be used together with gzip and copy a file to a remote server through ssh

```
cat backup.tar | gzip | ssh bashuser@192.168.1.105 "cat - > backup.tgz"
```

Compress the tar backup when it is on the network, but leave it uncompressed at the destination.

```
cat backup.tar | gzip | ssh user@192.168.1.105 "gunzip|cat - > backup.tar"
```

Same as the previous, but let ssh handle the compression

```
cat backup.tar | ssh -C bashuser@192.168.1.105 "cat - > backup.tar"
```

24.4. Backup Types

Linux uses **multilevel incremental** backups using distinct levels. A full backup is a backup at level 0. A higher level x backup will include all changes since the last level x-1 backup.

Suppose you take a full backup on Monday (level 0) and a level 1 backup on Tuesday, then the Tuesday backup will contain all changes since Monday. Taking a level 2 on Wednesday will contain all changes since Tuesday (the last level 2-1). A level 3 backup on Thursday will contain all changes since Wednesday (the last level 3-1). Another level 3 on Friday will also contain all changes since Wednesday. A level 2 backup on Saturday would take all changes since the last level 1 from Tuesday.

24.5. dump and restore

While **dump** is similar to tar, it is also very different because it looks at the file system. Where tar receives a lists of files to backup, dump will find files to backup by itself by examining ext2. Files found by dump will be copied to a tape or regular file. In case the target is not big enough to hold the dump (end-of-media), it is broken into multiple volumes.

Restoring files that were backed up with dump is done with the **restore** command. In the example below we take a full level 0 backup of two partitions to a SCSI tape. The no rewind is mandatory to put the volumes behind each other on the tape.

```
dump 0f /dev/nst0 /boot
dump 0f /dev/nst0 /
```

Listing files in a dump archive is done with **dump -t**, and you can compare files with **dump -C**.

You can omit files from a dump by changing the dump attribute with the **chattr** command. The d attribute on ext will tell dump to skip the file, even during a full backup. In the following example, /etc/hosts is excluded from dump archives.

```
chattr +d /etc/hosts
```

To restore the complete file system with **restore**, use the -r option. This can be useful to change the size or block size of a file system. You should have a clean file system mounted and cd'd into it. Like this example shows.

```
mke2fs /dev/hda3
mount /dev/hda3 /mnt/data
cd /mnt/data
```

```
restore rf /dev/nst0
```

To extract only one file or directory from a dump, use the -x option.

```
restore -xf /dev/st0 /etc
```

24.6. cpio

Different from tar and dump is **cpio** (Copy Input and Output). It can be used to receive filenames, but copies the actual files. This makes it an easy companion with find! Some examples below.

find sends filenames to cpio, which puts the files in an archive.

```
find /etc -depth -print | cpio -oaV -O archive.cpio
```

The same, but compressed with gzip

```
find /etc -depth -print | cpio -oaV | gzip -c > archive.cpio.gz
```

Now pipe it through ssh (backup files to a compressed file on another machine)

```
find /etc -depth -print|cpio -oaV|gzip -c|ssh server "cat - > etc.cpio.gz"
```

find sends filenames to cpio | cpio sends files to ssh | ssh sends files to cpio 'cpio extracts files'

```
find /etc -depth -print | cpio -oaV | ssh user@host 'cpio -imVd'
```

the same but reversed: copy a dir from the remote host to the local machine

```
ssh user@host "find path -depth -print | cpio -oaV" | cpio -imVd
```

24.7. dd

24.7.1. About dd

Some people use **dd** to create backups. This can be very powerful, but dd backups can only be restored to very similar partitions or devices. There are however a lot of useful things possible with dd. Some examples.

24.7.2. Create a CDROM image

The easiest way to create a **.ISO file** from any CD. The if switch means Input File, of is the Output File. Any good tool can burn a copy of the CD with this .ISO file.

```
dd if=/dev/cdrom of=/path/to/cdrom.ISO
```

24.7.3. Create a floppy image

A little outdated maybe, but just in case : make an image file from a 1.44MB floppy. Blocksize is defined by bs, and count contains the number of blocks to copy.

```
dd if=/dev/floppy of=/path/to/floppy.img bs=1024 count=1440
```

24.7.4. Copy the master boot record

Use dd to copy the **MBR** (Master Boot Record) of hard disk /dev/hda to a file.

```
dd if=/dev/hda of=/MBR.img bs=512 count=1
```

24.7.5. Copy files

This example shows how dd can copy files. Copy the file summer.txt to copy_of_summer.txt .

```
dd if=~/summer.txt of=~/copy_of_summer.txt
```

24.7.6. Image disks or partitions

And who needs ghost when dd can create a (compressed) image of a partition.

```
dd if=/dev/hdb2 of=/image_of_hdb2.IMG
dd if=/dev/hdb2 | gzip > /image_of_hdb2.IMG.gz
```

24.7.7. Create files of a certain size

dd can be used to create a file of any size. The first example creates a one MEBIbyte file, the second a one MEGAbyte file.

```
dd if=/dev/zero of=file1MB count=1024 bs=1024
dd if=/dev/zero of=file1MB count=1000 bs=1024
```

24.7.8. CDROM server example

And there are of course endless combinations with ssh and bzip2. This example puts a bzip2 backup of a cdrom on a remote server.

```
dd if=/dev/cdrom |bzip2|ssh user@host "cat - > /backups/cd/cdrom.iso.bz2"
```

24.8. split

The **split** command is useful to split files into smaller files. This can be useful to fit the file onto multiple instances of a medium too small to contain the complete file. In the example below, a file of size 5000 bytes is split into three smaller files, with maximum 2000 bytes each.

```
paul@laika:~/test$ ls -l
total 8
-rw-r--r-- 1 paul paul 5000 2007-09-09 20:46 bigfile1
paul@laika:~/test$ split -b 2000 bigfile1 splitfile.
paul@laika:~/test$ ls -l
total 20
-rw-r--r-- 1 paul paul 5000 2007-09-09 20:46 bigfile1
-rw-r--r-- 1 paul paul 2000 2007-09-09 20:47 splitfile.aa
-rw-r--r-- 1 paul paul 2000 2007-09-09 20:47 splitfile.ab
-rw-r--r-- 1 paul paul 1000 2007-09-09 20:47 splitfile.ac
```

24.9. Practice backup

!! Careful with tar options and the position of the backup file, mistakes can destroy your system!!

1. Create a directory (or partition if you like) for backups. Link (or mount) it under /mnt/backup.

2a. Use tar to backup /etc in /mnt/backup/etc_date.tgz, the backup must be gzipped. (Replace date with the current date)

2b. Use tar to backup /bin to /mnt/backup/bin_date.tar.bz2, the backup must be bzip2'd.

2c. Choose a file in /etc and /bin and verify with tar that the file is indeed backed up.

2d. Extract those two files to your home directory.

3a. Create a backup directory for your neighbour, make it accessible under /mnt/neighbourName

3b. Combine ssh and tar to put a backup of your /boot on your neighbours computer in /mnt/YourName

4a. Combine find and cpio to create a cpio archive of /etc.

4b. Choose a file in /etc and restore it from the cpio archive into your home directory.

5. Use dd and ssh to put a backup of the master boot record on your neighbours computer.

6. (On the real computer) Create and mount an ISO image of the ubuntu cdrom.

7. Combine dd and gzip to create a 'ghost' image of one of your partitions on another partition.

8. Use dd to create a five megabyte file in ~/testsplit and name it biggest. Then split this file in smaller two megabyte parts.

```
mkdir testsplit

dd if=/dev/zero of=~/testsplit/biggest count=5000 bs=1024

split -b 2000000 biggest parts
```

Chapter 25. Performance Monitoring

25.1. About Monitoring

Monitoring means obtaining information about the utilization of memory, CPU power, bandwidth and storage. You should start monitoring your system as soon as possible, to be able to create a **baseline**. Make sure that you get to know your system. *Boys, just give your computer a girls name and get to know her.* The baseline is important, it allows you to see a steady growth in CPU utilization or a steady decline in free disk space. It will allow you to plan for scaling up or scaling out.

Let us look at some tools that go beyond **ps fax**, **df -h**, **lspci**, **fdisk -l** and **du -sh**.

25.2. top

To start monitoring, you can use **top**. This tool will monitor Memory, CPU and running processes. Top will automatically refresh. Inside top you can use many commands, like **k** to kill processes, or **t** and **m** to toggle displaying task and memory information, or the number **1** to have one line per cpu, or one summary line for all cpu's.

```
top - 12:23:16 up 2 days,  4:01, 2 users, load average: 0.00, 0.00, 0.00
Tasks:  61 total,   1 running,  60 sleeping,   0 stopped,   0 zombie
Cpu(s):  0.3% us,  0.5% sy, 0.0% ni, 98.9% id, 0.2% wa, 0.0% hi, 0.0% si
Mem:    255972k total,   240952k used,    15020k free,    59024k buffers
Swap:   524280k total,      144k used,   524136k free,   112356k cached

PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
 1 root      16   0  2816  560  480 S  0.0  0.2   0:00.91 init
 2 root      34  19     0    0    0 S  0.0  0.0   0:00.01 ksoftirqd/0
 3 root       5 -10     0    0    0 S  0.0  0.0   0:00.57 events/0
 4 root       5 -10     0    0    0 S  0.0  0.0   0:00.00 khelper
 5 root      15 -10     0    0    0 S  0.0  0.0   0:00.00 kacpid
16 root       5 -10     0    0    0 S  0.0  0.0   0:00.08 kblockd/0
26 root      15   0     0    0    0 S  0.0  0.0   0:02.86 pdflush
...
```

You can customize top to display the columns of your choice, or to display only the processes that you find interesting.

```
[paul@RHELv4u3 ~]$ top p 3456 p 8732 p 9654
```

25.3. free

The **free** command is common on Linux to monitor free memory. You can use free to display information every x seconds, but the output is not ideal.

```
[paul@RHELv4u3 gen]$ free -om -s 10
total       used       free     shared    buffers     cached
Mem:        249        222          27          0          50         109
Swap:       511          0         511

total       used       free     shared    buffers     cached
Mem:        249        222          27          0          50         109
Swap:       511          0         511

[paul@RHELv4u3 gen]$
```

25.4. watch

It might be more interesting to combine free with the **watch** program. This program can also run commands with a delay, and can highlight changes (with the -d switch).

```
[paul@RHELv4u3 ~]$ watch -d -n 3 free -om
...
Every 3.0s: free -om                             Sat Jan 27 12:13:03 2007

total       used       free     shared    buffers     cached
Mem:           249        230          19          0          56         109
Swap:          511          0         511
```

25.5. vmstat

To monitor CPU, disk and memory statistics in one line there is **vmstat**. The screenshot below shows vmstat running every two seconds 100 times (or until the Ctrl-C). Below the r, you see the number of processes waiting for the CPU, sleeping processes go below b. Swap usage (swpd) stayed constant at 144 kilobytes, free memory dropped from 16.7MB to 12.9MB. See man vmstat for the rest

```
[paul@RHELv4u3 ~]$ vmstat 2 100
procs ----------memory--------- --swap-- ---io--- --system-- ---cpu----
r  b   swpd   free   buff  cache  si  so   bi   bo   in    cs us sy id wa
0  0    144  16708  58212 111612   0   0    3    4   75    62  0  1 99  0
0  0    144  16708  58212 111612   0   0    0    0  976    22  0  0 100 0
0  0    144  16708  58212 111612   0   0    0    0  958    14  0  1 99  0
1  0    144  16528  58212 111612   0   0    0   18 1432  7417  1 32 66  0
1  0    144  16468  58212 111612   0   0    0    0 2910 20048  4 95  1  0
1  0    144  16408  58212 111612   0   0    0    0 3210 19509  4 97  0  0
1  0    144  15568  58816 111612   0   0  300 1632 2423 10189  2 62  0 36
0  1    144  13648  60324 111612   0   0  754    0 1910  2843  1 27  0 72
0  0    144  12928  60948 111612   0   0  312  418 1346  1258  0 14 57 29
0  0    144  12928  60948 111612   0   0    0    0  977    19  0  0 100 0
0  0    144  12988  60948 111612   0   0    0    0  977    15  0  0 100 0
0  0    144  12988  60948 111612   0   0    0    0  978    18  0  0 100 0

[paul@RHELv4u3 ~]$
```

25.6. iostat

The **iostat** tool can display disk and cpu statistics. The -d switch below makes iostat only display disk information (500 times every two seconds). The first block displays statistics since the last reboot.

```
[paul@RHELv4u3 ~]$ iostat -d 2 500
Linux 2.6.9-34.EL (RHELv4u3.localdomain)        01/27/2007

Device:          tps   Blk_read/s   Blk_wrtn/s   Blk_read   Blk_wrtn
hdc             0.00         0.01         0.00       1080          0
sda             0.52         5.07         7.78     941798    1445148
sda1            0.00         0.01         0.00        968          4
sda2            1.13         5.06         7.78     939862    1445144
dm-0            1.13         5.05         7.77     939034    1444856
dm-1            0.00         0.00         0.00        360        288

Device:          tps   Blk_read/s   Blk_wrtn/s   Blk_read   Blk_wrtn
hdc             0.00         0.00         0.00          0          0
sda             0.00         0.00         0.00          0          0
sda1            0.00         0.00         0.00          0          0
sda2            0.00         0.00         0.00          0          0
dm-0            0.00         0.00         0.00          0          0
dm-1            0.00         0.00         0.00          0          0
...
[paul@RHELv4u3 ~]$
```

You can have more statistics using **iostat -d -x**, or display only cpu statistics with **iostat -c**.

```
[paul@RHELv4u3 ~]$ iostat -c 5 500
Linux 2.6.9-34.EL (RHELv4u3.localdomain)        01/27/2007

avg-cpu:  %user   %nice    %sys %iowait   %idle
0.31    0.02    0.52    0.23   98.92

avg-cpu:  %user   %nice    %sys %iowait   %idle
0.62    0.00   52.16   47.23    0.00

avg-cpu:  %user   %nice    %sys %iowait   %idle
2.92    0.00   36.95   60.13    0.00

avg-cpu:  %user   %nice    %sys %iowait   %idle
0.63    0.00   36.63   62.32    0.42

avg-cpu:  %user   %nice    %sys %iowait   %idle
0.00    0.00    0.20    0.20   99.59

 [paul@RHELv4u3 ~]$
```

25.7. mpstat

On multi-processor machines, **mpstat** can display statistics for all, or for a selected cpu.

```
paul@laika:~$ mpstat -P ALL
Linux 2.6.20-3-generic (laika)  02/09/2007

CPU %user   %nice   %sys %iowait   %irq   %soft  %steal   %idle   intr/s
all  1.77   0.03   1.37    1.03   0.02    0.39    0.00   95.40  1304.91
  0  1.73   0.02   1.47    1.93   0.04    0.77    0.00   94.04  1304.91
  1  1.81   0.03   1.27    0.13   0.00    0.00    0.00   96.76     0.00
paul@laika:~$
```

25.8. sadc and sar

The **sadc** tool writes system utilization data to **/var/log/sa/sa??**, where ?? is replaced with the current day of the month. By default, cron runs the **sal** script every 10 minutes, the sal script runs sadc for one second. Just before midnight every day, cron runs the **sa2** script, which in turn invokes **sar**. The sar tool will read the daily data generated by sadc and put it in /var/log/sa/sar??. These **sar reports** contain a lot of statistics.

You can also use sar to display a portion of the statistics that were gathered. Like this example for cpu statistics.

```
[paul@RHELv4u3 sa]$ sar -u | head
Linux 2.6.9-34.EL (RHELv4u3.localdomain)        01/27/2007

12:00:01 AM       CPU     %user     %nice   %system   %iowait    %idle
12:10:01 AM       all      0.48      0.01      0.60      0.04    98.87
12:20:01 AM       all      0.49      0.01      0.60      0.06    98.84
12:30:01 AM       all      0.49      0.01      0.64      0.25    98.62
12:40:02 AM       all      0.44      0.01      0.62      0.07    98.86
12:50:01 AM       all      0.42      0.01      0.60      0.10    98.87
01:00:01 AM       all      0.47      0.01      0.65      0.08    98.80
01:10:01 AM       all      0.45      0.01      0.68      0.08    98.78
[paul@RHELv4u3 sa]$
```

There are other useful sar options, like **sar -I PROC** to display interrupt activity per interrupt and per CPU, or **sar -r** for memory related statistics. Check the manual page of sar for more.

25.9. ntop

The **ntop** tool is not present in default Red Hat installs. Once run, it will generate a very extensive analysis of network traffic in html on http://localhost:3000 .

25.10. iftop

The **iftop** tool will display bandwidth by socket statistics for a specific network device. Not available on default Red Hat servers.

```
             1.91Mb        3.81Mb         5.72Mb        7.63Mb   9.54Mb
--------------|-------------|--------------|-------------|--------|----
laika.local        => barry                        4.94Kb  6.65Kb  69.9Kb
                   <=                              7.41Kb  16.4Kb   766Kb
laika.local        => ik-in-f19.google.com            0b   1.58Kb  14.4Kb
                   <=                                 0b    292b   41.0Kb
laika.local        => ik-in-f99.google.com            0b     83b   4.01Kb
                   <=                                 0b     83b   39.8Kb
laika.local        => ug-in-f189.google.com           0b     42b    664b
                   <=                                 0b     42b    406b
laika.local        => 10.0.0.138                      0b      0b    149b
                   <=                                 0b      0b    256b
laika.local        => 224.0.0.251                     0b      0b     86b
                   <=                                 0b      0b      0b
laika.local        => ik-in-f83.google.com            0b      0b     39b
                   <=                                 0b      0b     21b
```

Chapter 26. User Quota's

26.1. About Disk Quotas

To limit the disk space used by user, you can set up **disk quotas**. This requires adding **usrquota** and/or **grpquota** to one or more of the file systems in **/etc/fstab**.

```
root@RHELv4u4:~# cat /etc/fstab | grep usrquota
/dev/VolGroup00/LogVol02     /home     ext3     usrquota,grpquota   0 0
```

Next you need to remount the file system.

```
root@RHELv4u4:~# mount -o remount /home
```

The next step is to build the **quota.user** and/or **quota.group** files. These files (called the **quota files**) contain the table of the disk usage on that file system. Use the **quotacheck** command to accomplish this.

```
root@RHELv4u4:~# quotacheck -cug /home
root@RHELv4u4:~# quotacheck -avug
```

The **-c** is for create, **u** for user quota, **g** for group, **a** for checking all quota enabled file systems in /etc/fstab and **v** for verbose information. The next step is to edit individual user quotas with **edquota** or set a general quota on the file system with **edquota -t**. The tool will enable you to put **hard** (this is the real limit) and **soft** (allows a grace period) limits on **blocks** and **inodes**. The **quota** command will verify that quota for a user is set. You can have a nice overview with **repquota**.

The final step (before your users start complaining about lack of disk space) is to enable quotas with **quotaon(1)**.

```
root@RHELv4u4:~# quotaon -vaug
```

Issue the **quotaoff** command to stop all complaints.

```
root@RHELv4u4:~# quotaoff -vaug
```

26.2. Practice Disk quotas

1. Implement disk quotas on one of your new partitions. Limit one of your users to 10 megabyte.

2. Test that they work by copying many files to the quota'd partition.

Chapter 27. VNC

27.1. About VNC

VNC can be configured in gnome or KDE using the **Remote Desktop Preferences**. **VNC** can be used to run your desktop on another computer, and you can also use it to see and take over the Desktop of another user. The last part can be useful for help desks to show users how to do things. VNC has the added advantage of being operating system independent, a lot of products (realvnc, tightvnc, xvnc, ...) use the same protocol on Solaris, Linux, BSD and more.

27.2. VNC Server

Starting the vnc server for the first time.

```
[root@RHELv4u3 conf]# rpm -qa | grep -i vnc
vnc-server-4.0-8.1
vnc-4.0-8.1
[root@RHELv4u3 conf]# vncserver :2

You will require a password to access your desktops.

Password:
Verify:
xauth:  creating new authority file /root/.Xauthority

New 'RHELv4u3.localdomain:2 (root)' desktop is RHELv4u3.localdomain:2

Creating default startup script /root/.vnc/xstartup
Starting applications specified in /root/.vnc/xstartup
Log file is /root/.vnc/RHELv4u3.localdomain:2.log

[root@RHELv4u3 conf]#
```

27.3. VNC Client

You can now use the **vncviewer** from another machine to connect to your vnc server. It will default to a very simple graphical interface...

```
paul@laika:~$ vncviewer 192.168.1.49:2
VNC viewer version 3.3.7 - built Nov 20 2006 13:05:04
Copyright (C) 2002-2003 RealVNC Ltd.
Copyright (C) 1994-2000 AT&T Laboratories Cambridge.
See http://www.realvnc.com for information on VNC.
VNC server supports protocol version 3.8 (viewer 3.3)
Password:
VNC authentication succeeded
Desktop name "RHELv4u3.localdomain:2 (root)"
Connected to VNC server, using protocol version 3.3
...
```

If you don't like the simple twm window manager, you can comment out the last two lines of **~/.vnc/xstartup** and add a **gnome-session &** line to have vnc default to gnome instead.

```
[root@RHELv4u3 ~]# cat .vnc/xstartup
#!/bin/sh

# Uncomment the following two lines for normal desktop:
# unset SESSION_MANAGER
# exec /etc/X11/xinit/xinitrc

[ -x /etc/vnc/xstartup ] && exec /etc/vnc/xstartup
[ -r $HOME/.Xresources ] && xrdb $HOME/.Xresources
xsetroot -solid grey
vncconfig -iconic &
# xterm -geometry 80x24+10+10 -ls -title "$VNCDESKTOP Desktop" &
# twm &
gnome-session &
[root@RHELv4u3 ~]#
```

Don't forget to restart your vnc server after changing this file.

```
[root@RHELv4u3 ~]# vncserver -kill :2
Killing Xvnc process ID 5785
[root@RHELv4u3 ~]# vncserver :2

New 'RHELv4u3.localdomain:2 (root)' desktop is RHELv4u3.localdomain:2

Starting applications specified in /root/.vnc/xstartup
Log file is /root/.vnc/RHELv4u3.localdomain:2.log
```

27.4. Practive VNC

1. Use VNC to connect from one machine to another.

Chapter 28. Cloning a Linux Server

28.1. About cloning

You can have distinct goals for cloning a server. For instance a clone can be a cold iron backup system used for manual disaster recovery of a service. Or a clone can be created to serve in a test environment. Or you might want to make an almost identical server. Let's take a look at some offline and online ways to create a clone of a Linux server.

28.2. About offline cloning

The term offline cloning is used when you power off the running Linux server to create the clone. This method is easy since we don't have to consider open files and we don't have to skip virtual file systems like **/dev** or **/sys** . The offline cloning method can be broken down into these steps:

```
1. Boot source and target server with a bootable CD
2. Partition, format and mount volumes on the target server
3. Copy files/partitions from source to target over the network
```

The first step is trivial. The second step is explained in the Disk Management chapter. For the third step, you can use a combination of **ssh** or **netcat** with **cp**, **dd**, **dump** and **restore**, **tar**, **cpio**, **rsync** or even **cat**.

28.3. Offline cloning example

We have a working Red Hat Enterprise Linux 5 server, and we want a perfect copy of it on newer hardware. First thing to do is discover the disk layout.

```
[root@RHEL5 ~]# df -h
Filesystem            Size  Used Avail Use% Mounted on
/dev/sda2              15G  4.5G  9.3G  33% /
/dev/sda1              99M   31M   64M  33% /boot
```

The **/boot** partition is small but big enough. If we create an identical partition, then **dd** should be a good cloning option. Suppose the / partition needs to be enlarged on the target system. The best option then is to use a combination of **dump** and **restore**. Remember that dd copies blocks, whereas dump/restore copies files.

The first step to do is to boot the target server with a live CD and partition the target disk. To do this we use the Red Hat Enterprise Linux 5 install CD. At the CD boot prompt we type "linux rescue". The cd boots into a root console where we can use fdisk to discover and prepare the attached disks.

When the partitions are created and have their filesystem, then we can use dd to copy the /boot partition.

```
ssh root@192.168.1.40 "dd if=/dev/sda1" | dd of=/dev/sda1
```

Then we use a dump and restore combo to copy the / partition.

```
mkdir /mnt/x
mount /dev/sda2 /mnt/x
cd /mnt/x
ssh root@192.168.1.40 "dump -0 -f - /" | restore -r -f -
```

Chapter 29. Introduction to Samba

29.1. Verify installed version

To see the version of samba installed on RedHat, use rpm -qa. Looks like **samba** 3 in the screenshot here, version 3.0.10.

```
[paul@RHEL4b ~]$  rpm -qa | grep samba
samba-common-3.0.10-1.4E.9
samba-client-3.0.10-1.4E.9
system-config-samba-1.2.21-1
samba-swat-3.0.10-1.4E.9
samba-3.0.10-1.4E.9
[paul@RHEL4b ~]$
```

Use dpkg -l on Debian or Ubuntu. Our Feisty Fawn here uses Samba 3.0.24

```
paul@laika:~$ dpkg -l | grep samba
ii  samba-common    3.0.24-2ubuntu1.2    Samba common files used by both the...
paul@laika:~$
```

29.2. Installing Samba

Samba is installed by default on Red Hat Enterprise Linux. If Samba is not yet installed, then the easiest way is to use the graphical menu (Applications -- System Settings -- Add/Remove Applications) and select "Windows File Server" in the Server section. The non-graphical way is to either use rpm -i followed by the samba-version.rpm file.

```
[paul@RHEL4b ~]$  rpm -i samba-3.0.10-1.4E.9.rpm
```

Or if you have a subscription to RHN, then **up2date** is the tool to use.

```
[paul@RHEL4b ~]$  up2date -i samba
```

Ubuntu and Debian users can use the aptitude program.

```
paul@laika:~$ aptitude install samba-server
```

29.3. Documentation

Obviously there are manual pages for Samba. Don't forget **man smb.conf**.

```
[root@RHEL4b samba]# apropos samba
cupsaddsmb       (8)  - export printers to samba for windows clients
lmhosts          (5)  - The Samba NetBIOS hosts file
net              (8)  - Tool for administration of Samba and remote CIFS servers
pdbedit          (8)  - manage the SAM database (Database of Samba Users)
samba            (7)  - A Windows SMB/CIFS fileserver for UNIX
smb.conf [smb]   (5)  - The configuration file for the Samba suite
smbpasswd        (5)  - The Samba encrypted password file
smbstatus        (1)  - report on current Samba connections
swat             (8)  - Samba Web Administration Tool
tdbbackup        (8)  - tool for backing up and ... of samba .tdb files
[root@RHEL4b samba]#
```

Samba comes with excellent documentation in html and pdf format (and also as a free download from Samba.org and are for sale as a printed book). Red Hat Enterprise Linux installs the html and pdf version in /usr/share/doc by default.

```
[paul@RHEL4b ~]$ locate Samba-HOWTO-Collection.pdf
/usr/share/doc/samba-3.0.10/Samba-HOWTO-Collection.pdf
```

Ubuntu packages the docs as a seperate package from Samba.

```
root@laika:~# aptitude search samba | grep -i documentation
i   samba-doc                       - Samba documentation
i   samba-doc-pdf                   - Samba documentation (PDF format)
root@laika:~# find /usr/share/doc/samba-doc-pdf | grep -i howto
/usr/share/doc/samba-doc-pdf/Samba3-HOWTO.pdf.gz
```

Besides the howto, there is also an excellent book called **Samba by example** (again available as book in shops, and as a free pdf and html).

29.4. smb.conf

Samba configuration is done in the **smb.conf** file. The file can be edited manually, or you can use a web based interface like webmin or swat to manage it. The file is usually located in /etc/samba. You can find the exact location with **smbd -b**.

```
[root@RHEL4b ~]# smbd -b | grep CONFIGFILE
CONFIGFILE: /etc/samba/smb.conf
[root@RHEL4b ~]#
```

The default smb.conf file contains a lot of examples with explanations.

```
[paul@RHEL4b ~]$ ls -l /etc/samba/smb.conf
-rw-r--r--  1 root root 10836 May 30 23:08 /etc/samba/smb.conf
(...)
paul@laika:~$ ls -l /etc/samba/smb.conf
-rw-r--r-- 1 root root 10515 2007-05-24 00:21 /etc/samba/smb.conf
```

Below is an example of a very minimalistic smb.conf. It allows samba to start, and to be visible to other computers (Microsoft shows computers in Network Neighborhood or My Network Places).

```
[paul@RHEL4b ~]$ cat /etc/samba/smb.conf
[global]
workgroup = WORKGROUP
[firstshare]
path = /srv/samba/public
[paul@RHEL4b ~]$
```

Below is a screenshot of the **net view** command on Microsoft Windows XP sp2. It shows how the Samba server with the minimalistic smb.conf is visible to Microsoft computers nearby.

```
C:\Documents and Settings\paul>net view
Server Name            Remark

-------------------------------------------------------------------------------
\\RHEL4B               Samba 3.0.10-1.4E.9
\\W2000
\\WINXP
The command completed successfully.
```

Some parameters in smb.conf can get a long list of values behind them. You can continue a line (for clarity) on the next by ending the line with a backslash.

```
valid users = Serena, Venus, Lindsay \
              Kim, Justine, Sabine \
              Amelie, Marie, Suzanne
```

Curious but true, smb.conf accepts synonyms like **create mode** and **create mask**, and sometimes minor spelling errors like **browsable** and **browseable**. And on occasion you can even switch words, the **guest only** parameter is identical to **only guest**.

29.5. testparm

To verify the syntax of the smb.conf file, you can use testparm.

```
[paul@RHEL4b ~]$ testparm
Load smb config files from /etc/samba/smb.conf
Processing section "[firstshare]"
Loaded services file OK.
Server role: ROLE_STANDALONE
Press enter to see a dump of your service definitions

[paul@RHEL4b ~]$
```

An interesting option is **testparm -v**, which will output all the global options with their default value. The remark seen by the **net view** command is the default value for the "server string" option. Simply adding this value to the global section in smb.conf and restarting samba will change the option. After a while, the changed option is visible on the Microsoft computers.

```
C:\Documents and Settings\paul>net view
Server Name            Remark
```

```
-------------------------------------------------------------------------------
\\RHEL4B               Public File Server
\\W2000
\\WINXP
The command completed successfully.
```

The samba daemons are constantly (once every 60 seconds) checking the smb.conf file, so it is good practice to keep this file small. But it is also good practice to document your samba configuration, and to explicitly set options that have the same default values. The **testparm -s** option allows you to do both. It will output the smallest possible samba configuration file, while retaining all your settings. The idea is to have your samba configuration in another file (like smb.conf.full) and let testparm parse this for you. The screenshot below shows you how. First the smb.conf.full file with the explicitly set option workgroup to WORKGROUP.

```
[root@RHEL4b samba]# cat smb.conf.full
[global]
workgroup = WORKGROUP

# This is a demo of a documented smb.conf
# These two lines are removed by testparm -s

server string = Public Test Server

[firstshare]
path = /srv/samba/public
```

Next, we execute testparm with the -s option, and redirect stdout to the real smb.conf file.

```
[root@RHEL4b samba]# testparm -s smb.conf.full > smb.conf
Load smb config files from smb.conf.full
Processing section "[firstshare]"
Loaded services file OK.
```

And below is the end result. The two comment lines and the default option are no longer there.

```
[root@RHEL4b samba]# cat smb.conf
# Global parameters
[global]
server string = Public Test Server

[firstshare]
path = /srv/samba/public
[root@RHEL4b samba]#
```

29.6. Samba daemons

Samba 3 consists of three daemons, they are named **nmbd**, **smbd** and **winbind**. The **nmbd** daemon takes care of all the names and naming. It registers and resolves names, and handles browsing. It should be the first daemon to start. The **smbd** daemon manages file transfers and authentication. It should be started after nmbd. The **winbind** daemon is only started to handle Microsoft Windows domain membership.

You can start the daemons by invoking **/etc/init.d/smb start** (some systems use **/etc/init.d/samba**) on any linux. Red Hat derived systems are happy with **service smb start**.

```
[root@RHEL4b ~]# /etc/init.d/smb start
Starting SMB services:                                     [  OK  ]
Starting NMB services:                                     [  OK  ]
[root@RHEL4b ~]# service smb restart
Shutting down SMB services:                                [  OK  ]
Shutting down NMB services:                                [  OK  ]
Starting SMB services:                                     [  OK  ]
Starting NMB services:                                     [  OK  ]
[root@RHEL4b ~]#
```

29.7. smbclient

With **smbclient** you can see browsing and share information from your smb server. It will display all your shares, your workgroup, and the name of the Master Browser. The -N switch is added to avoid having to enter an empty password. The -L switch is followed by the name of the host to check.

```
[root@RHEL4b init.d]# smbclient -NL rhel4b
Anonymous login successful
Domain=[WORKGROUP] OS=[Unix] Server=[Samba 3.0.10-1.4E.9]

Sharename       Type      Comment
---------       ----      -------
firstshare      Disk
IPC$            IPC       IPC Service (Public Test Server)
ADMIN$          IPC       IPC Service (Public Test Server)
Anonymous login successful
Domain=[WORKGROUP] OS=[Unix] Server=[Samba 3.0.10-1.4E.9]

Server               Comment
---------            -------
RHEL4B               Public Test Server
WINXP

Workgroup            Master
---------            -------
WORKGROUP            WINXP
```

The screenshot below uses smbclient to display information about a remote smb server (in this case a Windows XP machine).

```
[root@RHEL4b init.d]# smbclient -NL winxp
Anonymous login successful
Domain=[WORKGROUP] OS=[Windows 5.1] Server=[Windows 2000 LAN Manager]

Sharename       Type      Comment
---------       ----      -------
Error returning browse list: NT_STATUS_ACCESS_DENIED
Anonymous login successful
Domain=[WORKGROUP] OS=[Windows 5.1] Server=[Windows 2000 LAN Manager]
```

```
Server               Comment
---------            -------
RHEL4B               Public Test Server
W2000
WINXP

Workgroup            Master
---------            -------
WORKGROUP            WINXP
```

29.8. smbtree

Another useful tool to troubleshoot Samba or simply to browse the SMB network is **smbtree**. In its simplest form, smbtree will do an anonymous browsing on the local subnet. displaying all SMB computers and (if authorized) their shares.

Let's take a look at two screenshots of smbtree in action (with blank password). The first one is taken immediately after booting four different computers (one MS Windows 2000, one MS Windows XP, one MS Windows 2003 and one RHEL 4 with Samba 3.0.10).

```
[paul@RHEL4b ~]$ smbtree
Password:
WORKGROUP
PEGASUS
 \\WINXP
 \\RHEL4B                       Pegasus Domain Member Server
Error connecting to 127.0.0.1 (Connection refused)
cli_full_connection: failed to connect to RHEL4B<20> (127.0.0.1)
 \\HM2003
[paul@RHEL4b ~]$
```

The information displayed in the previous screenshot looks incomplete. The browsing elections are still ongoing, the browse list is not yet distributed to all clients by the (to be elected) browser master. The next screenshot was taken about one minute later. And it shows even less.

```
[paul@RHEL4b ~]$ smbtree
Password:
WORKGROUP
 \\W2000
[paul@RHEL4b ~]$
```

So we wait a while, and then run smbtree again, this time it looks a lot nicer.

```
[paul@RHEL4b ~]$ smbtree
Password:
WORKGROUP
 \\W2000
PEGASUS
 \\WINXP
 \\RHEL4B                       Pegasus Domain Member Server
  \\RHEL4B\ADMIN$                IPC Service (Pegasus Domain Member Server)
  \\RHEL4B\IPC$                  IPC Service (Pegasus Domain Member Server)
  \\RHEL4B\domaindata            Active Directory users only
 \\HM2003
[paul@RHEL4b ~]$ smbtree --version
Version 3.0.10-1.4E.9
[paul@RHEL4b ~]$
```

I added the version number of smbtree in the previous screenshot, to show you the difference when using the latest version of smbtree (below a screenshot taken from Ubuntu Feisty Fawn). The latest version shows a more complete overview of machines and shares.

```
paul@laika:~$ smbtree --version
Version 3.0.24
paul@laika:~$ smbtree
Password:
WORKGROUP
 \\W2000
  \\W2000\firstshare
  \\W2000\C$             Default share
  \\W2000\ADMIN$         Remote Admin
  \\W2000\IPC$           Remote IPC
PEGASUS
 \\WINXP
cli_rpc_pipe_open: cli_nt_create failed on pipe \srvsvc to machine WINXP.
Error was NT_STATUS_ACCESS_DENIED
 \\RHEL4B                        Pegasus Domain Member Server
  \\RHEL4B\ADMIN$                IPC Service (Pegasus Domain Member Server)
  \\RHEL4B\IPC$                  IPC Service (Pegasus Domain Member Server)
  \\RHEL4B\domaindata            Active Directory users only
 \\HM2003
cli_rpc_pipe_open: cli_nt_create failed on pipe \srvsvc to machine HM2003.
Error was NT_STATUS_ACCESS_DENIED
paul@laika:~$
```

The previous screenshot also provides useful errors on why we cannot see shared info on computers winxp and w2003. Let us try the old smbtree version on our RHEL server, but this time with Administrator credentials (which are the same on all computers).

```
[paul@RHEL4b ~]$ smbtree -UAdministrator%Stargate1
WORKGROUP
  \\W2000
PEGASUS
  \\WINXP
    \\WINXP\C$             Default share
    \\WINXP\ADMIN$         Remote Admin
    \\WINXP\share55
    \\WINXP\IPC$           Remote IPC
  \\RHEL4B                 Pegasus Domain Member Server
    \\RHEL4B\ADMIN$        IPC Service (Pegasus Domain Member Server)
    \\RHEL4B\IPC$          IPC Service (Pegasus Domain Member Server)
    \\RHEL4B\domaindata    Active Directory users only
  \\HM2003
    \\HM2003\NETLOGON      Logon server share
    \\HM2003\SYSVOL        Logon server share
    \\HM2003\WSUSTemp      A network share used by Local Publishing ...
    \\HM2003\ADMIN$        Remote Admin
    \\HM2003\tools
    \\HM2003\IPC$           Remote IPC
    \\HM2003\WsusContent    A network share to be used by Local ...
    \\HM2003\C$             Default share
[paul@RHEL4b ~]$
```

As you can see, this gives a very nice overview of all SMB computers and their shares.

29.9. Samba Web Administration Tool (SWAT)

Samba comes with a web based tool to manage your samba configuration file. The tool is accessible with a web browser on port 901 of the host system. To enable the tool, first find out whether your system is using the inetd or the xinetd superdaemon.

```
 [root@RHEL4b samba]# ps fax | grep inet
 15026 pts/0    S+     0:00                      \_ grep inet
  2771 ?        Ss     0:00 xinetd -stayalive -pidfile /var/run/xinetd.pid
 [root@RHEL4b samba]#
```

Then edit the inetd.conf or change the disable = yes line in /etc/xinetd.d/swat to disable = no.

```
[root@RHEL4b samba]# cat /etc/xinetd.d/swat
# default: off
# description: SWAT is the Samba Web Admin Tool. Use swat \
#              to configure your Samba server. To use SWAT, \
#              connect to port 901 with your favorite web browser.
service swat
{
 port            = 901
 socket_type     = stream
 wait            = no
 only_from       = 127.0.0.1
 user            = root
 server          = /usr/sbin/swat
 log_on_failure  += USERID
 disable         = no
}
[root@RHEL4b samba]# /etc/init.d/xinetd restart
Stopping xinetd:                                           [  OK  ]
Starting xinetd:                                           [  OK  ]
[root@RHEL4b samba]#
```

Be careful when using SWAT, it erases alle your manually edited comments in smb.conf.

29.10. Practice

0. !! Make sure you know your student number, anything *ANYTHING* you name must include your student number!

1. Verify that you can logon to a Linux/Unix computer. Write down the name and ip address of this machine.

2. Do the same for all the other (virtual) machines available to you.

3. Verify networking by pinging the machines, if you like names, edit the appropriate hosts files.

4. Make sure Samba is installed, write down the version of Samba.

5. Open the Official Samba-3 howto pdf file that is installed on your computer. How many A4 pages is this file ? Then look at the same pdf on samba.org, it is updated regularly.

6. Take a backup copy of the original smb.conf, name it smb.conf.orig

7. Enable SWAT and take a look at it.

8. Stop the Samba server.

9. Create a minimalistic smb.conf.minimal and test it with testparm.

10. Start Samba with your minimal smb.conf.

11. Verify with smbclient that your Samba server works.

12. Verify that another (Microsoft) computer can see your Samba server.

13. Browse the network with net view and smbtree.

14. Change the "Server String" parameter in smb.conf. How long does it take before you see the change (net view, smbclient, My Network Places,...) ?

15. Will restarting Samba after a change to smb.conf speed up the change ?

16. Which computer is the master browser master in your workgroup ? What is the master browser ?

17. If time permits (or if you are waiting for other students to finish this practice), then install a sniffer (ethereal/wireshark) and watch the browser elections.

Chapter 30. Simple Samba File Servers

30.1. Read Only File Server

Let's start with setting up a very simple read only file server with Samba. Everyone (even anonymous guests) will receive read access.

The first step is to create a directory and put some test files in it.

```
[root@RHEL4b samba]# mkdir -p /srv/samba/readonly
[root@RHEL4b samba]# ls -l /srv/samba/
total 4
drwxr-xr-x  2 root root 4096 Jun 22 11:07 readonly
[root@RHEL4b samba]# cd /srv/samba/readonly/
[root@RHEL4b readonly]# echo "It is cold today." > winter.txt
[root@RHEL4b readonly]# echo "It is hot today." > summer.txt
[root@RHEL4b readonly]# ll
total 8
-rw-r--r--  1 root root 17 Jun 22 11:13 summer.txt
-rw-r--r--  1 root root 18 Jun 22 11:13 winter.txt
[root@RHEL4b readonly]#
```

Linux will always require a user account before giving access to files (the files in our example above are owned by root). So we will create a user for our readonly file server and make this user the owner of the directory and all of its files. (Strictly speaking, you can setup a Samba read only file server without creating an extra user account).

```
[root@RHEL4b ~]# useradd -c "Anonymous Samba Access" -p secret -s /bin/false Samba_nobod
[root@RHEL4b samba]# chown Samba_nobody.Samba_nobody /srv/samba/readonly/
[root@RHEL4b samba]# chmod 777 /srv/samba/readonly/
[root@RHEL4b samba]# ls -l /srv/samba/
total 4
drwxrwxrwx  2 Samba_nobody Samba_nobody 4096 Jun 22 11:09 readonly
[root@RHEL4b samba]# cd /srv/samba/readonly/
[root@RHEL4b readonly]# chown Samba_nobody.Samba_nobody *
[root@RHEL4b readonly]# ll
total 8
-rw-r--r--  1 Samba_nobody Samba_nobody 17 Jun 22 11:13 summer.txt
-rw-r--r--  1 Samba_nobody Samba_nobody 18 Jun 22 11:13 winter.txt
[root@RHEL4b samba]#
```

It is time to create the smb.conf file (feel free to test it with testparm). We put our file server in the default workgroup, give it a descriptive server string, and set the security to share level (more on this later). The share is called pubread, and access to the share is enforced by Samba (remember we gave 777 to the directory).

```
[root@RHEL4b samba]# cat smb.conf
[global]
workgroup = WORKGROUP
server string = Public Anonymous File Server
security = share
```

```
[pubread]
path = /srv/samba/readonly
comment = files to read
read only = Yes
guest ok = Yes
[root@RHEL4b samba]#
```

After testing with testparm, restart the samba server and verify the existence of the share with smbclient.

```
[root@RHEL4b readonly]# service smb restart
Shutting down SMB services:                                [  OK  ]
Shutting down NMB services:                                [  OK  ]
Starting SMB services:                                     [  OK  ]
Starting NMB services:                                     [  OK  ]
[root@RHEL4b readonly]# smbclient -L 127.0.0.1
Password:
Domain=[WORKGROUP] OS=[Unix] Server=[Samba 3.0.10-1.4E.9]

Sharename       Type      Comment
---------       ----      -------
pubread         Disk      files to read
IPC$            IPC       IPC Service (Public Anonymous File Server)
ADMIN$          IPC       IPC Service (Public Anonymous File Server)
Domain=[WORKGROUP] OS=[Unix] Server=[Samba 3.0.10-1.4E.9]

Server               Comment
---------            -------
RHEL4B               Public Test Server
WINXP

Workgroup            Master
---------            -------
WORKGROUP            WINXP
[root@RHEL4b readonly]#
```

The final test is to go to a Microsoft Windows computer and read a file on the Samba server. First we use the **net use** command to mount the pubread share on the driveletter k.

```
C:\Documents and Settings\paul>net use k: \\rhel4b\pubread
The command completed successfully.
```

Then we test looking at the contents of the share, and reading the files.

```
C:\Documents and Settings\paul>k:

K:\>dir
Volume in drive K is pubread
Volume Serial Number is 0D56-11F2

Directory of K:\

06/22/2007  11:13 AM    <DIR>          .
06/22/2007  11:09 AM    <DIR>          ..
06/22/2007  11:13 AM                18 winter.txt
```

```
06/22/2007  11:13 AM                17 summer.txt
2 File(s)             35 bytes
2 Dir(s)   2,763,522,048 bytes free

K:\>type winter.txt
It is cold today.

K:\>
```

Just to be on the safe side, let us try writing.

```
K:\>echo very cold > winter.txt
Access is denied.

K:\>
```

30.2. Practice

1. Create a directory in a good location (FHS) to share files for everyone to read.

2. Make sure the directory is owned properly, put a textfile in it, then share it with Samba.

3. Verify from your own and from another computer (smbclient, net use, ...) that the share is accessible for reading.

4. Make a backup copy of your smb.conf, name it smb.conf.ReadOnlyFileServer.

30.3. Writable File Server

In this second example, we will create a share where everyone can create files and write to files. Similar to before, we start by creating a directory, and setting ownership to our Samba_nobody user account.

```
[root@RHEL4b samba]# mkdir /srv/samba/writable
[root@RHEL4b samba]# chown Samba_nobody.Samba_nobody /srv/samba/writable/
[root@RHEL4b samba]# chmod 777 /srv/samba/writable/
```

Then we simply add a share to our file server by editing smb.conf. Below the check with testparm.

```
[root@RHEL4b samba]# testparm
Load smb config files from /etc/samba/smb.conf
Processing section "[pubread]"
Processing section "[pubwrite]"
Loaded services file OK.
Server role: ROLE_STANDALONE
Press enter to see a dump of your service definitions

# Global parameters
[global]
```

```
server string = Public Anonymous File Server
security = SHARE

[pubread]
comment = files to read
path = /srv/samba/readonly
guest ok = Yes

[pubwrite]
comment = files to read and write
path = /srv/samba/writable
read only = No
guest ok = Yes
```

Restart Samba, then onto the Windows XP machine and test our writing skills.

```
C:\Documents and Settings\paul>net use w: \\rhel4b\pubwrite
The command completed successfully.

C:\Documents and Settings\paul>w:

W:\>echo This is a write test > hello.txt

W:\>dir
Volume in drive W is pubwrite
Volume Serial Number is 0D56-272A

Directory of W:\

06/22/2007  12:29 PM    <DIR>          .
06/22/2007  12:26 PM    <DIR>          ..
06/22/2007  12:31 PM                23 hello.txt
1 File(s)             23 bytes
2 Dir(s)   2,763,522,048 bytes free

W:\>type hello.txt
type hello.txt
This is a write test

W:\>
```

There is one little issue though; the linux owner of the files created through this writable share is the linux guest account (usually named nobody).

```
[root@RHEL4b samba]# ls -l /srv/samba/writable/
total 4
-rwxr--r--  1 nobody nobody 23 Jun 22 12:31 hello.txt
-rwxr--r--  1 nobody nobody  0 Jun 22 12:33 test.txt
[root@RHEL4b samba]#
```

So this is not the cleanest solution. We will improve this in the next topic.

30.4. Forcing a User Owner

The Samba_nobody user account that we created in the previous examples is actually not used by Samba. It just owns the files and directories that we created for our shares.

The goal of this section is to force ownership of files created through the Samba share to belong to our Samba_nobody user. Remember, our server is still accessible to everyone, nobody needs to know this user account or password. We just want a clean linux server.

To accomplish this, we first have to tell Samba about this user. We can do this by adding the account to **smbpasswd**.

```
[root@RHEL4b samba]# smbpasswd -a Samba_nobody
New SMB password:
Retype new SMB password:
Added user Samba_nobody.
[root@RHEL4b samba]#
```

To find out where Samba keeps this information, use **smbd -b**. The PRIVATE_DIR variable will show you where the smbpasswd database is located.

```
[root@RHEL4b samba]# smbd -b | grep -i private
PRIVATE_DIR: /etc/samba
```

You can use a simple cat to see the contents of the smbpasswd database. The nobody user does not have a password, the Samba_nobody user does have one (it is secret).

```
[root@RHEL4b samba]# cat /etc/samba/smbpasswd
nobody:99:XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX:XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX...
Samba_nobody:502:552902031BEDE9EFAAD3B435B51404EE:878D8014606CDA29677A44...
[root@RHEL4b samba]#
```

Now that Samba knows about this user, we can adjust our writable share to force the ownership of files created through it. For this we use the **force user** and **force group** options. Now we can be sure that all files in the Samba writable share are owned by the same Samba_nobody user.

```
[root@RHEL4b samba]# testparm -s smb.conf 2>/dev/null | tail -7
[pubwrite]
comment = files to read and write
path = /srv/samba/writable
force user = Samba_nobody
force group = Samba_nobody
read only = No
guest ok = Yes
[root@RHEL4b samba]#
```

30.5. More about smbclient

Instead of going to the Microsoft machines, we can do the same tests from within linux with **smbclient**. This first screenshot shows how to verify that Samba is running on your localhost, how to list all the Samba shares, who is the Master Browser of the workgroup and some more information.

```
[paul@RHEL4b ~]$ smbclient -NL localhost
Domain=[WORKGROUP] OS=[Unix] Server=[Samba 3.0.10-1.4E.9]

Sharename       Type      Comment
---------       ----      -------
pubread         Disk      files to read
pubwrite        Disk      files to read and write
authwrite       Disk      authenticated users only
IPC$            IPC       IPC Service (Public Anonymous File Server)
ADMIN$          IPC       IPC Service (Public Anonymous File Server)
Domain=[WORKGROUP] OS=[Unix] Server=[Samba 3.0.10-1.4E.9]

Server               Comment
---------            -------
RHEL4B               Public Anonymous File Server
WINXP

Workgroup            Master
---------            -------
WORKGROUP            WINXP
[paul@RHEL4b ~]$
```

It can also be used to test user access to a Samba share. First an example of how to test anonymous access to our pubread share. If the connection is established, then we get an smb prompt. You can use exit or q to return to bash.

```
[paul@RHEL4b ~]$ smbclient //rhel4b/pubread -U%
Domain=[WORKGROUP] OS=[Unix] Server=[Samba 3.0.10-1.4E.9]
smb: \> dir
.                                    D        0  Fri Jun 22 11:13:15 2007
..                                   D        0  Fri Jun 22 13:03:54 2007
winter.txt                                   18  Fri Jun 22 11:13:11 2007
summer.txt                                   17  Fri Jun 22 11:13:15 2007

45734 blocks of size 262144. 10541 blocks available
smb: \> exit
[paul@RHEL4b ~]$
```

30.6. NetBIOS name resolving

If your clients are spread across multiple subnets, then it is likely there is a WINS (Microsoft Windows Internet Naming Service) or NBNS (NetBIOS Name Server) available to resolve NetBIOS names. You should then point Samba to the wins server with the **wins server** parameter.

```
wins server = 10.0.0.42
```

You can set the resolving order that Samba should use with the **name resolve order** parameter.

```
name resolve order = wins lmhosts host bcast
```

30.7. Practice

1. Create a directory and share it with Samba.

2. Make sure everyone can read and write files, test writing with smbclient and from a Microsoft computer.

3. Verify the ownership of files created by various users.

4. Use the "force user" and "force group" directives to force ownership of files created in this shared directory.

5. Test that Samba properly registers in a WINS server.

6. Test the working of force user with smbclient and/or net use and/or the MS Windows Explorer.

Chapter 31. Samba Servers with authentication and restrictions

31.1. Authenticated User Access

The goal of this example is to set up a file share accessible to a number of different users. The users will need to authenticate with their password before access to this share is granted. We will first create three randomly named users, each with their own password. First we add these users to linux.

```
[root@RHEL4b samba]# useradd -c "Serena Williams" -p SerenaW Serena
[root@RHEL4b samba]# useradd -c "Kim Clijsters" -p KimC Kim
[root@RHEL4b samba]# useradd -c "Martina Hingis" -p MartinaH Martina
```

Then we add them to the smbpasswd file, with the same password.

```
[root@RHEL4b samba]# smbpasswd -a Serena
New SMB password:
Retype new SMB password:
Added user Serena.
[root@RHEL4b samba]# smbpasswd -a Kim
New SMB password:
Retype new SMB password:
Added user Kim.
[root@RHEL4b samba]# smbpasswd -a Martina
New SMB password:
Retype new SMB password:
Added user Martina.
```

We add the following section to our smb.conf (and create the directory /srv/samba/authwrite).

```
[authwrite]
path = /srv/samba/authwrite
comment = authenticated users only
read only = No
guest ok = No
```

After restarting Samba, we test with different users from within Microsoft computers. First Kim from Windows XP.

```
C:\>net use m: \\rhel4b\authwrite /user:Kim KimC
The command completed successfully.

C:\>m:

M:\>echo greetings from Kim > greetings.txt
```

The next screenshot is Martina on a Windows 2000 computer, she succeeds in writing her files, but fails to overwrite the file from Kim.

```
C:\>net use k: \\rhel4b\authwrite /user:Martina MartinaH
The command completed successfully.

C:\>k:

K:\>echo greetings from martina > Martina.txt

K:\>echo test overwrite > greetings.txt
Access is denied.
```

You can also test connecting with authentication with smbclient, first we a wrong password, then with the correct one.

```
[paul@RHEL4b ~]$ smbclient //rhel4b/authwrite -UMartina%wrongpass
Domain=[WORKGROUP] OS=[Unix] Server=[Samba 3.0.10-1.4E.9]
tree connect failed: NT_STATUS_WRONG_PASSWORD
[paul@RHEL4b ~]$ smbclient //rhel4b/authwrite -UMartina%MartinaH
Domain=[WORKGROUP] OS=[Unix] Server=[Samba 3.0.10-1.4E.9]
smb: \> more Martina.txt
getting file \Martina.txt of size 25 as /tmp/smbmore.Uv6c86 (24.4 kb/s) (average 24.4 kb
greetings from martina
smb: \> q
[paul@RHEL4b ~]$
```

Congratulations, you now have a simple standalone Samba file server with authenticated access. And the files in the shares belong to their proper owners.

```
[root@RHEL4b samba]# ls -l /srv/samba/authwrite/
total 8
-rwxr--r--  1 Kim     Kim     17 Jun 22 13:05 greetings.txt
-rwxr--r--  1 Martina Martina 25 Jun 22 13:08 Martina.txt
```

31.2. Frequently used share settings

31.2.1. valid users

To restrict users per share, you can use the **valid users** parameter. In the example below, only the users listed as valid will be able to access the tennis share.

```
[tennis]
 path = /srv/samba/tennis
 comment = authenticated and valid users only
 read only = No
 guest ok = No
 valid users = serena, kim, venus, justine
```

31.2.2. invalid users

If you are paranoia, you can also use **invalid users** to explicitely deny the listed users access. When a user is in both lists, the user has no access!

```
[tennis]
 path = /srv/samba/tennis
 read only = No
 guest ok = No
 valid users = kim, serena, venus, justine
 invalid users = venus
```

31.2.3. create mask and inherit permissions

Similar to umask (but not inverted), you can use the **create mask** and **directory mask** to set default permissions for newly created files and directories.

```
[tennis]
 path = /srv/samba/tennis
 read only = No
 guest ok = No
 create mask = 644
```

With **inherit permissions = Yes** you can force newly created files and directories to inherit permissions from their parent directory, overriding the create mask and directory mask settings.

31.2.4. hosts allow

The **hosts allow** or **allow hosts** parameter is one of the key advantages of Samba. It allows access control of shares on the ip-address level. To allow only specific hosts to access a share, list the hosts, seperated by comma's.

```
allow hosts = 192.168.1.5, 192.168.1.40
```

Allowing entire subnets is done by ending the range with a dot.

```
allow hosts = 192.168.1.
```

Subnet masks can be added in the classical way.

```
allow hosts = 10.0.0.0/255.0.0.0
```

You can also allow an entire subnet with exceptions.

```
hosts allow = 10. except 10.0.0.12
```

31.2.5. hosts deny

The **hosts deny** or **deny hosts** parameter is the logical counterpart of the previous. The syntax is the same as for hosts allow.

```
hosts deny = 192.168.1.55, 192.168.1.56
```

31.2.6. hide unreadable

Setting **hide unreadable** to yes will prevent users from seeing files that cannot be read by them.

```
hide unreadable = yes
```

31.2.7. read list

One more setting before we go on to the next topic. Even on a writable share, you can set a list of read only users with the **read list** parameter.

```
[authwrite2]
 path = /srv/samba/authwrite2
 comment = authenticated users only
 read only = No
 guest ok = No
 read list = Martina, Roberto
```

31.3. Practice

0. Make sure you have properly named backups of your smb.conf of the previous practices.

1. Create three users (on the Unix and on the Samba), remember their passwords!

2. Set up a shared directory that is only accessible to authenticated users.

3. Verify that files created by these users belong to them.

4. Limit access to the sales share to Sandra, Ann and Veronique. Make sure that Roberto cannot access the share.

5. Even though the share is writable, Ann should only have read access.

6. Set the create mask for files to read and write for everyone, test that it works.

7. Limit one shared directory to the 192.168.1.0/24 subnet, and another share to the two computers with ip-addresses 192.168.1.33 and 172.17.18.19.

8. Make sure users can only see files and directories that they can read. Test that it works!!

9. If time permits (or if you are waiting for other students to finish this practice), then combine the "read only" and "writable" statements to check which one has priority. Then combine them with "read list", "write list", "hosts allow" and "hosts deny". Then

combine them with file permissions on the linux filesystem (chmod,chown) and make a table of minimal mandatory settings for readonly/readwrite shared directories.

Chapter 32. Samba Domain Member Server

32.1. smb.conf

The **workgroup** option in the global section should match the netbios name of the Active Directory domain. Authentication will not be handled by Samba now, but by the Active Directory Domain Controllers, so we set the **security** option to domain. Since linux requires a user account for every user accessing its file system, we need to provide Samba with a range of uid's and gid's that it can use to create these user accounts. The first Active Directory user to connect will receive linux uid 20000. Below is our new global section in smb.conf.

```
[global]
 workgroup = PEGASUS
 server string = Pegasus Domain Member Server
 security = Domain
 idmap uid = 20000-22000
 idmap gid = 20000-22000
 winbindd use default domain = Yes
```

Nothing special is required for the share section in smb.conf. Remember, we do not manually create users in smbpasswd or on the linux (/etc/passwd). Only Active Directory users are allowed access.

```
[domaindata]
 path = /srv/samba/domaindata
 comment = Active Directory users only
 read only = No
```

32.2. Joining the Active Directory Domain

While the Samba server is stopped, you can use **net rpc join** to join the Active Directory Domain.

```
[root@RHEL4b samba]# net rpc join -UAdministrator%Stargate1
Joined domain PEGASUS.
[root@RHEL4b samba]#
```

Time to start Samba followed by **winbind**.

```
[root@RHEL4b samba]# service smb start
Starting SMB services:                                     [  OK  ]
Starting NMB services:                                     [  OK  ]
[root@RHEL4b samba]# service winbindd start
Starting winbindd services:                                [  OK  ]
[root@RHEL4b samba]#
```

32.3. nsswitch.conf

We need to update the **/etc/nsswitch.conf** file now, so user group and host names can be resolved against the winbindd daemon.

```
[root@RHEL4b samba]# vi /etc/nsswitch.conf
[root@RHEL4b samba]# grep winbindd /etc/nsswitch.conf
passwd:     files winbindd
group:      files winbindd
hosts:      files dns winbindd
[root@RHEL4b samba]#
```

32.4. winbind

The **winbind** daemon is talking with the Active Directory domain. With **wbinfo** you can provide winbindd with credentials to talk to Active Directory.

```
[root@RHEL4b samba]# wbinfo --set-auth-user=Administrator%Stargate1
```

We can also use **wbinfo -a** to verify authentication of a user against Active Directory. Assuming a user account Venus with password VenusW is just created on the Active Directory, we get the following screenshot.

```
[root@RHEL4b samba]# wbinfo -a Venus%VenusW
plaintext password authentication succeeded
challenge/response password authentication succeeded
[root@RHEL4b samba]#
```

We can use **getent** to verify that winbindd is working and actually adding the Active directory users to /etc/passwd. The screenshot below shows that Kim and Serena are normal linux users in /etc/passwd, and that the Active Directory user Venus received uid 20000 in /etc/passwd.

```
[root@RHEL4b samba]# getent passwd Kim
Kim:x:504:504:Kim Clijsters:/home/Kim:/bin/bash
[root@RHEL4b samba]# getent passwd Serena
Serena:x:503:503:Serena Williams:/home/Serena:/bin/bash
[root@RHEL4b samba]# getent passwd Venus
venus:*:20000:20000::/home/PEGASUS/venus:/bin/false
```

Not all Active Directory user accounts added to /etc/passwd by winbindd, only those that have been used.

```
[root@RHEL4b samba]# getent passwd Justine
[root@RHEL4b samba]# wbinfo -a Justine%JustineH
plaintext password authentication succeeded
challenge/response password authentication succeeded
[root@RHEL4b samba]# getent passwd Justine
justine:*:20001:20000::/home/PEGASUS/justine:/bin/false
[root@RHEL4b samba]#
```

All the Active Directory users can now easily connect to the Samba share. Files created by them, belong to them.

```
[root@RHEL4b samba]# ll /srv/samba/domaindata/
total 0
-rwxr--r--  1 justine 20000 0 Jun 22 19:54 created_by_justine_on_winxp.txt
-rwxr--r--  1 venus   20000 0 Jun 22 19:55 created_by_venus.txt
-rwxr--r--  1 maria   20000 0 Jun 22 19:57 Maria.txt
```

32.5. Practice

1. Verify that you have a working Active Directory (AD) domain.

2. Setup Samba as a member server in the domain.

3. Verify the creation of a computer account in AD for your Samba server.

4. Verify the automatic creation of AD users in /etc/passwd with wbinfo and getent.

5. Connect to Samba shares with AD users, and verify ownership of their files.

Chapter 33. Samba Domain Controller

33.1. About Domain Controllers

33.1.1. Samba 3

Samba 3 can act as a domain controller in its own domain. In a Windows NT4 domain, with one Windows NT4 PDC and zero or more BDC's, Samba 3 can only be a member server. The same is valid for Samba 3 in an Active Directory Domain with Windows 2000 and/or Windows 2003 DC's. In short, a Samba 3 domain controller can not share domain control with Windows domain controllers.

33.1.2. Samba 4

Samba 4 can be a domain Controller in an Active Directory domain, but as of this writing, Samba 4 is not released for production!

33.1.3. About password backends

The example below uses the **tdbsam** password backend. Another option would be to use LDAP. Larger domains will benefit from using LDAP instead of the not so scalable tdbsam. When you need more than one Domain Controller, then the Samba team advises to not use tdbsam.

33.2. smb.conf

Now is a good time to start adding comments in your smb.conf. First we'll take a look at the naming of our domain and server in the **[global]** section, and at the domain controlling parameters. The security must be set to user (which is the default). Our Samba server is the most stable system in the network, so it should win all browser elections (**os level** above 32) to become the **browser master**, and it should accept domain logons (**domain logons = Yes**).

```
[global]
# names
 workgroup = SPORTS
 netbios name = DCSPORTS
 server string = Sports Domain Controller
# domain control parameters
 security = user
 os level = 80
 preferred master = Yes
 domain master = Yes
 domain logons = Yes
```

Then we create some sections for file shares, to test our Samba server. Users can all access the general sports file share, but only group members can access their own sport share.

```
[sports]
comment = Information about all sports
path = /srv/samba/sports
valid users = @ntsports
read only = No

[tennis]
comment = Information about tennis
path = /srv/samba/tennis
valid users = @nttennis
read only = No

[football]
comment = Information about football
path = /srv/samba/football
valid users = @ntfootball
read only = No
```

Part of the Microsoft definition of a domain controller is that it should have a **netlogon share**. This is the relevant part of smb.conf to create this netlogon share on Samba.

```
[netlogon]
comment = Network Logon Service
path = /srv/samba/netlogon
admin users = root
guest ok = Yes
browseable = No
```

33.3. Users and Groups

To be able to use users and groups in Samba, we have to set up some users and groups on the Linux computer.

```
[root@RHEL4b samba]# groupadd ntadmins
[root@RHEL4b samba]# groupadd ntsports
[root@RHEL4b samba]# groupadd nttennis
[root@RHEL4b samba]# groupadd ntfootball
[root@RHEL4b samba]# useradd -m -G ntadmins -p Stargate1 Administrator
[root@RHEL4b samba]# useradd -m -G ntsports,nttennis -p stargate Venus
[root@RHEL4b samba]# useradd -m -G ntsports,nttennis -p stargate Serena
[root@RHEL4b samba]# useradd -m -G ntsports,nttennis -p stargate Kim
[root@RHEL4b samba]# useradd -m -G ntsports,ntfootball -p stargate Figo
[root@RHEL4b samba]# useradd -m -G ntsports,ntfootball -p stargate Pfaff
```

Next we must make these users known to Samba with the smbpasswd tool. When you add the first user to **tdbsam**, the file **/etc/samba/passdb.tdb** will be created.

```
[root@RHEL4b samba]# smbpasswd -a Administrator
New SMB password:
```

```
Retype new SMB password:
Unable to open/create TDB passwd
pdb_getsampwnam: Unable to open TDB passwd (/etc/samba/passdb.tdb)!
TDBSAM version too old (0), trying to convert it.
TDBSAM converted successfully.
Added user Administrator.
[root@RHEL4b samba]#
```

Adding the second user generates less output.

```
[root@RHEL4b samba]# smbpasswd -a root
New SMB password:
Retype new SMB password:
Added user root.
```

33.4. About Computer Accounts

Every NT computer (Windows NT, 2000, XP, Vista) can become a member of a domain. Joining the domain (by right-clicking on My Computer) means that a computer account will be created in the domain. This computer account also has a password (but you cannot know it) to prevent other computers with the same name from accidentally becoming member of the domain. The computer account created by Samba is visible in the **/etc/passwd** file on linux. Computer accounts appear as a normal user account, but end their name with a dollar sign. Below a screenshot of the winxp$ computer account, created by Samba 3.

```
[root@RHEL4b samba]# tail -5 /etc/passwd
Serena:x:508:512::/home/Serena:/bin/bash
Kim:x:509:513::/home/Kim:/bin/bash
Figo:x:510:514::/home/Figo:/bin/bash
Pfaff:x:511:515::/home/Pfaff:/bin/bash
winxp$:x:512:516::/home/nobody:/bin/false
```

To be able to create the account, you will need to provide credentials of an account with the permission to create accounts (by default only root can do this on Linux). And we will have to tell Samba how to to this, by adding an **add machine script** to the global section of smb.conf.

```
add machine script = /usr/sbin/useradd -s /bin/false -d /home/nobody %u
```

You can now join a Microsoft computer to the sports domain (with the root user). After reboot of the Microsoft computer, you will be able to logon with Administrator (password Stargate1), but you will get an error about your roaming profile. We will fix this in the next section.

33.5. Roaming Profiles

For your information, if you want to force local profiles instead of roaming profiles, then simply add the following two lines to the global section in smb.conf.

```
logon home =
logon path =
```

Microsoft computers store a lot of User Metadata and application data in a user profile. Making this profile available on the network will enable users to keep their Desktop and Application settings across computers. User profiles on the network are called **roaming profiles** or **roving profiles**. The Samba domain controller can manage these profiles. First we need to add the relevant section in smb.conf.

```
[Profiles]
 comment = User Profiles
 path = /srv/samba/profiles
 readonly = No
 profile acls = Yes
```

Besides the share section, we also need to set the location of the profiles share (this can be another Samba server) in the global section.

```
 logon path = \\%L\Profiles\%U
```

The **%L** variable is the name of this Samba server, the **%U** variable translates to the username. After adding a user to smbpasswd and letting the user log on and off, the profile of the user will look like this.

```
[root@RHEL4b samba]# ll /srv/samba/profiles/Venus/
total 568
drwxr-xr-x  4 Venus Venus   4096 Jul  5 10:03 Application Data
drwxr-xr-x  2 Venus Venus   4096 Jul  5 10:03 Cookies
drwxr-xr-x  3 Venus Venus   4096 Jul  5 10:03 Desktop
drwxr-xr-x  3 Venus Venus   4096 Jul  5 10:03 Favorites
drwxr-xr-x  4 Venus Venus   4096 Jul  5 10:03 My Documents
drwxr-xr-x  2 Venus Venus   4096 Jul  5 10:03 NetHood
-rwxr--r--  1 Venus Venus 524288 Jul  5  2007 NTUSER.DAT
-rwxr--r--  1 Venus Venus   1024 Jul  5  2007 NTUSER.DAT.LOG
-rw-r--r--  1 Venus Venus    268 Jul  5 10:03 ntuser.ini
drwxr-xr-x  2 Venus Venus   4096 Jul  5 10:03 PrintHood
drwxr-xr-x  2 Venus Venus   4096 Jul  5 10:03 Recent
drwxr-xr-x  2 Venus Venus   4096 Jul  5 10:03 SendTo
drwxr-xr-x  3 Venus Venus   4096 Jul  5 10:03 Start Menu
drwxr-xr-x  2 Venus Venus   4096 Jul  5 10:03 Templates
[root@RHEL4b samba]#
```

33.6. Groups in NTFS acls

We have users on Unix, we have groups on Unix that contain those users.

```
[root@RHEL4b samba]# grep nt /etc/group
...
ntadmins:x:506:Administrator
ntsports:x:507:Venus,Serena,Kim,Figo,Pfaff
nttennis:x:508:Venus,Serena,Kim
ntfootball:x:509:Figo,Pfaff
```

```
[root@RHEL4b samba]#
```

We already added Venus to the **tdbsam** with **smbpasswd**.

```
smbpasswd -a Venus
```

Does this mean that Venus can access the tennis and the sports shares ? Yes, all access works fine on the Samba server. But the nttennis group is not available on the windows machines. To make the groups available on windows (like in the ntfs security tab of files and folders), we have to map unix groups to windows groups. To do this, we use the **net groupmap** command.

```
[root@RHEL4b samba]# net groupmap add ntgroup="tennis" unixgroup=nttennis type=d
No rid or sid specified, choosing algorithmic mapping
Successully added group tennis to the mapping db
[root@RHEL4b samba]# net groupmap add ntgroup="football" unixgroup=ntfootball type=d
No rid or sid specified, choosing algorithmic mapping
Successully added group football to the mapping db
[root@RHEL4b samba]# net groupmap add ntgroup="sports" unixgroup=ntsports type=d
No rid or sid specified, choosing algorithmic mapping
Successully added group sports to the mapping db
[root@RHEL4b samba]#
```

Now you can use the Samba groups on all NTFS volumes on members of the domain.

33.7. logon scripts

Before testing a logon script, make sure it has the proper carriage returns that DOS files have.

```
[root@RHEL4b netlogon]# cat start.bat
net use Z: \\DCSPORTS0\SPORTS
[root@RHEL4b netlogon]# unix2dos start.bat
unix2dos: converting file start.bat to DOS format ...
[root@RHEL4b netlogon]#
```

Then copy the scripts to the netlogon share, and add the following parameter to smb.conf.

```
logon script = start.bat
```

33.8. Practice

1. Setup Samba as a domain controller.

2. Create the shares salesdata, salespresentations and meetings. Salesdata must be accessible to all sales people and to all managers. SalesPresentations is only for all sales people. Meetings is only accessible to all managers. Use groups to accomplish this.

3. Join a Microsoft computer to your domain. Verify the creation of a computer account in /etc/passwd.

4. Setup and verify the proper working of roaming profiles.

5. Find information about home directories for users, set them up and verify that users receive their home directory mapped under the H:-drive in MS Windows Explorer.

6. Use a couple of samba domain groups with members to set acls on ntfs. Verify that it works!

7. Knowing that the %m variable contains the computername, create a seperate log file for every computer(account).

8. Knowing that %s contains the client operating system, include a smb.%s.conf file that contains a share. (The share will only be visible to clients with that OS).

9. If time permits (or if you are waiting for other students to finish this practice), then combine "valid users" and "invalid users" with groups and usernames with "hosts allow" and "hosts deny" and make a table of which get priority over which.

Chapter 34. Samba Print Servers

34.1. Simple CUPS Print Server

Let us start by setting up a Samba print server that serves two printers which are set up with the CUPS web interface (http://localhost:631). We make these printers available to everyone for printing. We set up the CUPS printers without a driver (raw printing device). The **lpstat** tool will see the printers like this.

```
[root@RHEL4b samba]# lpstat -t
scheduler is running
system default destination: HPColor
device for HPBlack: socket://192.168.1.244:9100
device for HPColor: parallel:/dev/lp0
HPBlack accepting requests since Jan 01 00:00
HPColor accepting requests since Jan 01 00:00
printer HPBlack is idle.  enabled since Jan 01 00:00
printer HPColor is idle.  enabled since Jan 01 00:00
```

The windows clients need to install the correct printer driver themselves, so the spooler just sends the jobs to the print device (without any kind of processing or interpreting of the print jobs). Our smb.conf looks like this.

```
[global]
 server string = Public Anonymous Print Server
 security = share
 disable spoolss = No
 printing = cups

[printers]
 path = /var/spool/samba
 read only = Yes
 printable = Yes
 use client driver = Yes
```

Let's do a quick check with smbclient.

```
[root@RHEL4b samba]# smbclient -NL 127.0.0.1
Domain=[WORKGROUP] OS=[Unix] Server=[Samba 3.0.10-1.4E.9]

Sharename       Type      Comment
---------       ----      -------
IPC$            IPC       IPC Service (Public Anonymous Print Server)
ADMIN$          IPC       IPC Service (Public Anonymous Print Server)
HPBlack         Printer   Local Raw Printer
HPColor         Printer   Local Raw Printer
...
```

That looks ok. Now you can add the printer to windows computers in the workgroup, just browse to your Samba server in the add printer wizard. Or you can connect with the **net use** command as shown below.

```
C:\shov>net use lpt1: \\rhel4b\HPColor
The command completed successfully.

C:\shov>net use
New connections will be remembered.


Status       Local     Remote                    Network

-------------------------------------------------------------------------------
OK           LPT1      \\rhel4b\HPColor          Microsoft Windows Network
The command completed successfully.

C:\shov>print shovel.bat
C:\shov\shovel.bat is currently being printed
```

After printing a test page (by rightclicking on the printer icon in windows and then clicking on the print test page button of the properties dialog box) and issuing the print command from within Firefox, the print queue looks like this.

```
[root@RHEL4b samba]# lpq -a
Rank    Owner   Job     File(s)                         Total Size
active  nobody  4       smbprn.00000001 Test Page       112640 bytes
1st     nobody  5       smbprn.00000002 Mozilla Firefox 120832 bytes
```

For troubleshooting, it can be useful to stop (pause) the printer. This way the jobs stay in the queue.

```
[root@RHEL4b samba]# lpstat -t
scheduler is running
system default destination: HPColor
device for HPBlack: socket://192.168.1.244:9100
device for HPColor: parallel:/dev/lp0
HPBlack accepting requests since Jan 01 00:00
HPColor accepting requests since Jan 01 00:00
printer HPBlack disabled since Jan 01 00:00 -
Paused
printer HPColor is idle.  enabled since Jan 01 00:00
HPBlack-4           nobody       112640   Sat 07 Jul 2007 07:59:33 AM CEST
HPBlack-5           nobody       120832   Sat 07 Jul 2007 08:00:04 AM CEST
```

34.2. Simple BSD Print Server

The default BSD style print commands (also refered to as LPD/LPR) are defined in rfc 1179. The smb.conf file is similar to the one for CUPS printing, except that CUPS is the default. The file now looks like this.

```
[global]
 server string = Public Anonymous Print Server
 printing = bsd
 load printers = yes

[printers]
 path = /var/spool/samba
 writable = no
 printable = Yes
 public = yes
```

Testparm however gives us some more information on values used for the print commands.

```
[root@RHEL4b samba]# testparm
Load smb config files from /etc/samba/smb.conf
Processing section "[printers]"
Loaded services file OK.
Server role: ROLE_STANDALONE
Press enter to see a dump of your service definitions

# Global parameters
[global]
 server string = Public Anonymous Print Server
 printing = bsd
 print command = lpr -r -P'%p' %s
 lpq command = lpq -P'%p'
 lprm command = lprm -P'%p' %j

[printers]
 path = /var/spool/samba
 guest ok = Yes
 printable = Yes
 browseable = No
[root@RHEL4b samba]#
```

34.3. Simple Unix SysV Print Server

SystemV style printing uses the lp command in this form.

```
lp -dprinter -s file
```

Since by default this command does not remove the file, we have to add this removal to smb.conf. So here is a simple smb.conf to share Unix System V type printers with Samba.

```
[global]
 server string = Public Anonymous Print Server
 printing = sysv
 load printers = yes

[printers]
 path = /var/spool/samba
 writable = no
 printable = Yes
 public = yes
 print command = lp -d%p -s %s ; rm %s
```

34.4. Samba Prining tips

The **printable = Yes** line must always be present in Samba printer shares, even in the **[printers]** section. It is also important to have a naming convention that prevents printers from having the same name as users. The **[homes]** section automatically creates a share for each user with that username, so it cannot be also a printer share.

To troubleshoot the print command, you can da a little trick in smb.conf. Instead of the actual print command, construct the printers section in smb.conf like this.

```
[printers]
 path = /var/spool/samba
 writable = no
 printable = Yes
 public = yes
 print command = echo "lpr -r -P'%p' %s" >> /tmp/bsdprint.log
```

Nothing will be printed, but you can test the print command that is generated by Samba. In this case, the log file looks like this.

```
[root@RHEL4b samba]# cat /tmp/bsdprint.log
lpr -r -P'HP400' smbprn.00000012.ARQtkM
lpr -r -P'HP400' smbprn.00000013.YbFkuN
lpr -r -P'HP400' smbprn.00000017.NeDuGj
[root@RHEL4b samba]#
```

Here is a list variables that are used by Samba for printing.

```
%s filename with path (of the file to be printed)
%f filename without path
%p name of the destination unix printer
%j print job number
```

34.5. Practice

1. Create two printers (with lpadmin or with the cups web interface) and pause(stop) them.

2. Serve these printers with Samba. Connect with a Microsoft computer and test printing.

3. Make sure only Isabelle and Caroline can access one of the printers.

4. Make sure they have to be on the 10.5.0.0/16 subnet to access the printer.

5. If time permits... There are some issues with a BSD printer. Your manager asks you to log the lpr command syntax, its stdout and its stderr to three different files.

Chapter 35. Introduction to Apache

35.1. about apache

According to NetCraft (http://news.netcraft.com/archives/web_server_survey.html) about seventy percent of all web servers are running on Apache. Some people say that the name is derived from **a patchy** web server, because of all the patches people wrote for the NCSA httpd server.

35.2. is apache installed ?

To verify whether Apache is installed, use the proper tools (rpm, dpkg, ...) and grep for apache or httpd.

This Red Hat Enterprise 4 Server has apache installed.

```
[paul@rhel4 ~]$ rpm -qa | grep -i httpd
httpd-2.0.52-25.ent
httpd-manual-2.0.52-25.ent
system-config-httpd-1.3.1-1
httpd-devel-2.0.52-25.ent
httpd-suexec-2.0.52-25.ent
```

This Ubuntu also has apache installed.

```
paul@laika:~$ dpkg -l | grep apache
ii  apache2                  2.2.3-3.2build1      Next generation, scalable, ...
ii  apache2-mpm-prefork      2.2.3-3.2build1      Traditional model for Apach...
ii  apache2-utils            2.2.3-3.2build1      utility programs for webser...
ii  apache2.2-common         2.2.3-3.2build1      Next generation, scalable, ...
ii  libapache2-mod-php5      5.2.1-0ubuntu1.2     server-side, HTML-embedded ...
```

35.3. is apache running ?

This is how apache looks when it is installed on Red Hat Enterprise Linux 4, running named as **httpd**.

```
[root@RHELv4u3 ~]# /etc/init.d/httpd status
httpd is stopped
[root@RHELv4u3 ~]# service httpd start
Starting httpd:                                            [  OK  ]
[root@RHELv4u3 ~]# ps -C httpd
PID TTY          TIME CMD
4573 ?        00:00:00 httpd
4576 ?        00:00:00 httpd
4577 ?        00:00:00 httpd
4578 ?        00:00:00 httpd
4579 ?        00:00:00 httpd
4580 ?        00:00:00 httpd
```

```
4581 ?        00:00:00 httpd
4582 ?        00:00:00 httpd
4583 ?        00:00:00 httpd
[root@RHELv4u3 ~]#
```

And here is Apache running on Ubuntu, named as **apache2**.

```
root@laika:~# ps -C apache2
PID TTY          TIME CMD
6170 ?        00:00:00 apache2
6248 ?        00:00:01 apache2
6249 ?        00:00:01 apache2
6250 ?        00:00:00 apache2
6251 ?        00:00:01 apache2
6252 ?        00:00:01 apache2
7520 ?        00:00:01 apache2
8943 ?        00:00:01 apache2
root@laika:~# /etc/init.d/apache2 status
* Usage: /etc/init.d/apache2 {start|stop|restart|reload|force-reload}
root@laika:~#
```

To verify that apache is running, open a web browser on the web server, and browse to http://localhost. An Apache test page should be shown. The http://localhosts/manual url will give you an extensive Apache manual. The second test is to connect to your Apache from another computer.

35.4. apache configuration

Configuring Apache changed a bit the past couple of years. But it still takes place in **/etc/httpd** or **/etc/apache**.

```
[root@RHELv4u3 ~]# cd /etc/httpd/
[root@RHELv4u3 httpd]# ll
total 32
lrwxrwxrwx  1 root root   25 Jan 24 09:28 build -> ../../usr/lib/httpd/build
drwxr-xr-x  7 root root 4096 Jan 24 08:48 conf
drwxr-xr-x  2 root root 4096 Jan 24 09:29 conf.d
lrwxrwxrwx  1 root root   19 Jan 24 08:48 logs -> ../../var/log/httpd
lrwxrwxrwx  1 root root   27 Jan 24 08:48 modules -> ../../usr/lib/httpd/modules
lrwxrwxrwx  1 root root   13 Jan 24 08:48 run -> ../../var/run
[root@RHELv4u3 httpd]#
```

The main configuration file for the Apache server on RHEL is **/etc/httpd/conf/httpd.conf**, on Ubuntu it is **/etc/apache2/apache2.conf**. The file explains itself, and contains examples for how to set up virtual hosts or configure access.

35.5. virtual hosts

Virtual hosts can be defined by ip-address, by port or by name (host record). (The new way of defining virtual hosts is through seperate config files in the conf.d directory.) Below is a very simple virtual host definition.

```
[root@rhel4 conf]# tail /etc/httpd/conf/httpd.conf
#
# This is a small test website
#
<VirtualHost testsite.local:80>
ServerAdmin webmaster@testsite.local
DocumentRoot /var/www/html/testsite/
ServerName testsite.local
ErrorLog logs/testsite.local-error_log
CustomLog logs/testsite.local-access_log common
</VirtualHost>
[root@rhel4 conf]#
```

Should you put this little index.html file in the directory mentioned in the above screenshot, then you can access this humble website.

```
[root@rhel4 conf]# cat /var/www/html/testsite/index.html
<html>
 <head><title>Test Site</title></head>
 <body>
  <p>This is the test site.</p>
 </body>
</html>
```

Below is a sample virtual host configuration. This virtual hosts overrules the default Apache **ErrorDocument** directive.

```
<VirtualHost 83.217.76.245:80>
ServerName cobbaut.be
ServerAlias www.cobbaut.be
DocumentRoot /home/paul/public_html
ErrorLog /home/paul/logs/error_log
CustomLog /home/paul/logs/access_log common
ScriptAlias /cgi-bin/ /home/paul/cgi-bin/
<Directory /home/paul/public_html>
 Options Indexes IncludesNOEXEC FollowSymLinks
 allow from all
</Directory>
ErrorDocument 404 http://www.cobbaut.be/cobbaut.php
</VirtualHost>
```

35.6. aliases and redirects

Apache supports aliases for directories, like this example shows.

```
Alias /paul/ "/home/paul/public_html/"
```

Similarly, content can be redirected to another website or web server.

```
Redirect permanent /foo http://www.foo.com/bar
```

35.7. securing directories with htpasswd and .htaccess

You can secure files and directories in your website with a userid/password. First, enter your website, and use the **htpasswd** command to create a **.htpasswd file** that contains a userid and an (encrypted) password.

```
[root@rhel4 testsite]# htpasswd -c .htpasswd pol
New password:
Re-type new password:
Adding password for user pol
[root@rhel4 testsite]# cat .htpasswd
pol:x5vZlyw1V6KXE
[root@rhel4 testsite]#
```

You can add users to this file, just don't use the -c switch again.

```
[root@rhel4 testsite]# htpasswd .htpasswd kim
New password:
Re-type new password:
Adding password for user kim
[root@rhel4 testsite]# cat .htpasswd
pol:x5vZlyw1V6KXE
kim:6/RbvugwsgOI6
[root@rhel4 testsite]#
```

You have now defined two users. Next create a subsdirectory that you want to protect with these two accounts. And put the following .htaccess file in that subdirectory.

```
[root@rhel4 kimonly]# pwd
/var/www/html/testsite/kimonly
[root@rhel4 kimonly]# cat .htaccess
AuthUserFile /var/www/html/testsite/.htpasswd
AuthGroupFile /dev/null
AuthName "test access title"
AuthType Basic

<Limit GET POST>
require valid-user
</Limit>
[root@rhel4 kimonly]#
```

Finally, don't forget to verify that AllowOverride is set to All in the general Apache configuration file.

```
# AllowOverride controls what directives may be placed in .htaccess files.
# It can be "All", "None", or any combination of the keywords:
#   Options FileInfo AuthConfig Limit
#
AllowOverride All
```

From now on, when a user accesses a file in that subdirectory, that user will have to provide a userid/password combo that is defined in your .htpasswd.

35.8. more on .htaccess

You can do much more with **.htaccess**. One example is to use .htaccess to prevent people from certain domains to access your website. Like in this case, where a number of referer spammers are blocked from the website.

```
paul@lounge:~/cobbaut.be$ cat .htaccess
# Options +FollowSymlinks
RewriteEngine On
RewriteCond %{HTTP_REFERER} ^http://(www\.)?buy-adipex.fw.nu.*$ [OR]
RewriteCond %{HTTP_REFERER} ^http://(www\.)?buy-levitra.asso.ws.*$ [NC,OR]
RewriteCond %{HTTP_REFERER} ^http://(www\.)?buy-tramadol.fw.nu.*$ [NC,OR]
RewriteCond %{HTTP_REFERER} ^http://(www\.)?buy-viagra.lookin.at.*$ [NC,OR]
...
RewriteCond %{HTTP_REFERER} ^http://(www\.)?www.healthinsurancehelp.net.*$ [NC]
RewriteRule .* - [F,L]
paul@lounge:~/cobbaut.be$
```

35.9. traffic

Apache keeps a log of all visitors. The **webalizer** is often used to parse this log into nice html statistics.

35.10. practice: apache

1. Verify that Apache is installed and running.

2. Browse to the Apache HTML manual from another computer.

3. Create a virtual hosts that listens to port 8247.

4. Create a virtual hosts that listens on another ip-address.

5. Test from another computer that all virtual hosts work.

6. Protect a subdirectory of a website with .htpasswd and .htaccess.

Chapter 36. Routers

36.1. terminology

36.1.1. router or firewall

A router is a device that connects two networks. A **firewall** is a device that besides acting as a **router**, also contains (and implements) rules to determine whether packets are allowed to travel from one network to another. A firewall can be configured to block access based on networks, hosts, protocols and ports. Firewalls can also change the contents of packets while forwarding them.

36.1.2. packet forwarding

Packet forwarding means allowing packets to go from one network to another. When a multihomed host is connected to two different networks, and it allows packets to travel from one network to another through its two network interfaces, it is said to have enabled **packet forwarding**.

36.1.3. packet filtering

Packet filtering is very similar to packet forwarding, but every packet is individually tested against rules that decide on allowing or dropping the packet. The rules are stored by iptables.

36.1.4. stateful

A **stateful** firewall is an advancement over stateless firewalls that inspect every individual packet. A stateful firewall will keep a table of active connections, and is knowledgeable enough to recognise when new connections are part of an active session. Linux iptables is a stateful firewall.

36.1.5. NAT (network address translation)

A **NAT** device is a router that is also changing the source and/or target ip-address in packets. It is typically used to connect multiple computers in a private address range (rfc 1918) with the (public) internet. A NAT can hide private addresses from the internet.

It is important to understand that people and vendors do not always use the right term when referring to a certain type of NAT. Be sure you talk about the same thing. We can distuinguish several types of NAT.

36.1.5.1. PAT (port address translation)

NAT often includes PAT. A **PAT** device is a router that is also changing the source and/or target tcp/udp port in packets. PAT is Cisco terminology and is used by **SNAT**, **DNAT**, **masquerading** and **port forwarding** in Linux. RFC 3022 calls it **NAPT** and defines the NAT/PAT combo as "traditional NAT". A device sold to you as a NAT-device will probably do NAT and PAT.

36.1.5.2. SNAT (source network address translation)

A **SNAT** device is changing the source ip-address when a packet passes our NAT. SNAT configuration with iptables includes a fixed target source address.

36.1.5.3. masquerading

Masquerading is a form of SNAT that will hide the (private) source ip-addresses of your private network using a public ip-address. Masquerading is common on dynamic internet interfaces (broadband modem/routers). Masquerade configuration with iptables uses a dynamic target source address.

36.1.5.4. DNAT (destination network address translation)

A **DNAT** device is changing the destination ip-address when a packet passes our NAT.

36.1.5.5. port forwarding

When static DNAT is set up in a way that allows outside connections to enter our private network, then we call it **port forwarding**.

36.2. packet forwarding

36.2.1. about packet forwarding

Packet forwarding means allowing packets to go from one network to another. When a multihomed host is connected to two different networks, and it allows packets to travel from one network to another through its two network interfaces, it is said to have enabled packet forwarding.

36.2.2. /proc/sys/net/ipv4/ip_forward

Whether a host is forwarding packets is defined in **/proc/sys/net/ipv4/ip_forward**. The following screenshot shows how to enable packet forwarding on Linux.

```
[root@RHEL5 ~]# echo 1 > /proc/sys/net/ipv4/ip_forward
```

The next command shows how to disable packet forwarding.

```
[root@RHEL5 ~]# echo 0 > /proc/sys/net/ipv4/ip_forward
```

Use cat to check if packet forwarding is enabled.

```
[root@RHEL5 ~]# cat /proc/sys/net/ipv4/ip_forward
```

36.2.3. /etc/sysctl.conf

By default, most Linux computers are not configured for automatic packet forwarding. To enable packet forwarding whenever the system starts, change the **net.ipv4.ip_forward** variable in **/etc/sysctl.conf** to the value 1.

```
[root@RHEL5 ~]# grep ip_forward /etc/sysctl.conf
net.ipv4.ip_forward = 0
```

36.2.4. Practice: packet forwarding

1. Set up two dsl (Damn Small Linux) machines, one on vmnet1, the other on vmnet8. Make sure they both get an ip-address in the correct subnet. These two machines will be 'left' and 'right' from the 'router'.

2. Set up a RHEL server with two network cards, one on vmnet1, the other on vmnet8. This computer will be the 'router'. Complete the table below with the relevant names, ip-addresses and mac-addresses.

Table 36.1. Packet Forwarding Exercise

	left:	**router:**		**right:**
MAC				
IP				

3. How can you verify whether the RHEL will allow packet forwarding by default or not ? Test that you can ping from the RHEL to the two dsl machines, and from the two dsl machines to the RHEL. Use **arp -a** to make sure you are connected with the correct MAC addresses.

4. Ping from one dsl to the other. Enable and/or disable packet forwarding on the RHEL server and verify what happens to the ping between the two dsl machines. If you do not succeed in pinging between the two dsl machines (on different subnets), then use a sniffer like wireshark or tcpdump to discover the problem.

5. Use wireshark or tcpdump -xx to answer the following questions. Does the source MAC change when a packet passes through the filter ? And the destination MAC ? What about source and destination IP-addresses ?

36.2.5. Solution: packet forwarding

1. Set up two dsl (Damn Small Linux) machines, one on vmnet1, the other on vmnet8. Make sure they both get an ip-address in the correct subnet. These two machines will be 'left' and 'right' from the 'router'.

The configuration of the dsl machines can be similar to the following two screenshots. Both machines must be in a different subnet (here 192.168.187.0/24 and 172.16.122.0/24)

```
root@ttyp1[root]# ifconfig eth0 | grep -A1 eth0
eth0 Link encap:Ethernet  HWaddr 00:0C:29:08:F4:C1
     inet addr:192.168.187.130  Bcast:192.168.187.255  Mask:255.255.255.0
root@ttyp1[root]# route
Kernel IP routing table
Destination    Gateway         Genmask        Flags Metric Ref  Use Iface
192.168.187.0  *               255.255.255.0  U     0      0      0 eth0
default        192.168.187.128 0.0.0.0        UG    0      0      0 eth0
root@ttyp1[root]#
```

```
root@ttyp1[root]# ifconfig eth0 | grep -A1 eth0
eth0 Link encap:Ethernet  HWaddr 00:0C:29:6E:1A:AA
     inet addr:172.16.122.129  Bcast:172.16.122.255  Mask:255.255.255.0
root@ttyp1[root]# route
Kernel IP routing table
Destination    Gateway         Genmask        Flags Metric Ref  Use Iface
172.16.122.0   *               255.255.255.0  U     0      0      0 eth0
default        172.16.122.128  0.0.0.0        UG    0      0      0 eth0
root@ttyp1[root]#
```

2. Set up a RHEL server with two network cards, one on vmnet1, the other on vmnet8. This computer will be the 'router'.

The 'router' can be set up like this screenshot shows.

```
[root@RHEL5 ~]# ifconfig | grep -A1 eth
eth1 Link encap:Ethernet  HWaddr 00:0C:29:8C:90:49
     inet addr:192.168.187.128  Bcast:192.168.187.255  Mask:255.255.255.0
--
eth2 Link encap:Ethernet  HWaddr 00:0C:29:8C:90:53
     inet addr:172.16.122.128  Bcast:172.16.122.255  Mask:255.255.255.0
[root@RHEL5 ~]#
```

Your setup may use different ip and mac addresses than the ones in the table below. This table serves as a reference for the screenshots from this solution to the practice.

Table 36.2. Packet Forwarding Solution

left: dsl	router: RHEL5		right: dsl
00:0c:29:08:f4:c1	00:0c:29:8c:90:49	00:0c:29:8c:90:53	00:0c:29:6e:1a:aa
192.168.187.130	192.168.187.128	172.16.122.128	172.16.122.129

3. How can you verify whether the RHEL will allow packet forwarding by default or not ? Test that you can ping from the RHEL to the two dsl machines, and from the two dsl machines to the RHEL. Use **arp -a** to make sure you are connected with the correct MAC addresses.

This can be done with "**grep ip_forward /etc/sysctl.conf**" (1 is enabled, 0 is disabled).

```
[root@RHEL5 ~]# grep ip_for /etc/sysctl.conf
net.ipv4.ip_forward = 0
```

4. Ping from one dsl to the other. Enable and/or disable packet forwarding on the RHEL server and verify what happens to the ping between the two dsl machines. If you do not succeed in pinging between the two dsl machines (on different subnets), then use a sniffer like ethereal or tcpdump to discover the problem.

Did you forget to add a default gateway to the dsl machines ? Use **route add default gw 'ip-address'**.

You should be able to ping when packet forwarding is enabled (and both default gateways are properly configured). The ping will not work when packet forwarding is disabled or when gateways are not configured correctly.

5. Use wireshark or tcpdump -xx to answer the following questions. Does the source MAC change when a packet passes through the filter ? And the destination MAC ? What about source and destination IP-addresses ?

Both MAC addresses are changed when passing the router. The screenshots below show tcpdump -xx output on the router. The first one is taken on the eth1(vmnet1) interface in the 192.168.187.0/24 network, the second one is from the other interface (eth2 on vmnet8 in 172.16.122.0/24). The first six bytes are the destination MAC, the next six are the source.

```
[root@RHEL5 ~]# tcpdump -xx -i eth1
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth1, link-type EN10MB (Ethernet), capture size 96 bytes
04:18:23.817854 IP 192.168.187.130 > 172.16.122.129: ICMP echo request...
 0x0000:  000c 298c 9049 000c 2908 f4c1 0800 4500
 0x0010:  0054 0000 4000 4001 97ec c0a8 bb82 ac10
 0x0020:  7a81 0800 3b28 a717 0006 8059 d148 d614
 0x0030:  0300 0809 0a0b 0c0d 0e0f 1011 1213 1415
 0x0040:  1617 1819 1a1b 1c1d 1e1f 2021 2223 2425
 0x0050:  2627 2829 2a2b 2c2d 2e2f 3031 3233 3435
04:18:23.817962 IP 172.16.122.129 > 192.168.187.130: ICMP echo reply...
```

```
 0x0000:  000c 2908 f4c1 000c 298c 9049 0800 4500
 0x0010:  0054 d364 0000 3f01 0588 ac10 7a81 c0a8
 0x0020:  bb82 0000 4328 a717 0006 8059 d148 d614
 0x0030:  0300 0809 0a0b 0c0d 0e0f 1011 1213 1415
 0x0040:  1617 1819 1a1b 1c1d 1e1f 2021 2223 2425
 0x0050:  2627 2829 2a2b 2c2d 2e2f 3031 3233 3435
```

```
[root@RHEL5 ~]# tcpdump -xx -i eth2
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth2, link-type EN10MB (Ethernet), capture size 96 bytes
04:18:33.904697 IP 192.168.187.130 > 172.16.122.129: ICMP echo request...
 0x0000:  000c 296e 1aaa 000c 298c 9053 0800 4500
 0x0010:  0054 0000 4000 3f01 98ec c0a8 bb82 ac10
 0x0020:  7a81 0800 2320 a717 0008 8a59 d148 e41a
 0x0030:  0300 0809 0a0b 0c0d 0e0f 1011 1213 1415
 0x0040:  1617 1819 1a1b 1c1d 1e1f 2021 2223 2425
 0x0050:  2627 2829 2a2b 2c2d 2e2f 3031 3233 3435
04:18:33.944514 IP 172.16.122.129 > 192.168.187.130: ICMP echo reply...
 0x0000:  000c 298c 9053 000c 296e 1aaa 0800 4500
 0x0010:  0054 d366 0000 4001 0486 ac10 7a81 c0a8
 0x0020:  bb82 0000 2b20 a717 0008 8a59 d148 e41a
 0x0030:  0300 0809 0a0b 0c0d 0e0f 1011 1213 1415
 0x0040:  1617 1819 1a1b 1c1d 1e1f 2021 2223 2425
 0x0050:  2627 2829 2a2b 2c2d 2e2f 3031 3233 3435
```

Chapter 37. Iptables

37.1. about iptables

Iptables is a user-space application that allows a user to configure the Linux kernel's Netfilter. By default there are three tables in the kernel that contain sets of rules. The filter table is used for packet filtering, the NAT table for address translation and the mangle table for special-purpose processing of packets. Series of rules in each table are called a **chain**.

The following screenshot shows how to stop and start iptables.

```
[root@RHEL5 ~]# /etc/init.d/iptables stop
[root@RHEL5 ~]# /etc/init.d/iptables start
[root@RHEL5 ~]#
```

37.2. packet filtering

37.2.1. about packet filtering

Packet filtering is a bit more than packet forwarding. Packet forwarding only uses a routing table to make decisions, the kernel now also uses a list of rules. So with packet filtering, the kernel will inspect each packet and decide based on iptables rules to allow or drop a packet.

37.2.2. filter table

The filter table in iptables has three chains (sets of rules). The INPUT chain is used for any packet coming into the system. The OUTPUT chain is for any packet leaving the system. And the FORWARD chain is for packets that are forwarded (routed) through the system.

The screenshot below shows how to list the filter table and all its rules.

```
[root@RHEL5 ~]# iptables -t filter -nL
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
[root@RHEL5 ~]#
```

As you can see, all three chains in the filter table are set to ACCEPT everything. ACCEPT is the default behaviour.

37.2.3. Changing default policy rules

To start, let's set the default policy for all three chains to drop everything. Note that you might lose your connection when typing this over ssh ;-).

```
[root@RHEL5 ~]# iptables -P INPUT DROP
[root@RHEL5 ~]# iptables -P FORWARD DROP
[root@RHEL5 ~]# iptables -P OUTPUT DROP
```

Next, we allow the server to use its own loopback device (this allows the server to access its services running on localhost). We first append a rule to the INPUT chain to allow (ACCEPT) traffic from the lo (loopback) interface, then we do the same to allow packets to leave the system through the loopback interface.

```
[root@RHEL5 ~]# iptables -A INPUT -i lo -j ACCEPT
[root@RHEL5 ~]# iptables -A OUTPUT -o lo -j ACCEPT
```

Looking at the filter table again (omitting -t filter because it is the default table).

```
[root@RHEL5 ~]# iptables -nL
Chain INPUT (policy DROP)
target     prot opt source               destination
ACCEPT     all  --  0.0.0.0/0            0.0.0.0/0

Chain FORWARD (policy DROP)
target     prot opt source               destination

Chain OUTPUT (policy DROP)
target     prot opt source               destination
ACCEPT     all  --  0.0.0.0/0            0.0.0.0/0
```

37.2.4. Allowing ssh over eth0

This example show how to add two rules to allow ssh access to your system from outside.

```
[root@RHEL5 ~]# iptables -A INPUT -i eth0 -p tcp --dport 22 -j ACCEPT
[root@RHEL5 ~]# iptables -A OUTPUT -o eth0 -p tcp --sport 22 -j ACCEPT
```

The filter table will look something like this screenshot (note that -v is added for more verbose output).

```
[root@RHEL5 ~]# iptables -nvL
Chain INPUT (policy DROP 7 packets, 609 bytes)
 pkts bytes target prot opt in    out   source     destination
    0     0 ACCEPT all  --  lo    *     0.0.0.0/0  0.0.0.0/0
    0     0 ACCEPT tcp  --  eth0  *     0.0.0.0/0  0.0.0.0/0  tcp dpt:22

Chain FORWARD (policy DROP 0 packets, 0 bytes)
 pkts bytes target prot opt in    out   source     destination
```

```
Chain OUTPUT (policy DROP 3 packets, 228 bytes)
 pkts bytes target prot opt in    out   source     destination
    0     0 ACCEPT all  --  *     lo    0.0.0.0/0  0.0.0.0/0
    0     0 ACCEPT tcp  --  *     eth0  0.0.0.0/0  0.0.0.0/0  tcp spt:22
[root@RHEL5 ~]#
```

37.2.5. Allowing access from a subnet

This example shows how to allow access from any computer in the 10.1.1.0/24 network, but only through eth1. There is no port (application) limitation here.

```
[root@RHEL5 ~]# iptables -A INPUT -i eth1 -s 10.1.1.0/24 -p tcp -j ACCEPT
[root@RHEL5 ~]# iptables -A OUTPUT -o eth1 -d 10.1.1.0/24 -p tcp -j ACCEPT
```

Together with the previous examples, the policy is expanding.

```
[root@RHEL5 ~]# iptables -nvL
Chain INPUT (policy DROP 7 packets, 609 bytes)
 pkts bytes target prot opt in    out   source      destination
    0     0 ACCEPT all  --  lo    *     0.0.0.0/0   0.0.0.0/0
    0     0 ACCEPT tcp  --  eth0  *     0.0.0.0/0   0.0.0.0/0  tcp dpt:22
    0     0 ACCEPT tcp  --  eth1  *     10.1.1.0/24 0.0.0.0/0

Chain FORWARD (policy DROP 0 packets, 0 bytes)
 pkts bytes target prot opt in    out   source      destination

Chain OUTPUT (policy DROP 3 packets, 228 bytes)
 pkts bytes target prot opt in    out   source      destination
    0     0 ACCEPT all  --  *     lo    0.0.0.0/0   0.0.0.0/0
    0     0 ACCEPT tcp  --  *     eth0  0.0.0.0/0   0.0.0.0/0  tcp spt:22
    0     0 ACCEPT tcp  --  *     eth1  0.0.0.0/0   10.1.1.0/24
```

37.2.6. iptables save

Use **iptables save** to automatically implement these rules when the firewall is (re)started.

```
[root@RHEL5 ~]# /etc/init.d/iptables save
Saving firewall rules to /etc/sysconfig/iptables:          [  OK  ]
[root@RHEL5 ~]#
```

37.2.7. scripting example

You can write a simple script for these rules. Below is an example script that implements the firewall rules that you saw before in this chapter.

```
#!/bin/bash
# first cleanup everything
iptables -t filter -F
iptables -t filter -X
iptables -t nat -F
iptables -t nat -X

# default drop
iptables -P INPUT DROP
iptables -P FORWARD DROP
iptables -P OUTPUT DROP

# allow loopback device
iptables -A INPUT -i lo -j ACCEPT
iptables -A OUTPUT -o lo -j ACCEPT

# allow ssh over eth0 from outside to system
iptables -A INPUT -i eth0 -p tcp --dport 22 -j ACCEPT
iptables -A OUTPUT -o eth0 -p tcp --sport 22 -j ACCEPT

# allow any traffic from 10.1.1.0/24 to system
iptables -A INPUT -i eth1 -s 10.1.1.0/24 -p tcp -j ACCEPT
iptables -A OUTPUT -o eth1 -d 10.1.1.0/24 -p tcp -j ACCEPT
```

37.2.8. Allowing ICMP(ping)

When you enable iptables, you will get an **'Operation not permitted'** message when trying to ping other hosts.

```
[root@RHEL5 ~# ping 192.168.187.130
PING 192.168.187.130 (192.168.187.130) 56(84) bytes of data.
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
```

The screenshot below shows you how to setup iptables to allow a ping from or to your machine.

```
[root@RHEL5 ~]# iptables -A INPUT -p icmp --icmp-type any -j ACCEPT
[root@RHEL5 ~]# iptables -A OUTPUT -p icmp --icmp-type any -j ACCEPT
```

The previous two lines do not allow other computers to route ping messages through your router, because it only handles INPUT and OUTPUT. For routing of ping, you will need to enable it on the FORWARD chain. The following command enables routing of icmp messages between networks.

```
[root@RHEL5 ~]# iptables -A FORWARD -p icmp --icmp-type any -j ACCEPT
```

37.2.9. Practice: packet filtering

1. Make sure you can ssh to your router-system when iptables is active.

2. Make sure you can ping to your router-system when iptables is active.

3. Define one of your networks as 'internal' and the other as 'external'. Configure the router to allow visits to a website (http) to go from the internal network to the external network (but not in the other direction).

4. Make sure the internal network can ssh to the external, but not the other way around.

37.2.10. Solution: packet filtering

A possible solution, where dsl is the internal and dsr is the external network.

```
#!/bin/bash

# first cleanup everything
iptables -t filter -F
iptables -t filter -X
iptables -t nat -F
iptables -t nat -X

# default drop
iptables -P INPUT DROP
iptables -P FORWARD DROP
iptables -P OUTPUT DROP

# allow loopback device
iptables -A INPUT -i lo -j ACCEPT
iptables -A OUTPUT -o lo -j ACCEPT

# question 1: allow ssh over eth0
iptables -A INPUT -i eth0 -p tcp --dport 22 -j ACCEPT
iptables -A OUTPUT -o eth0 -p tcp --sport 22 -j ACCEPT

# question 2: Allow icmp(ping) anywhere
iptables -A INPUT -p icmp --icmp-type any -j ACCEPT
iptables -A FORWARD -p icmp --icmp-type any -j ACCEPT
iptables -A OUTPUT -p icmp --icmp-type any -j ACCEPT

# question 3: allow http from internal(dsl) to external(dsr)
iptables -A FORWARD -i eth1 -o eth2 -p tcp --dport 80 -j ACCEPT
iptables -A FORWARD -i eth2 -o eth1 -p tcp --sport 80 -j ACCEPT

# question 4: allow ssh from internal(dsl) to external(dsr)
iptables -A FORWARD -i eth1 -o eth2 -p tcp --dport 22 -j ACCEPT
iptables -A FORWARD -i eth2 -o eth1 -p tcp --sport 22 -j ACCEPT

# allow http from external(dsr) to internal(dsl)
# iptables -A FORWARD -i eth2 -o eth1 -p tcp --dport 80 -j ACCEPT
# iptables -A FORWARD -i eth1 -o eth2 -p tcp --sport 80 -j ACCEPT

# allow rpcinfo over eth0 from outside to system
# iptables -A INPUT -i eth2 -p tcp --dport 111 -j ACCEPT
# iptables -A OUTPUT -o eth2 -p tcp --sport 111 -j ACCEPT
```

37.3. network address translation

37.3.1. about NAT

A NAT device is a router that is also changing the source and/or target ip-address in packets. It is typically used to connect multiple computers in a private address range with the (public) internet. A NAT can hide private addresses from the internet.

NAT was developed to mitigate the use of real ip addresses, to allow private address ranges to reach the internet and back, and to not disclose details about internal networks to the outside.

The nat table in iptables adds two new chains. PREROUTING allows altering of packets before they reach the INPUT chain. POSTROUTING allows altering packets after they exit the OUTPUT chain.

Use **iptables -t nat -nvL** to look at the NAT table. The screenshot below shows an empty NAT table.

```
[root@RHEL5 ~]# iptables -t nat -nL
Chain PREROUTING (policy ACCEPT)
target     prot opt source               destination

Chain POSTROUTING (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
[root@RHEL5 ~]#
```

37.3.2. SNAT (Source NAT)

The goal of source nat is to change the source address inside a packet before it leaves the system (e.g. to the internet). The destination will return the packet to the NAT-device. This means our NAT-device will need to keep a table in memory of all the packets it changed, so it can deliver the packet to the original source (e.g. in the private network).

Because SNAT is about packets leaving the system, it uses the POSTROUTING chain.

Here is an example SNAT rule. The rule says that packets coming from 10.1.1.0/24 network and exiting via eth1 will get the source ip-address set to 11.12.13.14. (Note that this is a one line command!)

```
iptables -t nat -A POSTROUTING -o eth1 -s 10.1.1.0/24 -j SNAT \
--to-source 11.12.13.14
```

Of course there must exist a proper iptables filter setup to allow the packet to traverse from one network to the other.

37.3.3. SNAT example setup

This example script uses a typical nat setup. The internal (eth0) network has access via SNAT to external (eth1) webservers (port 80).

```
#!/bin/bash
#
# iptables script for simple classic nat websurfing
# eth0 is internal network, eth1 is internet
#
echo 0 > /proc/sys/net/ipv4/ip_forward
iptables -P INPUT ACCEPT
iptables -P OUTPUT ACCEPT
iptables -P FORWARD DROP
iptables -A FORWARD -i eth0 -o eth1 -s 10.1.1.0/24 -p tcp \
--dport 80 -j ACCEPT
iptables -A FORWARD -i eth1 -o eth0 -d 10.1.1.0/24 -p tcp \
--sport 80 -j ACCEPT
iptables -t nat -A POSTROUTING -o eth1 -s 10.1.1.0/24 -j SNAT \
--to-source 11.12.13.14
echo 1 > /proc/sys/net/ipv4/ip_forward
```

37.3.4. IP masquerading

IP masquerading is very similar to SNAT, but is meant for dynamic interfaces. Typical example are broadband 'router/modems' connected to the internet and receiving a different ip-address from the isp, each time they are cold-booted.

The only change needed to convert the SNAT script to a masquerading is one line.

```
iptables -t nat -A POSTROUTING -o eth1 -s 10.1.1.0/24 -j MASQUERADE
```

37.3.5. DNAT (Destination NAT)

DNAT is typically used to allow packets from the internet to be redirected to an internal server (in your DMZ) and in a private address range that is inaccessible directly form the internet.

This example script allows internet users to reach your internal (192.168.1.99) server via ssh (port 22).

```
#!/bin/bash
#
# iptables script for DNAT
# eth0 is internal network, eth1 is internet
#
echo 0 > /proc/sys/net/ipv4/ip_forward
iptables -P INPUT ACCEPT
iptables -P OUTPUT ACCEPT
iptables -P FORWARD DROP
iptables -A FORWARD -i eth0 -o eth1 -s 10.1.1.0/24 -j ACCEPT
iptables -A FORWARD -i eth1 -o eth0 -p tcp --dport 22 -j ACCEPT
iptables -t nat -A PREROUTING -i eth1 -p tcp --dport 22 \
-j DNAT --to-destination 10.1.1.99
echo 1 > /proc/sys/net/ipv4/ip_forward
```

Chapter 38. Introduction to squid

38.1. about proxy servers

38.1.1. usage

A **proxy server** is a server that caches the internet. Clients connect to the proxy server with a request for an internet server. The proxy server will connect to the internet server on behalf of the client. The proxy server will also cache the pages retrieved from the internet server. A proxy server may provide pages from his cache to a client, instead of connecting to the internet server to retrieve the (same) pages.

A proxy server has two main advantages. It improves web surfing speed when returning cached data to clients, and it reduces the required bandwidth (cost) to the internet.

Smaller organizations sometimes put the proxy server on the same physical computer that serves as a NAT to the internet. In larger organizations, the proxy server is one of many servers in the DMZ.

When web traffic passes via a proxy server, it is common practice to configure the proxy with extra settings for access control. Access control in a proxy server can mean user account access, but also website(url), ip-address or dns restrictions.

38.1.2. open proxy servers

You can find lists of open proxy servers on the internet that enable you to surf anonymously. This works when the proxy server connects on your behalf to a website, without logging your ip-address. But be careful, these (listed) open proxy servers could be created in order to eavesdrop upon their users.

38.1.3. squid

This chapter is an introduction to the **squid** proxy server (http://www.squid-cache.org). The version used is 2.5.

```
[root@RHEL4 ~]# rpm -qa | grep squid
squid-2.5.STABLE6-3.4E.12
[root@RHEL4 ~]#
```

38.2. squid proxy server

38.2.1. /etc/squid/squid.conf

Squid's main configuration file is **/etc/squid/squid.conf**. The file explains every parameter in great detail. It can be a good idea to start by creating a backup of this file.

```
[root@RHEL4 /etc/squid/]# cp squid.conf squid.conf.original
```

38.2.2. /var/spool/squid

The **squid** proxy server stores its cache by default in **/var/spool/squid**. This setting is configurable in /etc/squid/squid.conf.

```
[root@RHEL4 ~]# grep "^# cache_dir" /etc/squid/squid.conf
# cache_dir ufs /var/spool/squid 100 16 256
```

It is possible that in a default setup where squid has never run, that the /var/spool/squid directories do not exist.

```
[root@RHEL4 ~]# ls -al /var/spool/squid
ls: /var/spool/squid: No such file or directory
```

Running **squid -z** will create the necessary squid directories.

```
[root@RHEL4 ~]# squid -z
2008/09/22 14:07:47| Creating Swap Directories
[root@RHEL4 ~]# ls -al /var/spool/squid
total 80
drwxr-x---   18 squid squid 4096 Sep 22 14:07 .
drwxr-xr-x   26 root  root  4096 May 30  2007 ..
drwxr-xr-x  258 squid squid 4096 Sep 22 14:07 00
drwxr-xr-x  258 squid squid 4096 Sep 22 14:07 01
drwxr-xr-x  258 squid squid 4096 Sep 22 14:07 02
...
```

38.2.3. port 3128 or port 8080

By default the squid proxy server will bind to port 3128 to listen to incoming requests.

```
[root@RHEL4 ~]# grep "default port" /etc/squid/squid.conf
#       The default port number is 3128.
```

Many organizations use port 8080 instead.

```
[root@RHEL4 ~]# grep 8080 /etc/squid/squid.conf
http_port 8080
```

38.2.4. /var/log/squid

The standard log file location for squid is **/var/log/squid**.

```
[root@RHEL4 ~]# grep "/var/log" /etc/squid/squid.conf
# cache_access_log /var/log/squid/access.log
# cache_log /var/log/squid/cache.log
# cache_store_log /var/log/squid/store.log
```

38.2.5. access control

The default squid setup only allows localhost access. To enable access for a private network range, look for the "INSERT YOUR OWN RULE(S) HERE..." sentence in squid.conf and add two lines similar to the screenshot below.

```
# INSERT YOUR OWN RULE(S) HERE TO ALLOW ACCESS FROM YOUR CLIENTS

acl company_network src 192.168.1.0/24
http_access allow company_network
```

Restart the squid server, and now the local private network can use the proxy cache.

38.2.6. testing squid

First, make sure that the server running squid has access to the internet.

```
[root@RHEL4 ~]# wget -q http://linux-training.be/index.html
[root@RHEL4 ~]# ls -l index.html
-rw-r--r--  1 root root 2269 Sep 18 13:18 index.html
[root@RHEL4 ~]#
```

Then configure a browser on a client to use the proxy server. OR you could set the HTTP_PROXY (sometimes http_proxy) variable to point command line programs to the proxy.

```
[root@fedora ~]# export HTTP_PROXY=http://192.168.1.39:8080
[root@ubuntu ~]# export http_proxy=http://192.168.1.39:8080
```

Testing a client machine can then be done with wget (wget -q is used to simplify the screenshot).

```
[root@RHEL5 ~]# > /etc/resolv.conf
```

```
[root@RHEL5 ~]# wget -q http://www.linux-training.be/index.html
[root@RHEL5 ~]# ls -l index.html
-rw-r--r-- 1 root root 2269 Sep 18  2008 index.html
[root@RHEL5 ~]#
```

38.2.7. name resolution

You need name resolution working on the squid server, but you don't need name resolution on the clients.

```
[paul@RHEL5 ~]$ wget http://grep.be
--14:35:44--  http://grep.be
Resolving grep.be... failed: Temporary failure in name resolution.
[paul@RHEL5 ~]$ export http_proxy=http://192.168.1.39:8080
[paul@RHEL5 ~]$ wget http://grep.be
--14:35:49--  http://grep.be/
Connecting to 192.168.1.39:8080... connected.
Proxy request sent, awaiting response... 200 OK
Length: 5390 (5.3K) [text/html]
Saving to: `index.html.1'

100%[================================>] 5,390       --.-K/s   in 0.1s

14:38:29 (54.8 KB/s) - `index.html' saved [5390/5390]

[paul@RHEL5 ~]$
```

Chapter 39. Introduction to bind

39.1. DNS History

Today, **DNS** or **Domain Name System** is a worldwide distributed hierarchical database. It's primary function is to resolve names to ip addresses, and to point to internet servers providing SMTP or LDAP services.

In the seventies, only a few hundred computers were connected to the internet. To resolve names, computers had a flat file that contained a table to resolve hostnames to ip-addresses. This local file was downloaded from hosts.txt on an ftp server in Stanford.

In 1984 **Paul Mockapetris** created DNS, a distributed treelike hierarchical database.

ICANN..............

39.2. DNS Structure

39.2.1. root

DNS is a tree structure. The top of the tree is called the **root**. There are thirteen root servers on the internet, they are named A to M. Journalist often refer to these servers as **the master servers of the internet**, because if these servers go down, then nobody can (use names to) connect to websites.

The root servers are not thirteen physical machines, in fact ... (expand later with mirror info...)

39.2.2. top level domains (TLD)

Below the root level are the **top level domains** or **TLD's**. Originally there were only seven defined.com(mercial),.edu(cational),.gov(ernment),.int(ernational),.mil(ilitary),.net(work) and .org for non-commercial organizations. The .arpa domain was also used, it will be explained later.

Country TLD's were defined for individual countries, like .be for Belgium and .fr for France.

In the 21st century new TLD's were defined like .museum .info .biz and .aero.

39.2.3. domains

One level below the top level domains are the **domains**. Examples of domain names are google.com or linux-training.be. Domains can have subdomains (also called child domains).

39.2.4. fully qualified domain name

The **fully qualified domain name** or **FQDN** is the combination of the hostname of a machine appended with its domain name.

If for example a system is called **wolf** and it is in the domain **stargate.local**, then the FQDN of this system is **wolf.stargate.local**.

39.3. How DNS works

39.3.1. zone

A zone is a portion of the DNS tree. A DNS server that is controlling a zone, is said to be the **authoritative** DNS server for that zone. A zone is a collection of **resource records**. There are several types of resource records, for example A, PTR, NS, MX, SOA and CNAME.

39.3.1.1. A record

The **A record**, which is also called a **host record** contains the ipv4-address of a computer. When a DNS client queries a DNS server for an A record, then the DNS server will resolve the hostname in the query to an ip-address. An **AAAA record** is similar but contains an ipv6 address instead of ipv4.

39.3.1.2. PTR record

A **PTR record** is the reverse of an A record. It contains the name of a computer and can be used to resolve an ip-address to a hostname.

39.3.1.3. NS record

A **NS record** or **nameserver record** is a record that points to a DNS name server (in this zone). You can list all your name servers for your DNS zone in distinct NS records.

39.3.1.4. SOA record

The SOA record of a zone contains meta information about the zone itself. The contents of the SOA record is explained in detail in the section about zone transfers. There is exactly one SOA record for each zone.

39.3.1.5. CNAME record

A **CNAME record** maps a hostname to a hostname, creating effectively an alias for an existing hostname. The name of the mail server is often aliased to **mail** or **smtp**, and the name of a web server to **www**.

39.3.1.6. MX record

The **MX** points to an SMTP server. When you send an email to another domain, then your mail server will need the MX record of the target domain's mail server.

39.3.2. master and slave

There are several reasons to create more than one name server in a zone. One server might not be able to answer to all queries, or you might want some fault tolerance to mitigate the impact of hardware failure. When adding a **secondary DNS server** to a zone, then you will configure this server as a **slave server** to the **primary server**. The primary server then becomes the **master server** of the slave server.

Very often the primary DNS server is the master server of all slaves. Sometimes a slave server is master server for a second line slave server.

39.3.3. zone transfers

The slave server receives a copy of the zone database using a **zone transfer**. Zone transfers are requested by the slave servers at regular intervals. Those intervals are defined in the **SOA record**.

The SOA record contains a **refresh** value. If this is set to 30 minutes, then the slave server will request a copy of the zone file every 30 minutes. There is also a **retry** value. The retry value is used when the master server did not reply to the last zone transfer request. The value for **expiry time** says how long the slave server will answer to queries, without receiving a zone update.

Zone transfers only occur when the zone database was updated (meaning when one or more resource records were added, removed or changed on the master server). The slave server will compare the **serial number** of its own copy of the SOA record with the serial number of its master's SOA record. When both serial numbers are the same, then no update is needed (because no records were added, removed or deleted). When the slave has a lower serial number than its master, then a zone transfer is requested.

39.3.4. full or incremental zone transfers

When a zone tranfer occurs, this can be either a full zone transfer or an incremental zone transfer. The decision depends on the size of the transfer that is needed to completely update the zone on the slave server. An incremental zone transfer is prefered when the total size of changes is smaller than the size of the zone database. Full zone transfers use the **axfr** protocol, incremental zone transfer use the **ixfr** protocol.

39.3.5. DNS cache

DNS is a caching protocol. When a client queries its local DNS server, and the local DNS server is not authoritative for the query, then this server will go looking for an authoritative name server in the DNS tree. The local name server will first query a root server, then a TLD server and then a domain server. When the local name server resolves the query, then it will relay this information to the client that submitted the query, and it will also keep a copy of these queries in its cache. So when a(nother) client submits the same query to this name server, then it will retrieve this information form its cache.

For example, a client queries for the A record on www.linux-training.be to its local server. This is the first query ever received by this local server. The local server checks that it is not authoritative for the linux-training.be domain, nor for the .be TLD, and it is also not a root server. So the local server will use the root hints to send an **iterative** query to a root server. The root server will reply with a reference to the server that is authoritative for the .be domain (root DNS servers do not resolve fqdn's, and root servers do not respond to recursive queries). The local server will then sent an iterative query to the authoritative server for the .be TLD. This server will respond with a reference to the name server that is authoritative for the linux-training.be domain. The local server will then sent the query for www.linux-training.be to the authoritative server (or one of its slave servers) for the linux-training.be domain. When the local server receives the ip-address for www.linux-training.be, then it will provide this information to the client that submitted this query. Besides caching the A record for www.linux-training.be, the local server will also cache the NS and A record for the linux-training.be name server and the .be name server.

39.3.6. caching only server

A DNS server that is set up without its own zone, but that is connected to other name servers and caches the queries is called a **caching only name server**.

39.3.7. iterative or recursive query

A **recursive query** is a DNS query where the client that is submitting the query expects a complete answer. An **iterative query** is a DNS query where the client does not expect a complete answer. Iterative queries usually take place between name servers. The root name servers do not respond to recursive queries.

39.4. old stuff....work in progress

Forward lookup zones are most common, they contain host or A records to translate hostnames or Fully Qualified Domain Names (FQDN) to ip addresses. Reverse lookup zones contain PTR records, they translate ip addresses to hostnames or FQDN's.

The internet contains thirteen logical DNS servers for the top of the hierarchy. This top is called the root, and is represented with a dot. Below the root are the Top Level Domains (TLD's). There are common TLD's like .com, .net. .info. aero. .museum, .gov, .mil, .edu and others. And there are country TLD's, like .be for Belgium and .fr for France.

The internet root name servers will only answer iterative queries, most local DNS servers will answer to recursive queries.

39.5. bind

One of the most common name servers on Linux is the Berkeley Internet Name Domain (bind) server. Use rpm or dpkg to verify whether it is installed.

```
[root@RHEL4b etc]# rpm -qa | grep -i bind
ypbind-1.17.2-8
bind-chroot-9.2.4-16.EL4
bind-utils-9.2.4-16.EL4
bind-devel-9.2.4-16.EL4
bind-libs-9.2.4-16.EL4
bind-9.2.4-16.EL4
```

39.6. named

The software is called 'bind', the daemon runs as 'named' ! So look for the named daemon, the named manual pages and /etc/named.conf to work with bind.

```
[root@RHEL4b etc]# apropos named | grep -i domain
named                (8)  - Internet domain name server
```

39.7. Caching only Name Server

A caching only name server is a DNS server that is not authoritative for any zone. It forwards queries to other DNS servers and locally caches the results.

The default /etc/named.conf on RHEL is a caching only name server.

39.8. Our first zone

The way to set up zones in /etc/named.conf is to create a zone entry with a reference to another file located in /var/named.

Here is an example of such an entry in /etc/named.conf

```
zone "classdemo.local" IN {
 type master;
 file "classdemo.local.zone";
 allow-update { none; };
};
```

To create the zone file, the easy method is to copy an existing zone file (this is easier than writing from scratch).

```
[root@RHEL4b named]# cd /var/named/
[root@RHEL4b named]# pwd
/var/named
[root@RHEL4b named]# cp localhost.zone classdemo.local.zone
[root@RHEL4b named]#
```

Here is an example of a zone file.

```
[root@RHEL4b named]# cat classdemo.local.zone
$TTL    86400
$ORIGIN classdemo.local.
@       IN SOA  rhel4b.classdemo.local.   admin.classdemo.local. (
                         2007083100      ; serial
                         3H              ; refresh
                         900             ; retry
                         1W              ; expiry
                         1D )            ; minimum

                IN NS           rhel4b.classdemo.local.
                IN MX     10    mail.classdemo.local.
                IN A            192.168.1.191

rhel4b          IN      A       192.168.1.191
mail            IN      A       192.168.1.191
www             IN      A       192.168.1.191
ftp             IN      A       192.168.1.191
server2         IN      A       192.168.1.1
```

39.9. Starting the name server

When starting the name server, don't forget to look at the log file to verify that all your zones are properly configured.

```
[root@RHEL4b etc]# service named restart
Stopping named:                                            [  OK  ]
Starting named:                                            [  OK  ]
[root@RHEL4b etc]# service named status
number of zones: 9
debug level: 0
xfers running: 0
xfers deferred: 0
soa queries in progress: 0
query logging is OFF
server is up and running
[root@RHEL4b etc]#
```

39.10. Practice DNS

1. Set up a working DNS server with your own zone. Test that it works.

2. Set up a master and a slave server.

Appendix A. Keyboard settings

A.1. About Keyboard Layout

Many people (like US-Americans) prefer the default US-qwerty keyboard layout. So when you are not from the USA and want a local keyboard layout on your system, then the best practice is to select this keyboard at installation time. Then the keyboard layout will always be correct. Also, whenever you use ssh to remotely manage a linux system, your local keyboard layout will be used, independent of the server keyboard configuration. So you will not find much information on changing keyboard layout on the fly on linux, because not many people need it. Below are some tips to help you.

A.2. X Keyboard Layout

This is the relevant portion in /etc/X11/xorg.conf, first for Belgian azerty, then for US-qwerty.

```
[paul@RHEL5 ~]$ grep -i xkb /etc/X11/xorg.conf
        Option      "XkbModel" "pc105"
        Option      "XkbLayout" "be"
```

```
[paul@RHEL5 ~]$ grep -i xkb /etc/X11/xorg.conf
        Option      "XkbModel" "pc105"
        Option      "XkbLayout" "us"
```

When in Gnome or KDE or any other graphical environment, look in the graphical menu in preferences, there will be a keyboard section to choose your layout. Use the graphical menu instead of editing xorg.conf.

A.3. Shell Keyboard Layout

When in bash, take a look in the /etc/sysconfig/keyboard file. Below a sample US-qwerty configuration, followed by a Belgian azerty configuration.

```
[paul@RHEL5 ~]$ cat /etc/sysconfig/keyboard
KEYBOARDTYPE="pc"
KEYTABLE="us"
```

```
[paul@RHEL5 ~]$ cat /etc/sysconfig/keyboard
KEYBOARDTYPE="pc"
KEYTABLE="be-latin1"
```

The keymaps themselves can be found in /usr/share/keymaps or /lib/kbd/keymaps.

```
[paul@RHEL5 ~]$ ls -l /lib/kbd/keymaps/
total 52
drwxr-xr-x 2 root root 4096 Apr  1 00:14 amiga
drwxr-xr-x 2 root root 4096 Apr  1 00:14 atari
drwxr-xr-x 8 root root 4096 Apr  1 00:14 i386
drwxr-xr-x 2 root root 4096 Apr  1 00:14 include
drwxr-xr-x 4 root root 4096 Apr  1 00:14 mac
lrwxrwxrwx 1 root root    3 Apr  1 00:14 ppc -> mac
drwxr-xr-x 2 root root 4096 Apr  1 00:14 sun
```

Index

Symbols

;, 56
!, 71, 73
!!, 73
? (file globbing), 70
/, 32
/bin, 33
/bin/bash, 51
/bin/csh, 51
/bin/dash, 51
/bin/ksh, 51
/bin/sh, 51
/boot, 33
/boot/grub, 33
/boot/grub/grub.conf, 33
/boot/grub/menu.lst, 33
/dev, 34, 44
/dev/hdX, 154
/dev/ht, 263
/dev/nst, 263
/dev/null, 34
/dev/pts/1, 34
/dev/random, 48
/dev/sdX, 154
/dev/st, 263
/dev/tty1, 34
/dev/urandom, 48, 49
/dev/zero, 48
/etc, 34
/etc/apache, 317
/etc/apache2/apache2.conf, 317
/etc/at.allow, 243
/etc/at.deny, 243
/etc/bashrc, 136
/etc/cron.allow, 243
/etc/cron.deny, 243
/etc/debian-version, 4
/etc/default/useradd, 83
/etc/ethers, 254
/etc/exports, 240
/etc/filesystems, 35, 41
/etc/fstab, 164, 167, 240, 253, 277
/etc/gentoo-release, 5
/etc/group, 97, 101
/etc/gshadow, 99
/etc/hosts, 49
/etc/httpd, 317
/etc/httpd/conf/httpd.conf, 317
/etc/inetd.conf, 233
/etc/init.d, 200
/etc/init.d/rcS, 198
/etc/init.d/samba, 286
/etc/init.d/smb, 286
/etc/inittab, 197
/etc/login.defs, 87
/etc/lsb-release, 5
/etc/lvm/.cache, 179
/etc/mandriva-release, 5
/etc/modprobe.conf, 213, 227
/etc/modprobe.d/, 213
/etc/mtab, 44, 166
/etc/network/interfaces, 223
/etc/nsswitch.conf, 304
/etc/passwd, 82, 101, 308
/etc/profile, 136
/etc/protocols, 219
/etc/raidtab, 173
/etc/rc.d/init.d, 200
/etc/rc.d/rc.sysinit, 198
/etc/rc3.d, 199
/etc/redhat-release, 4
/etc/resolv.conf, 49
/etc/samba/passdb.tdb, 307
/etc/samba/smb.conf, 283
/etc/services, 219, 233
/etc/shadow, 85
/etc/shells, 90, 116
/etc/skel, 35, 89
/etc/slackware-version, 5
/etc/squid/squid.conf, 335
/etc/ssh, 235
/etc/ssh/ssh_config, 234
/etc/ssh/sshd_config, 234
/etc/sudoers, 93
/etc/SuSE-release, 5
/etc/sysconfig, 35
/etc/sysconfig/, 221
/etc/sysconfig/firstboot, 36
/etc/sysconfig/harddisks, 36
/etc/sysconfig/hwconf, 36
/etc/sysconfig/iptables, 230
/etc/sysconfig/keyboard, 36
/etc/sysconfig/network, 221

/etc/sysconfig/networking, 220
/etc/sysconfig/network-scripts, 222
/etc/sysctl.conf, 323
/etc/syslog.conf, 248, 250
/etc/X11/xorg.conf, 35
/etc/xinetd.conf, 232
/etc/xinetd.d, 232
/home, 36
/lib, 37
/lib/kbd/keymaps/, 36
/lib/modules, 37, 210, 215
/lib/modules/<kernel-version>/modules.dep, 212
/lib32, 37
/lib64, 37
/media, 38
/mnt, 38
/opt, 38
/proc, 38
/proc/cmdline, 40
/proc/cpuinfo, 39
/proc/devices, 44
/proc/filesystems, 35, 41
/proc/interrupts, 41
/proc/kallsyms, 209
/proc/kcore, 42
/proc/mdstat, 42, 174
/proc/meminfo, 43, 252
/proc/modules, 43, 210
/proc/mounts, 44, 166
/proc/partitions, 44
/proc/scsi/scsi, 156
/proc/swaps, 45, 252
/proc/sys, 40
/proc/sys/net/ipv4/ip_forward, 322
/root, 45
/root/anaconda-ks.cfg, 255
/sbin, 45
/sbin/init, 197
/sbin/telinit, 201
/srv, 45
/sys, 45
/tmp, 46
/usr, 46
/usr/sbin/system-config-kickstart, 255
/usr/src, 205
/var, 46
/var/cache, 46
/var/lib/nfs/etab, 240
/var/lib/rpm, 47
/var/log, 46, 245
/var/log/auth.log, 248
/var/log/btmp, 245, 247
/var/log/lastlog, 245
/var/log/messages, 218, 245
/var/log/sa, 275
/var/log/secure, 247
/var/log/squid, 336
/var/log/wtmp, 245
/var/run, 46
/var/run/utmp, 245
/var/spool, 46
/var/spool/up2date, 47
/var/tmp, 46
., 112
./configure, 261
.., 112
.bash_login, 136
.bash_logout, 136
.bashprofile, 136
.bashrc, 136
.htaccess, 320
.htpasswd, 319
.profile, 136
.vimrc, 80
`(backtick), 68
~/.ssh/authorized_keys, 237
~/.ssh/id_rsa, 236
~/.ssh/id_rsa.pub, 236
'(single quote), 68
[, 70
$? (shell variables), 61
$ (shell variables), 60
$$, 127
$HISTFILE, 72
$HISTFILESIZE, 73
$HISTSIZE, 72
$LANG, 71
$PATH, 51, 136
$PPID, 127
* (file globbing), 70
\, 58
&, 57
&&, 57
#!/bin/bash, 116
>, 137

>>, 138
>|, 138
|, 141
1>, 138
2>, 138
2>&1, 138
777, 104

A
A (DNS record), 339
AAAA (DNS record), 339
absolute and relative paths, 14
access time, 153
AIX, 8
alias(shell), 52
alien(1), 261
apache2, 317
apropos, 11
aptitude, 8
aptitude(1), 7, 206
arp, 219
arp(1), 224
at(1), 242
ATA, 153
atapi, 153
atq(1), 242
atrm(1), 243
axfr, 341

B
backticks, 68
badblocks(1), 154
bash, 2
Belenix, 8
bg(1), 132
block device, 153
bonding (network cards), 227
bootp, 222, 254
Bourne again shell, 51
bridge, 219
browser master, 306
BSD, 2
BSD Net/2, 2
bzip2(1), 264

C
cable select, 153
cahing only name server, 341
Canonical, 6, 197
case sensitive, 32
cat(1), 28
cd, 13
cd -, 14
cd .., 14
cd ~, 13
CentOS, 6
chage(1), 88
chain(iptables), 327
chainloader, 195
chattr(1), 267
chgrp(1), 101
chkconfig, 36, 201
chmod(1), 103
chmod +x, 105, 115
chown(1), 101
chsh(1), 90
CIDR, 219
classful, 219
CMDBA, 10
CMDEV, 9
CNAME (DNS record), 340
comm(1), 148
command line scan, 53
command mode (vi), 78
compress(1), 149
cp(1), 22, 23
cpio(1), 268
crontab(1), 243
crontab(5), 243
crypt, 86
Ctrl-Alt-Delete, 199
Ctrl D, 28
Ctrl-Z, 131
current directory, 13
cut(1), 143
cylinder, 153

D
daemon, 127, 197
dd(1), 160, 253, 268
debian, 6
Debian, 8
default gateway, 224
Dennis Ritchie, 1
depmod(1), 212
devfs, 46
df(1), 166, 167, 272

dhclient(1), 223
dhcp, 222
dhcpd.conf, 255
diff(1), 148
directory, 111, 162
disk platters, 153
distributions, 4
dmesg(1), 156
DNAT, 322
DNS, 338
domain(DNS), 339
domain name system (DNS), 338
Douglas McIlroy, 1
du(1), 167, 272
dump(1), 267
dumpkeys(1), 36

E

e2fsck(1), 165
echo(1), 53, 67
echo $-, 72
edquota(1), 277
egrep, 198
ELF, 38
elilo, 194
el torito, 163
emacs, 78
embedding(shell), 67
env(1), 62, 62
environment variable, 60
EOF, 139
Eric Allman, 248
ethereal, 234
ethtool(1), 225
exec, 127
executables, 33
export, 63
exportfs(1), 240
ext2, 162, 163
ext3, 162
extended partition, 158

F

fat16, 162
fat32, 162
fd (partition type), 172
fdisk(1), 155, 158, 159, 160, 171, 272
Fedora, 5, 5, 5
fg(1), 132
FHS, 33
file(1), 20, 37
file globbing, 70
file ownership, 101
file system, 161
Filesystem Hierarchy Standard, 33
find(1), 108, 112, 147
firewall, 321
FireWire, 45
flame wars, 78
fork, 127
for loop (bash), 117
FQDN, 219, 339
free(1), 43, 252, 272
FreeBSD, 2, 7
freedom of speech, 3
fsck(1), 164
ftp://ftp.kernel.org, 204
fully qualified domain name, 339

G

gateway, 224
gcc, 2
getent(1), 304
GID, 97
glob(7), 70
gnome-session, 279
GNU/Hurd, 2
GNU/Solaris, 8
GNU Project, 2
gpasswd, 98
GPL, 3
grep, 43, 198
grep(1), 142
grep -i, 143
grep -v, 143
groupadd(1), 97
groupdel(1), 98
groupmod(1), 98
groups(1), 98
grpquota, 277
grub, 194, 194
grub.conf, 194
grub-install, 196
gzip(1), 264

H
hard link, 112
hdparm(1), 156
head(1), 27
head (hard disk device), 153
hidden files, 15
history, 73
host, 219
host (DNS record), 339
host id, 219
hostname, 219
HP-UX, 8
htpasswd(1), 319
http://en.wikipedia.org/wiki/, 12
http://www.kernel.org, 204
http://www.tldp.org, 12
http://xkcd.com, 6, 92, 151
httpd, 316
hub, 219

I
icmp, 219
id(1), 84
IEEE 1394, 45
ifcfg(1), 226
ifcfg-eth0, 222
ifconfig(1), 220, 228
ifdown(1), 223, 227
if then else (bash), 119
iftop(1), 275
ifup(1), 223, 227
igmp, 219
inetd, 231
init, 127, 196
initng, 197
initrd, 195, 208
inode, 111, 112
insert mode (vi), 78
insmod(1), 211, 211
iostat(1), 274
ip-address, 219
iptables, 229, 327
iptables save, 329
iso9660, 162, 268
iterative query, 341
ixfr, 341

J
JBOD, 170
jobs, 131
joe, 78
joliet, 162
journaling, 162
Jumpstart, 254

K
Ken Thompson, 1
Kerberos, 240
keymaps(5), 36
kickstart, 254, 255
kill(1), 127, 130, 200
kill -9, 130
killall, 130
kmyfirewall, 230
Korn Shell, 90
ks.cfg, 255
kudzu, 36

L
last(1), 246
lastb(1), 247
lastlog(1), 246
less(1), 29
let, 120
lilo, 194, 194, 196
lilo.conf, 196
Linus Torvalds, 3
ln(1), 112
loadkeys(1), 36
locate(1), 147
logger(1), 250
logical AND, 57
logical drive, 158
logical OR, 58
Logiciel Libre, 4
login, 246
logrotate(1), 251
LPIC 1 Certification, 9
LPIC 2 Certification, 9
lpstat(1), 312
ls(1), 15, 111, 111
ls -l, 16
lsmod, 43
lsmod(1), 210
lspci(1), 272

lsscsi(1), 157
lvcreate(1), 184, 187, 188
lvdisplay(1), 180, 189
lvextend(1), 186, 189
LVM, 176
lvmdiskscan(1), 177
lvol0, 185
lvremove(1), 185
lvrename(1), 186
lvs(1), 180
lvscan(1), 180

M
MAC, 219
MacOSX, 8
magic(5), 21
make, 217
make(1), 261
make bzImage, 214
make clean, 214
make menuconfig, 214
make modules, 215
make mrproper, 213
make xconfig, 214
man, 10
man(1), 282
man hier, 32
man -k, 11
man man, 11
Mark Shuttleworth, 6
master (hard disk device), 153
master boot record, 160
master server (DNS), 340
mbr, 160
MBR, 269
mdadm(1), 173
menu.lst, 194
mirror, 170
mkdir, 17
mkdir -p, 17
mke2fs(1), 162, 163, 188
mkfile(1), 253
mkfs(1), 162, 163
mkinitrd(1), 162, 216
mknod(1), 263
mkswap(1), 252
modinfo, 217
modinfo(1), 211
modprobe(1), 211, 212, 227
more(1), 29
mount, 35, 44
mount(1), 165, 166, 240
mounting, 165
mount point, 166
mpstat(1), 274
mt(1), 264
mv(1), 24
MX (DNS record), 340

N
nano, 78
NAPT, 322
NAS, 238
NAT, 321
NCP, 238
NetBSD, 2, 7
net groupmap, 310
net rpc join, 303
netstat(1), 224
net use, 292, 312
net view, 284, 284
network id, 219
Nexenta, 8
NFS, 238, 239
nfs, 254
nice(1), 129
nmbd, 285
noclobber, 138
nodev, 35, 41
nounset(shell), 61
Novell, 6
Novell Certified Linux Professional, 10
NS (DNS record), 339
ntop(1), 275

O
od(1), 149
OpenBSD, 2, 7, 234
OpenSSH, 8, 234
openssl(1), 86
OpenSUSE, 7

P
package manager, 4
packet filtering, 321
packet forwarding, 321
paging, 252

Parallel ATA, 153
parent directory, 14
parted(1), 159
partition, 157
partprobe(1), 161
passwd(1), 85, 85, 86
PAT, 322
pgrep(1), 131
pico, 78
PID, 127
ping, 219
pipe, 141
pkill(1), 131
popd, 20
port forwarding, 322
portmap, 239
poweroff, 201
PPID, 127
primary group, 83
primary partition, 158
primary server (DNS), 340
process, 127
process ID, 127
proxy server, 334
ps, 128
ps(1), 272
ps -ef, 129
ps fax, 129
PTR (DNS record), 339
pushd, 20
pvchange(1), 182
pvcreate(1), 181, 187, 188
pvdisplay(1), 178, 188
pvmove(1), 182
pvremove(1), 181
pvresize(1), 182
pvs(1), 178
pvscan(1), 178
pwd, 13

Q
quota.group, 277
quota.user, 277
quota's, 277
quota(1), 277
quotacheck(1), 277
quotaoff(1), 277
quotaon(1), 277

R
RAID, 170
RAID 1, 170
RAID 5, 170
random number generator, 48
rarp, 254
read, 120
recursive query, 341
Red Hat, Inc., 5
Red Hat Desktop, 5
Red Hat Enterprise Linux, 5, 5
Red Hat Network, 47
Red Hat Update Agent, 47
reiserfs, 163
Remote Desktop, 278
rename(1), 24
renice(1), 129
repeater, 219
repository, 4
repquota(1), 277
resize2fs(1), 189
respawn(init), 199
restore(1), 267
RHCE, 9
RHEL AS, 5
RHEL ES, 5
RHEL WS, 5
Richard Stallman, 2
rlogin, 234
rm(1), 22, 113
rmdir, 17
rmmod(1), 212
rm -rf, 22
roaming profiles(samba), 309
rock ridge, 162
root(DNS), 338, 338
root directory, 32
rootsquash, 240
rotational latency, 153
route(1), 224, 224
router, 219
RPC, 239
rpcinfo(1), 239
rpm, 5, 257
RPM, 47
rsh, 234
runlevel, 197
runlevel(1), 201

S
sa2(1), 275
sadc(1), 275
sal, 275
samba, 282
sample.ks, 255
SAN, 238
sar(1), 275, 275
SATA, 153
scp(1), 236
SCSI, 153
scsi_info(1), 157
SCSI ID, 153
secondary server (DNS), 340
sector, 153
seek time, 153
segment, 219
service(1), 200, 225, 230, 286
set, 72
set(shell), 62
set +x, 54
SetGID, 108
setgid, 108
set -x, 54
sfdisk(1), 161
shell comment, 58
shell escaping, 58
shell expansion, 54, 66
shopt, 122
shutdown, 201
silo, 194
skeleton, 35
slave (hard disk device), 153
slave server (DNS), 340
slice, 157
SMB, 238
smbclient(1), 286, 295
smbd, 283, 285, 295
smbpasswd(1), 295, 310
smbtree(1), 287
SMF, 197
SNAT, 322
SOA (DNS record), 340
soft link, 112
Solaris, 8
sort(1), 145
split(1), 270
squid, 334, 335
ssh, 234, 237
ssh-keygen(1), 235
ssh -X, 237
stateful firewall, 321
stderr, 137
stdin, 137, 141
stdout, 137, 141
sticky bit, 108
strings(1), 30
striped disk, 170
su -, 137
su(1), 91, 91
subnet, 219
subnet mask, 221
sudo(1), 93
sudo su -, 93
Sun, 197
SunOS, 8
Sun Solaris, 8
Suse, 6
swapoff(1), 252
swapon(1), 252
swap partition, 163
swap partition(s), 45
swapping, 252
swap space, 252
switch, 219
symbolic link, 112
sysfs, 45
syslogd, 248
System.map, 209
system-config-network, 222
system-config-network-cmd, 225
system-config-securitylevel, 230
System V, 196

T
tac(1), 29
tail(1), 27, 251
tar(1), 261, 265, 266
tcp, 219
tdbsam, 306, 307
tee(1), 142
telinit(1), 201
telnet, 234
test, 118
testparm(1), 284, 285
tftp, 254

Theo De Raadt, 7
time(1), 214
TLD, 338
top(1), 129, 252, 272
top level domain, 338
touch(1), 21
tr(1), 144
track, 153
Tru64, 8
tune2fs(1), 162, 164, 168
type(shell), 51

U
Ubuntu, 4, 6, 8
Ubuntu, Linux for Human Beings, 6
udf, 163
udp, 219
umask(1), 105
unalias(shell), 53
uname(1), 204
Unbreakable Linux, 6
uncompress(1), 149
UNICS, 1
uniq(1), 146
Universally Unique Identifier, 168
unset, 72
unset(shell), 62
until loop (bash), 117
updatedb(1), 147
update-rc.d, 201
upstart, 197
usb, 45
useradd, 83
useradd(1), 86, 89
useradd -D, 83
userdel(1), 83
usermod, 97
usermod(1), 83, 88, 89
usrquota, 277
UUID, 168

V
vfat, 162
vgchange(1), 184
vgcreate(1), 183, 187, 188
vgdisplay(1), 179
vgextend(1), 183
vgmerge(1), 184
vgreduce(1), 183
vgremove(1), 183
vgs(1), 179
vgscan(1), 179
vi, 78
vigr(1), 99
vim, 78
vimtutor, 78
vipw(1), 89
virtual memory, 252
visudo(1), 92
vmlinuz, 208
vmstat(1), 273
vnc, 278
vncviewer(1), 278
vol_id(1), 168
vrije software, 4

W
w(1), 84
watch(1), 251, 273
wbinfo(1), 304
wc(1), 145
webalizer, 320
while loop (bash), 117
who(1), 84, 246
who am i, 84
whoami(1), 84
who -r, 201
wild cards, 71
winbind, 285, 303, 304
WINS, 296

X
X, 35
Xen, 6
xinetd, 232
xstartup(vnc), 279
X Window System, 35

Y
yaboot, 194
yum, 259

Z
z/IPL, 194
zfs, 163
zombie, 127
zone transfer (DNS), 340