Linux system administration — Students' notes

Anton Altaparmakov Darran Bryant David Carter Bob Dowling
Ben Harris Julian King Stella Page

December 16, 2003

Contents

0 Introductory remarks **21**

0.1 This course . . . 21

0.2 Linux . . . 23

1 Installing Red Hat Linux **31**

1.1 Concepts . . . 32

1.2 Before you install . . . 38

1.3 An installation walk through . . . 41

2 XFree86 **61**

2.1 Introduction . . . 61

2.1.1 Graphics cards . . . 61

2.1.2 Framebuffers and VESA mode . . . 61

2.1.3 Some DDC magic . . . 61

2.1.4 Power Management . . . 62

2.2 Configuring XFree86 . . . 62

2.2.1 Device . . . 67

2.2.2 Mice . . . 68

2.2.3 Keyboards . . . 71

2.2.4 Files . . . 72

2.2.5 Monitor . . . 73

2.2.6 Screen . . . 75

2.3 ServerLayout . . . 77

2.4 Commercial X servers . . . 78

2.5 Window Managers . . . 78

2.6 Play time 80

2.7 External Documentation 82

2.8 Graphical Logins 82

3 The Red Hat Software Package System 83

3.1 Software management 83

3.2 Querying packages 85

3.3 Installing packages 91

3.4 Dependencies 94

3.5 Removing packages 96

3.6 Overriding dependency checks 97

3.7 Upgrading packages 98

3.8 Checking a package 100

3.9 Postinstall scripts 103

3.10 Miscellaneous notes 104

3.11 Building software from source code 104

4 Boot sequence 109

4.1 Introduction 109

4.2 BIOS – the Basic Input/Output System 109

4.3 GNU GRUB – The Grand Unified Bootloader 111

4.4 Kernel startup 115

4.5 `init` 116

4.6 Changing runlevels 122

5 Network concepts 125

6 The kernel **149**

6.1 `/proc`: The window on the kernel . . . 149

6.2 Kernel upgrades . . . 151

7 Service configuration **155**

7.1 Continually running services . . . 155

7.2 Network services run from `xinetd` . . . 158

7.3 Services run from `crond` . . . 161

7.4 Log files . . . 164

7.5 A review of common services . . . 165

8 Managing users and groups **171**

9 Practical filesystems **183**

9.1 Introduction . . . 183

9.2 Hardware . . . 183

9.3 Filetypes . . . 184

9.4 What is in a name? . . . 185

9.5 Partitioning disks . . . 185

9.6 `fdisk` . . . 185

9.7 Things you need to know to use `fdisk` . . . 186

9.8 Using `fdisk` . . . 186

9.9 Creating a partition . . . 188

9.10 Filesystems . . . 189

9.11 Creating a filesystem . . . 190

9.12 Mounting filesystems . . . 190

9.13 `/etc/fstab` . . . 192

9.14 `tunefs` 192
9.15 Backups 193
9.16 Filesystems concepts 194
9.16.1 Mode 195
9.16.2 Timestamps 195
9.17 `fsck` 196
9.18 Advanced Filesystems 197

10 Security **199**

10.1 Defences 200
10.2 Access controls 202
10.3 Password security 205
10.4 Network sniffing 206
10.5 SSH – the Secure Shell 207
10.6 X11 security 210
10.7 Threat models – know your enemy 212
10.8 Dealing with attacks 214

11 The Network File System **217**

11.1 Introduction 217
11.2 NFS experiments on a single system 219
11.3 NFS server configuration 220
11.4 NFS client configuration 222
11.5 The Automounter 225

Slides

1 Welcome! . . . 21

2 Introductions . . . 21

3 Style . . . 21

4 Prerequisites . . . 22

5 Background . . . 23

6 Checking your workstation . . . 24

7 Agenda . . . 31

8 Hardware issues . . . 32

9 Disk Structure and Partitioning . . . 33

10 Disk partitioning examples . . . 33

11 Disk Partitioning utilities . . . 34

12 Understanding Device Names . . . 35

13 The Unix Filesystem and Mount points . . . 35

14 Mount Points and Disk Partitions . . . 36

15 A simple partitioning scheme . . . 37

16 Installing GRUB/LILO . . . 37

17 So, what do you need to know? . . . 38

18 Networking . . . 39

19 Installation images . . . 40

20 Making Boot Disks . . . 40

21 The Installation Program User Interface . . . 41

22 Exercise: Checking DDC . . . 62

23 How to Configure XFree86 . . . 63

24 Exercise: Fetching the Matrox drivers . . . 64

25 Exercise: Installing Matrox drivers 64

26 Exercise: redhat-config-xfree86 65

27 Exercise: Testing XF86Config configuration 66

28 XF86Config 67

29 Default Graphics card config 67

30 Device directives 68

31 Default Mouse Configuration 69

32 Mouse types under X 70

33 Multiple Mice 70

34 Default keyboard setup 71

35 Exercise: Setting up the keyboard 72

36 Default “Files” section 72

37 Setting the font path 73

38 Default Monitor section 74

39 Exercise: Updating the monitor configuration 74

40 Monitor section directives 75

41 Default Screen section 75

42 Display Subsection 76

43 Exercise: Setting up the Screen section 77

44 Default SeverLayout 78

45 Exercise: Creating a new user 79

46 Before 79

47 Exercise: Gnome/Metacity config files 79

48 After 80

49 Exercise: The Gnome desktop environment 81

50 Exercise: The Gnome desktop environment (optional) 81
51 Problems with software management 83
52 Concepts of package management 84
53 Manipulating packages 84
54 What packages are installed? 85
55 What XFree86 packages are installed? 85
56 Enquiring of individual packages 86
57 Detailed information on a package 86
58 Listing the files in a package 87
59 Determining the package for a file 88
60 Exercise: Packages before they are installed 88
61 Architecture tags 88
62 Exercise: Querying a package file 89
63 Exercise: Querying a package file for information 90
64 Exercise: Querying a package file for a file listing 90
65 Loading Red Hat's signature file 90
66 Installing a package 91
67 Failing to install `tetex-xdvi-1.0.7-66` 92
68 Looking for the `libwww` libraries 92
69 Finding the `libwww` libraries 93
70 The RPM database package 93
71 Failing to install `tetex-xdvi-1.0.7-66` again 93
72 Successful installation of `tetex-xdvi-1.0.7-66` 94
73 Concepts of dependencies 94
74 Possible requirements 95

75 Using `rpm --query --requires` 95

76 Using `rpm --query --provides` 96

77 Exercise: Resolving dependencies 96

78 Erasing an installed package 97

79 Failing to erase a package 97

80 Successfully erasing packages 97

81 Erasing and then replacing a depended-on package 98

82 What is upgrading? 98

83 A simple upgrade—CUPS 99

84 Downgrading 99

85 Failing to downgrade 99

86 Forcing a downgrade with `--oldpackage` 100

87 Freshening—upgrading only if necessary 100

88 Freshening `cups` 100

89 What can go wrong with a package? 100

90 Why can a package go wrong? 101

91 Verifying a package 101

92 Verification codes 102

93 Verifying a file's package 103

94 Verifying everything 103

95 Not just a collection of files 103

96 Postinstall script example 104

97 Miscellaneous notes 104

98 Why build from source code? 104

99 Bits of a source tree 105

100 Exercise: Building nano—GNU pico 105
101 Exercise: Building gtypist 106
102 System startup overview 109
103 BIOS responsibilities 109
104 BIOS configuration 110
105 BIOS limits 110
106 GNU GRUB 111
107 Typical GRUB boot process 111
108 grub.conf, global settings 113
109 grub.conf, menu entries 113
110 Exercise: Modifying grub.conf 114
111 GRUB menu and prompt 114
112 Useful kernel parameters 114
113 Exercise: Kernel parameters 115
114 Kernel messages 115
115 Exercise: Looking at the boot messages 115
116 Initial RAM disc 116
117 Exercise: Seeing the initial RAM disk 116
118 init(8) 116
119 Runlevels 117
120 /etc/inittab 117
121 /etc/rc.d/rc.sysinit 118
122 rc*n*.d scripts 119
123 Interactive startup 120
124 Exercise: Interactive startup 120

125 respawn entries 121

126 Exercise: Changing the number of text consoles 121

127 telinit and runlevel 122

128 Exercise: Changing runlevel 122

129 shutdown(8) 123

130 halt(8), reboot(8), poweroff(8) 123

131 Exercise: shutdown and friends 124

132 Internet network connections 125

133 Setting up a network connection—1 126

134 Setting up a network connection—2 126

135 Setting up a network connection—3 127

136 Setting up a network connection—4 127

137 Extracts from /etc/services 129

138 Using rpcinfo 129

139 Example ifconfig output 130

140 Example netstat output (some columns dropped) 130

141 A TCP packet over ethernet (simplified) 132

142 The address resolution protocol 132

143 Examining the ARP cache 133

144 A system's routing table (some columns dropped) 133

145 Examples of masking 134

146 Routing to 131.111.8.42 134

147 Examples of ifconfig and route 135

148 A typical /etc/sysconfig/network 135

149 A typical /etc/sysconfig/network-scripts/ifcfg-eth0 136

150 The DNS 136
151 A records 136
152 An A record 136
153 Multiple A records 137
154 Looking up a FQDN 137
155 Format of `/etc/hosts` 138
156 Handling incomplete names 138
157 Handling incomplete names: `pcttr101` 139
158 Handling incomplete names: `pcttr101.titan1.pwf` 139
159 Some other common record types in the DNS 140
160 CNAME Records 140
161 MX Records 140
162 Structure of DNS 141
163 Domain structure 141
164 Zone structure 141
165 PTR records 142
166 PTR records — Structure of key 142
167 NS Records 142
168 Chronology of a lookup 143
169 Top level records 143
170 Stage two—the `.com` zone 144
171 Stage three—the `.redhat.com` zone 144
172 Stage four—the answer 145
173 `dig`'s output—The top 146
174 `dig`'s output—The answer section 146

175 dig's output—The authority section . . . 146

176 dig's output—The additional information . . . 147

177 dig's output—The tail . . . 147

178 Exercise: Looking up a PTR record . . . 147

179 Looking in /proc . . . 149

180 Confusing ls . . . 149

181 Exercise: Playing with /proc . . . 150

182 Exercise: Using /proc/*PID* . . . 151

183 Files relevant to Red Hat's kernel . . . 151

184 Packages for Red Hat's kernel . . . 151

185 Upgrading Red Hat's kernel—1 . . . 152

186 Upgrading Red Hat's kernel—2 . . . 152

187 Upgrading Red Hat's kernel—3 . . . 152

188 Booting off the new kernel . . . 153

189 Checking /boot/grub/grub.conf . . . 153

190 Editing /boot/grub/grub.conf . . . 153

191 Exercises . . . 154

192 Types of service . . . 155

193 Examples of continually running services . . . 155

194 The comments at the top of /etc/init.d/gpm . . . 156

195 Listing all services . . . 156

196 Listing just one service . . . 157

197 Changing when the gpm service is launched . . . 157

198 Turning off a service in the current runlevel . . . 158

199 The service program: Syntactic sugar for start-up scripts . . . 158

200 Examples of on-demand services . . . 158
201 The file `/etc/xinetd.conf` . . . 159
202 The file `/etc/xinetd.d/rsync` . . . 159
203 `xinetd` keywords . . . 159
204 Three ways to send a HUP signal to the running `xinetd` . . . 161
205 Disabling an `xinetd` service . . . 161
206 Examples of periodically run services . . . 161
207 The `crond` control directories . . . 162
208 `/etc/crontab` . . . 162
209 Format of `/etc/crontab` job lines . . . 162
210 Numeric constructions in crontab files . . . 163
211 The `/etc/cron.daily/logrotate` file . . . 163
212 Exercise: Writing cron jobs . . . 164
213 Log rotation . . . 164
214 A rotation instruction in `/etc/logrotate.conf` . . . 164
215 The rotation of `/var/log/cron` . . . 165
216 Exercise: Analysing your services . . . 165
217 Agenda . . . 171
218 Background—So what are users? . . . 171
219 The Password file . . . 172
220 The Shadow file . . . 172
221 The password aging mechanism . . . 173
222 Setting the password aging values . . . 173
223 Groups, file ownerships and the primary group . . . 174
224 Managing users . . . 175

225 useradd 175

226 userdel 176

227 usermod 177

228 UPG - User private group scheme 178

229 User private group scheme 178

230 UPG - Sharing directories 178

231 Sharing directories - Why it works 180

232 System wide configuration file 180

233 Exercise 182

234 Additional Exercise 182

235 Filetypes 184

236 Disk partitioning tools 185

237 IDE hard drive devices 186

238 Generic Hard drive devices 186

239 Partition devices 186

240 Working out which device to use. 186

241 Example output from fdisk -l 187

242 Options on fdisk 188

243 fdisk options explained 188

244 Creating a new partition example 189

245 Exercise: Create a new partition 189

246 Exercise: Create a filesystem 190

247 Interesting mount options 191

248 Exercise: Mount the filesystem 191

249 /etc/fstab explained 192

250 Exercise: Set and change the volume label . . . 193

251 Exercise: Dumping and moving `/var` . . . 194

252 Inode pointer and data layout . . . 195

253 Some magic for `fsck` . . . 196

254 Exercise: Using `fsck` to fix a filesystem . . . 196

255 Some more magic for `fsck` . . . 197

256 So you want to run a secure Linux system, do you? . . . 199

257 Components of security . . . 199

258 Components of insecurity . . . 199

259 Keeping informed . . . 200

260 Limiting services . . . 200

261 Keeping up-to-date . . . 201

262 Friendly probing . . . 201

263 Exercise: Looking at probing output . . . 201

264 Access controls . . . 202

265 `tcp_wrappers` . . . 202

266 `hosts.allow` and `hosts.deny` . . . 202

267 Exercise: `hosts.allow` and `hosts.deny` . . . 203

268 Service-specific controls . . . 203

269 CUDN access controls . . . 204

270 Password encryption . . . 205

271 User folly . . . 205

272 Network sniffing . . . 206

273 Preventing sniffing . . . 206

274 SSH – the Secure Shell . . . 207

275 SSH client usage 208

276 Advanced(ish) SSH usage 209

277 SSH server operation 209

278 X11 security 210

279 Host-based security 210

280 Authentication tokens 211

281 X forwarding over SSH 211

282 Script kiddies 212

283 Chronology of an attack 213

284 Detecting a break-in 214

285 So, you've been hacked? 214

286 Determining the extent of the damage 215

287 Re-installation 215

288 Cleaning up 215

289 Introduction to NFS 217

290 NFS daemons 218

291 Exercise: Simple NFS experiment: Part I 219

292 Exercise: Simple NFS experiment: Part II 219

293 Access control for NFS servers: `/etc/exports` 220

294 Exercise: Exporting a directory to other systems 222

295 NFS client side configuration: `mount`(8) and `/etc/fstab` 222

296 Common mount options for NFS: see `nfs`(5) 223

297 Some useful diagnostic tools for NFS clients 224

298 Exercise: Mounting a directory from a remote system 225

299 Automounters 225

300 Checking `autofs` is installed 225

301 Automounter configuration 226

302 Exercise: Using autofs 226

0 Introductory remarks

0.1 This course

- Find a spare computer
- Make sure you can read this
- Please use the same machine all week

Slide 1: Welcome!

The course and the course givers

The people giving this course are, essentially, the inhabitants of Balfour 3, a largish office in the Computing Service. We are the ones with the most in depth knowledge within the Computing Service of the various Unix operating systems and of Linux in particular.

The course will be very intensively taught. There will be almost continual hands-on practical work and there will be a number of us wandering about to help you during the practical stages. During the practical sets, please feel free to discuss the matter with your neighbours. The practical work should not be done in the atmosphere of a monastery, but in a large collaborative free-for-all. We do request that you set pagers, mobile phones and other things that may distract to a silent mode of operation. Failing that, please switch them off.

- Anton Altaparmakov
- Darran Bryant
- Bob Dowling
- Ben Harris
- Julian King
- *Yourselves!*

Slide 2: Introductions

- Lecturing
- Practical work
- Cooperative
- Demonstrators
- Please don't rush ahead
- Pagers, mobile phones, ...

Slide 3: Style

- You can use the Unix command line

  `bash`

- You can use a Unix editor

  `vi`, `emacs`, `pico`

Slide 4: Prerequisites

Prerequisites

This course does have a few prerequisites. It is an introduction to Linux *system administration*, not to Linux *use*. We do assume that you have used the Linux command line before, running the `bash` shell (which is the Linux default shell) and that you can use one of the common editors, `vi`, `emacs` or `pico`. (Strictly speaking, `pico` is *not* a "common editor", but it is common within Cambridge and is routinely installed on Red Hat Linux systems so we're providing it as part of the installations here.)

Course content

This course splits into a number of sections:

Introduction: This bit.

Prerequisites: Some things we really want you to know.

Installation: The installation of Red Hat Linux on a system and the underlying concepts you need to understand to answer the questions correctly.

The Windowing system: The X server and its clients and libraries. How to configure it and how to drive the hardware.

Software packages: How to manipulate software on a Red Hat Linux box.

Boot sequence: How it boots, how the various services get started and why it goes wrong.

Network concepts: What's really going on in networking.

The kernel: The heart of the system and how to upgrade it.

Service configuration: The various services and how to switch them on or off.

Users & groups: How to create users and groups of users. How to use the groups to get permissions right. How to get users the environment they need.

File systems: How they work, how to create them, how they fail and how to fix them when they do.

Security: Why bother securing your system? How to do it once you've decided it is important. (Hint: it is.)

Network file system: Sharing files over the network.

Network services: Web servers, mail servers, time servers, server servers.

- Linux - the kernel
- GNU - the programs
  FSF, GPL
- Distributions
  Red Hat, Debian, SuSE, Slackware, ...
- Supported hardware lists

Slide 5: Background

0.2 Linux

Background

Open Source: Simply the idea that the source code should be made available to the customer. An early example was the BSD Unix licence which allowed people access to the source and allowed them to adapt it and sell on their adaptation, not necessarily with the source code.

GNU: The Free Software Foundation worked on the GNU (GNU's Not Unix) project to provide large amounts of open source software. They also launched the GNU General Public License, or Copyleft, which stopped the users of the code from redistributing it without the source code and without attaching the GPL to their copy or derivative. This is the most open of the open source licences.

Linux: Linus Torvalds released a Unix kernel to replace Minix on 17th September, 1991 under the terms of the GNU GPL. He later declared his plans for Linux to be "World Domination". Granted an honorary doctorate by the University of Helsinki in 1999. Strictly speaking, Linux is just the base kernel, the core of the system. The majority of the tools and commands that run over it come from the GNU stable and are also covered by the GPL. For this reason, some of the FSF insist that what is typically called "Linux" should really be called "GNU/Linux".

Distributions: Any operating system is more than just a kernel and some programs and libraries. It also needs a means to install it and means to manage it. There is also a great deal of flexibility regarding the location of these programs which needs to be decided in a consistent manner. Solving these problems is typically done by producing a collection of the parts designed for certain locations and with a set of tools to assist in the installation. This is, basically, what we call a "distribution". The earliest real distribution was "Slackware", which is now showing its age. The other common distributions today are Red Hat Linux which we are using and which is also common in the U.S.A., S.u.S.E. Linux which is more common on mainland Europe, TurboLinux which has its strength in the Pacific Rim and Debian Linux which has worldwide appeal and which most connection with the original volunteer ethos underlying Linux.

Supported hardware lists: Various parts of Linux need to communicate with the hardware. Because some vendors release their hardware expecting Microsoft Windows to be run on it and nothing else they tend not to release the information needed for Linux to communicate with them. As a result there is often a lag between new pieces of hardware being released and Linux being able to drive them. While this is less of a problem than it used to be, "supported hardware lists" are still maintained listing the hardware that Linux can drive.

- Screen
- Keyboard
- Mouse
- Posture

Slide 6: Checking your workstation

The GNU General Public License

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software–to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

GNU GENERAL PUBLIC LICENSE
TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

<one line to give the program's name and a brief idea of what it does.> Copyright (C) <year> <name of author>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

Gnomovision version 69, Copyright (C) year name of author Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type 'show w'. This is free software, and you are welcome to redistribute it under certain conditions; type 'show c' for details.

The hypothetical commands 'show w' and 'show c' should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than 'show w' and 'show c'; they could even be mouse-clicks or menu items–whatever suits your program.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the program, if necessary. Here is a sample; alter the names:

Yoyodyne, Inc., hereby disclaims all copyright interest in the program 'Gnomovision' (which makes passes at compilers) written by James Hacker.

<signature of Ty Coon>, 1 April 1989 Ty Coon, President of Vice

This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Library General Public License instead of this License.

1 Installing Red Hat Linux

- Concepts:
  - Hardware issues, Disks, Partitioning, Mount Points, GRUB/LILO.
  - Greater detail will be given in later sections.
- Installation Prerequisites:
  - What you need to know about your system before you install Red Hat Linux.
  - Networking issues.
- The Installation Interface:
  - Virtual consoles.
- An installation walk through.
- Your installation.

Slide 7: Agenda

Intro

This section outlines certain concepts that you will be required to understand to install Red Hat Linux. The concepts covered are not explained in great detail. The aim is to give you enough information to get you through the installation. Most of the topics discussed are explained in far more depth in the following sections. Red Hat Linux provides many ways to perform the installation such as booting from CD-ROM, booting from local disk, FTP, HTTP and NFS. During this course we are going to use the NFS method to install the operating system. By the end of this section you will have Red Hat Linux installed on your system.

You will also be familiar with:

- Hardware Issues
- Partitions and mount points
- Packages
- GRUB/LILO
- Virtual consoles
- Different installation classes
- The installation program's user interface

- Supported Architectures
- Device drivers
  - latest and greatest may not be supported.
  - developers do not have a fixed relationship with the hardware vendors.
- CPUs
  - The installation program automatically probes for the number of CPUs.
  - Two kernels installed:
  - `grub.conf` entries are `Red Hat Linux (kernel version)` and `Red Hat Linux (kernel version-smp)`
  - `lilo.conf` entries are `linux` and `linux-up`
- Memory
  - Red Hat 9 on 32-bit x86 can autodetect up to a gigabyte of memory.
  - kernel hacks are needed to detect more.

Slide 8: Hardware issues

1.1 Concepts

General Issues

There are a number of hardware issues, which need to be addressed before you can install Red Hat.

Architectures: Linux can run on a variety of hardware platforms for example x86, Itanium and Compaq Alpha. This course will concentrate on the x86 platform. However a similar process can be used to install Linux on other architectures.

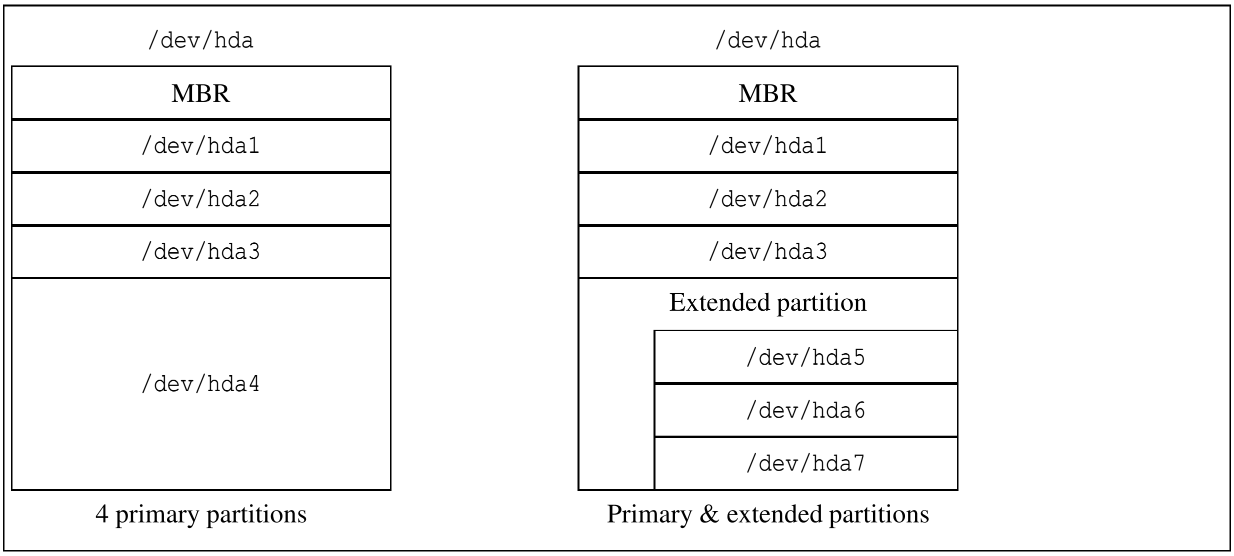
Device drivers: Unfortunately the latest and greatest hardware is sometimes not supported by Linux. This is due to the software development community not having a fixed relationship with the hardware vendors. The Red Hat hardware compatibility list can be found at `http://hardware.redhat.com/hcl` and is worth checking if you wish to buy a new machine to run Red Hat on. A list of supported graphics cards for Linux can be found at: `http://www.xfree86.org`.

CPUs: A maximum of sixteen CPUs are supported by the kernel.

Memory: The 2.4 kernel which is supplied with Red Hat Linux 9 can autodetect up to 1 gigabyte of memory on a 32-bit processor machine. If you want more memory to be recognised then you will need to perform some non-trivial kernel modifications.

- Basic Disk Structure:
  - Master Boot Record (MBR).
  - A number of paritions.
- Master Boot Record:
  - Partition table stores info on how the partitons are laid out.
  - Boot loader stores info on where to boot from.
- Partitions:
  - Up to 4 primary partitions.
  - Can convert a primary partition into an extended partition.
  - Extended partitions are containers for logical partitions.
  - Up to 12 logical partitions can exist within an extended partition.

Slide 9: Disk Structure and Partitioning



Slide 10: Disk partitioning examples

A brief description about disk structure

Partitioning provides a mechanism that allows discs to be spilt into slices. These smaller slices are easier to manage and also provide a certain amount of flexibility.

Each hard drive can be split up into a maximum of four primary partitions. If additional partitions are required then one of the four primary partitions can be divided further by making an extended partition. An extended partition is simply a container which holds logical partitions (see slide 10).

- Partitioning Tool:
  - Disk Druid (only available from within the installation program).
- Disk Druid:
  - Easy to use for standard partitioning schemes.
  - Makes intelligent assumptions about the use of extended and logical partitions.
  - You can assign mount points to partitions.
  - *BUT:*
  - it does impose restrictions.
  - you can't edit the existing partition set up.
  - it decides where partitions go on the disk and what device names are allocated.

Slide 11: Disk Partitioning utilities

Disk partitioning utilities

The partitioning tool used by the installation program is called Disk Druid and it is only available from within the Red Hat installation program.

It offers a graphical representation of your hard drive(s) and allows you to partition your hard drive(s) by using the mouse. It also allows you to assign mount points.

Disk Druid uses the following rules when assigning partitions to device names:

- The 4 primary partitions are assigned unique device names of `/dev/hda1`, `/dev/hda2`, `/dev/hda3` and `/dev/hda4` respectively.
- If one of the primary partitions is used for an extended partition then the logical partitions within this extended partition are assigned device names sequentially upwards from `/dev/hda5`. For example the first logical partition will always be assigned `/dev/hda5`, the second `/dev/hda6` and so on.

- Device names are allocated to each partition.
- Primary partition are assigned `/dev/`*`xxy`*`[1-4]`.
- Logical partitions are assigned sequentially upwards from `/dev/`*`xxy`*`5`.
- `/dev/`*`xxyN`*
  - *xx*: Indicates what type of device, `hd` (IDE), `sd` (SCSI)
  - *y*: Indicates which disk, `a` (the first disk), `b` (the second disk)
  - *N* : The actual partition number.
- `/dev/hda5`
  - the 1st logical partition on the 1st IDE drive.
- `/dev/sdb7`
  - the 3rd logical partition on the 2nd SCSI drive.

Slide 12: Understanding Device Names

Understanding Device Names

The device names allocated to the individual partitions are deciphered as follows:
`/dev/`*`xxyN`*

`dev`	This directory contains all the device files. All possible devices reside under `/dev/`.
xx	These two characters indicate what type of device the partition is on. For example `hd` indicates an IDE disk, `sd` would indicate a SCSI device.
y	This indicates which device the partition is on. For example `hda` would be the first IDE disk.
N	This denotes the partition. For example `/dev/sdb7` would be the third logical partition on the second SCSI drive.

- A quick Unix filesystem review:
  - built up in an inverted tree structure.
  - the top of the tree is called root or `/`.
- Mounting:
  - is a mechanism of attaching an extra branch to the tree structure.
  - it maps partitions onto reference points in the filesystems.
  - e.g. `mount /dev/hda8 /usr/local`
  - the filesystem on `/dev/hda8` can now be accessed through the `/usr/local` mount point.
- Mount points are assigned to partitions during the installation.
  - this tells the installation where to put things!

Slide 13: The Unix Filesystem and Mount points

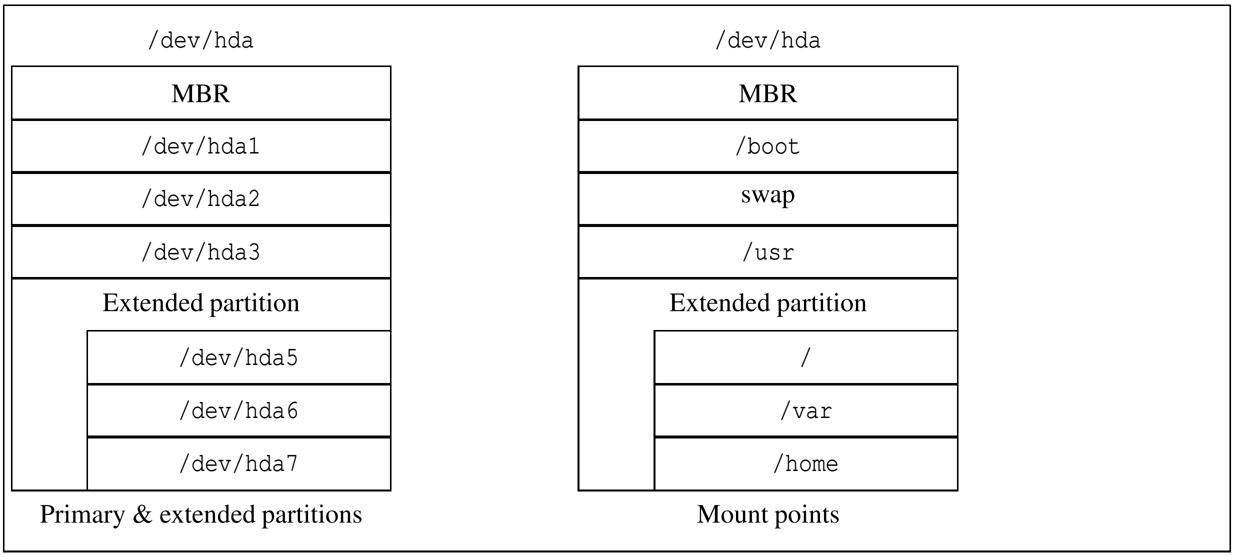
The Unix filesystem and Mount points

The Unix file system is built up in an inverted tree structure. At the top of the structure is a directory called root (`/`). Directories in general are just files that contain subdirectories and other files. The root directory contains several subdirectories which have names such as `/etc`, `/sbin`, `/lib`, `/proc` and so on. Each of these subdirectories contain files that are grouped together under a particular subdirectory because they provide a similar function. For example the `/dev` directory is reserved for file system entries that represent devices that are attached to the system. The `/lib` directory should contain only those libraries that are needed to execute the binaries in `/bin` and `/sbin`.

Here is a brief description of the subdirectories that can be found directly under the root directory:

`bin`	Essential command binaries.
`boot`	Static files of the boot loader.
`dev`	Device files.
`etc`	Host-specific system configuration.
`home`	User home directories.
`lib`	Essential shared libraries and kernel modules.
`mnt`	Mount point of temporary partitions.
`opt`	Add-on application software packages.
`root`	Home directory for the root user.
`sbin`	Essential system binaries.
`tmp`	Temporary files.
`usr`	Secondary hierarchy.
`var`	Variable data.

The complete standard can be found at `http://www.pathname.com/fhs/`



Slide 14: Mount Points and Disk Partitions

Partitions and Mount points

Subdirectories under the root filesystem do not have to live on the same physical disk partition. For example the root directory (`/`) could reside on `/dev/hda6` while `/home` could reside on `/dev/hda7`.

To attach `/home` to the main (`/`) filesystem we use a mechanism called "mounting". Mounting simply attaches an extra subdirectory to the existing file system under a special point in the filesystem called a mount point. These mount points are transparent and just look and act like any other subdirectories.

Mount points can be assigned to partitions during the installation by using Disk Druid or after the installation by adding entries to the `/etc/fstab` file. The `/etc/fstab` file is a lookup table that is read during the system start up. It simply maps mount points to device names.

Slide 14 shows the relationship between mount points and physical partitions.

- Partitioning is a very contentious issue amongst seasoned installers.
- You need at least 2 *partitions* for the installation:
  - A root partition or (/).
  - Swap (There is a maximum of 8 swap partitions).
- Red Hat also recommends a `/boot`.
  - it contains the kernel and a small number of files used during the bootstrapping process.
- *BUT:* It is best to spilt certain mount points onto separate partitions.
  - this provides some extra flexibility and resilience.
  - Could split `/usr`, `/home`, `/data`, `/var`.
  - Don't separate `/etc`, `/lib`, `/sbin`, `/dev`.

Slide 15: A simple partitioning scheme

Partitioning issues

Partitioning is a very contentious issue among seasoned Linux installers. The partitioning scheme that you use depends mainly on what the machine is going to be used for. For example if you have a large web server that generates a copious amount of log files then perhaps it would be a good idea to have a separate, larger than normal, `/var`.

The `/boot` partition contains all the files which are needed to bootstrap the system. The idea is to keep this partition located near the front of the disk therefore overcoming the 1024 cylinder limit imposed by some older makes of BIOS.

- GRUB (GRand Unified Boootloader) or LILO (The Linux Loader) – used to boot the system.
- The default is GRUB.
- By default it get installed in the MBR.
- You can change where GRUB is installed during the installation?
  - in the Master Boot Record (can boot both Linux and Win 95/98/2K/XP).
  - or in the first sector of your root partition (this doesn't overwrite the current system loader).

Slide 16: Installing GRUB/LILO

Installing GRUB/LILO

To allow the system to be booted without a floppy you are going to need a boot loader. The installation program provides you with two boot loaders for you to choose from.

GRUB is the GRand Unified Boootloader and is installed by default. LILO is the Linux Loader.

During the installation you can also choose where you wish to install GRUB. You can either install it in the MBR or you can install it on the first sector of your root partition. Installing GRUB on the first sector of the root partition is the recommended place if you wish to keep the system loader that is already installed in the MBR.

If you currently have Windows 98, Windows 95 or Windows 2000 installed on your system you can still install GRUB into the MBR. GRUB is quite friendly and will allow you to boot either operating system.

By default GRUB in installed in the MBR. If you are not sure whether or not installing GRUB will destroy your current set up then it is suggested that you take a copy of the MBR before the installation.

1.2 Before you install

Gathering Information

Installing Red Hat Linux is a relatively straightforward process. However in order for the installation to go as smoothly as possible there are a number of prerequisite tasks that you should perform.

- The things you should know about:
  - Hard Drive(s): How many, size, IDE or SCSI ?
  - Network card: Make and model number?
  - Memory: How much?
  - Mouse: Type, number of buttons, protocol?
  - Monitor: Make, model and manufacturer?
  - Graphics card: Make, model and VRAM?
- So, how do you find out?

  - look at the vendor supplied documentation, use Windows or open the box.
- These days the autoprobe facility does a good job at detecting hardware.

Slide 17: So, what do you need to know?

Hardware

In order to install Red Hat you will need to know certain information about the machine on which you are planning to put the operating system. On most newer systems the installation is able to probe your system to identify what hardware is installed.

Once you have found out what hardware configuration your machine has, write it down! This will save you time on the next occasion you update or reinstall your system.

Once you have collected the information concerning your system it is worth checking with the Red Hat compatibility list to see if your hardware will work with the operating system.

- Your network details can be obtained from your CO or <ip-register@ucs.cam.ac.uk>.
- Networking:

  Machine Name: the name you wish to call your machine.

  IP number: a unique number used to identify your machine.

  Netmask: a number that defines the scope of your local network.

  Default gateway: the gateway out of your subnet.

  Nameserver: the system that resolves machine names to IP address.

Slide 18: Networking

Networking

As part of networking your system you are going to need an IP address. This is usually assigned by <ip-register@ucs.cam> address or your computer officer.

- IP Address

  This is a series of four numbers between `0` and `254` separated by dots. It will be something like `131.111.8.2`. This is the number that is unique to your computer and corresponds to your machine's name.

- Netmask

  This is like an IP address but very likely the numbers will all be 0 or 255. It will usually have the value `255.255.255.0` though some (typically large) departments have the value `255.255.0.0`. The netmask is used to determine whether or not an outgoing packet is destined for a machine on the local subnet or not. If the packet in question is local then the packet is sent directly to the machine, otherwise the packet will be sent to the router.

- Default router

  This is also called the gateway. Common (but not universal) practice within the University is for the default router address to be the same as the machine's IP address except for the last of the four numbers being replaced by `62`. So in the example above the default router might be `131.111.8.62`.

- Nameserver

  IP register will also give you the IP addresses of your nearest the nameserver. Some departments have their own nameserver, in which case you will be told about it. Otherwise you will have been given the IP addresses of the two central nameservers that we run: `131.111.8.42` and `131.111.12.20`.

- bootdisk.img - primary boot image for all install methods.
- A secondary driver disk will be needed.
- The following driver disks are available:

  `drvnet.img` - Supplemental Network Drivers.

  `pcmciadd.img` - PCMCIA Driver Diskette.

  `drvblock.img` - Supplemental Block Device Drivers.
- So which one do you use?

  depends on your machine's configuration.

  and where the copy of the installation media resides.

Slide 19: Installation images

- Where do you get the boot disk from?

  - located in the `images` directory.
- You need:

  - a blank, formatted, high-density (1.44 MB), 3.5-inch disk.

  - a machine capable of writing to 3.5-inch disks.

  – For example to make a boot disk under a Linux-like OS from our NFS server:

```
# mount nfs-uxsup.csx.cam.ac.uk:/linux/redhat/9/en/os/i386/ /mnt
# cd /mnt/images
# dd if=bootdisk.img of=/dev/fd0
```

Slide 20: Making Boot Disks

First stage installer: The boot diskette

A boot diskette will be needed if you cannot boot from the CD-ROM or if you wish to install from a network, block, or PCMCIA device.

A If you choose to create a boot diskette then you must also create the appropriate driver diskette for your booting scenario. See slide 19 for details.

The diskette images can be found in the images directory on your Red Hat Linux CD-ROM.

The second stage installer

Once found and loaded, the second stage installer contains the majority of the installation.

- It is a graphical user interface (GUI).
  - you point and click to navigate screens or enter text fields.
  - you can also use the [Tab] and [Return] keys.
- Virtual Consoles:
  Console 1, [Ctrl]-[Alt]-[F1], the installation dialogue.
  Console 2, [Ctrl]-[Alt]-[F2], the shell prompt.
  Console 3, [Ctrl]-[Alt]-[F3], the installation log.
  Console 4, [Ctrl]-[Alt]-[F4], the system log, messages from the kernel etc.
  Console 5, [Ctrl]-[Alt]-[F5], other messages.
  Console 7, [Ctrl]-[Alt]-[F7], the X graphical display.
- Can be very helpful if you encounter a problem during the installation!

Slide 21: The Installation Program User Interface

Virtual Consoles

The Red Hat installation procedure provides a number of virtual consoles that can be used during the installation. These can be extremely helpful if you encounter problems while installing. Not only do they provide a number of diagnostic messages but also provide you with a mechanism, a shell, where you can enter commands.

There are six virtual consoles available and you can switch between them using a single keystroke.

Generally there is no need to switch between different consoles during the installation.

1.3 An installation walk through

Booting the Installation Program

This part of the course is designed to give you a step by step account of a typical dialogue that occurs when installing Red Hat Linux over the network from Unix Support's NFS server.

Since we are performing a network installation we are going to need two floppy disks.

The first will contain the initial installation program (`bootdisk.img`) and the second will store a copy of the network drivers (`drvnet.img`).

To make the boot floppies under Linux:

```
# mount nfs-uxsup.csx.cam.ac.uk:/linux/redhat/9/en/os/386/ /mnt
# cd /mnt/images/
# dd if=bootdisk.img of=/dev/fd0
(Insert a different floppy disk)
# dd if=drvnet.img of=/dev/fd0
```

To start the installation procedure off, insert the boot floppy (`bootdisk.img`) into your computer's first floppy disk drive and reboot. Your BIOS settings may need to be changed to allow you to boot from the disk. After a short delay a screen containing the boot prompt should appear. This screen contains a variety of boot options.

Further information on the different boot options available can be obtained by pressing the [F2] function key.

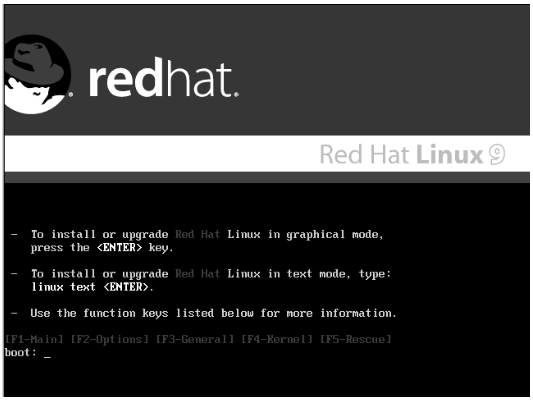


Figure 1: Boot Options

For this course: Simply press the [Return] key.

Choosing a language

The next screen is a simple question asking you which language you wish to use during the installation.

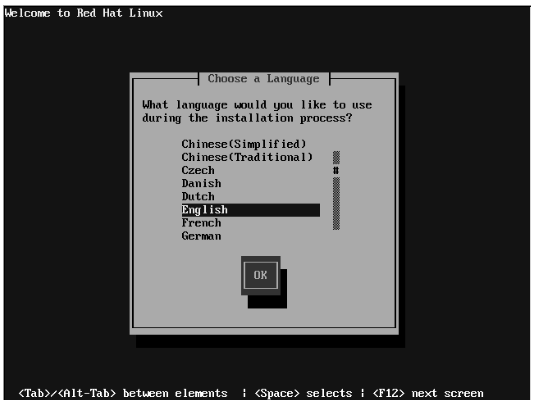


Figure 2: Choosing a language

For this course: Select *English*.

Selecting the keyboard

Next, the installation asks you what keyboard type you are using.

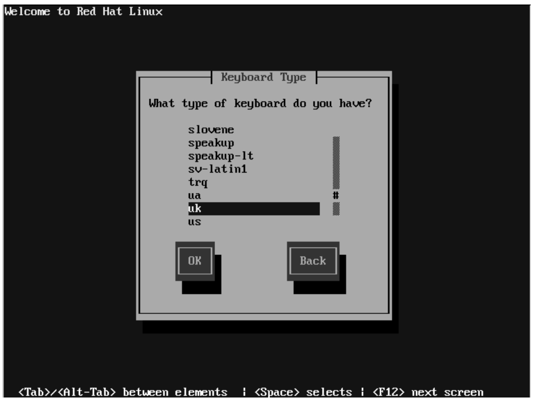


Figure 3: Selecting the Keyboard

For this course: Select *uk*.

Selecting an installation method

Next, you will be asked which type of installation method you are planning to use.

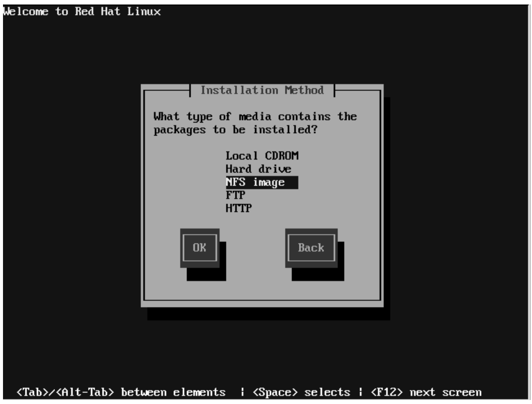


Figure 4: Selecting an Installation Method

For this course: We are installing Red Hat from an NFS server so select *NFS image*.

No driver found

Next, you will be asked if you would like to manually select your driver or to use a driver disk.

Figure 5: No driver found

For this course: Select *Use a driver disk*.

Driver Disk Source

Next, you will be asked which device contains the driver disk.

Figure 6: Selecting the driver disk source

For this course: Insert the network driver disk and select *fd0*.

Configuring the network

You will now be asked to enter the machine's network configuration. Most of this information can be found from your local C.O. or on the letter sent to you from the IP-register group.

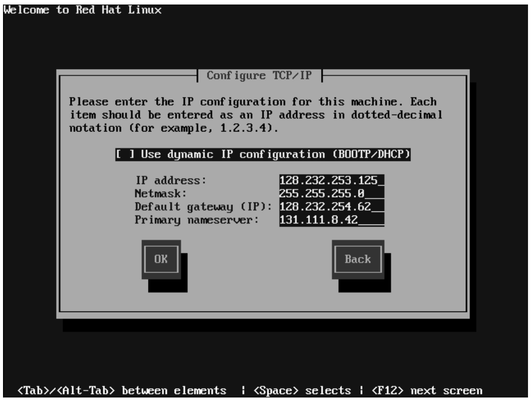


Figure 7: Configuring the network

For this course:	Do not select the dynamic IP configuration option.	
	IP address:	`128.232.253.`*`xx`*
	Netmask	`255.255.255.192`
	Default gateway	`128.232.253.62`
	Primary nameserver	`131.111.8.42`

NFS setup

The next screen asks you for information regarding your NFS server. You must enter the fully qualified domain name or the IP address of your NFS server. You must also enter the name of the exported directory that contains the Red Hat Linux CD.

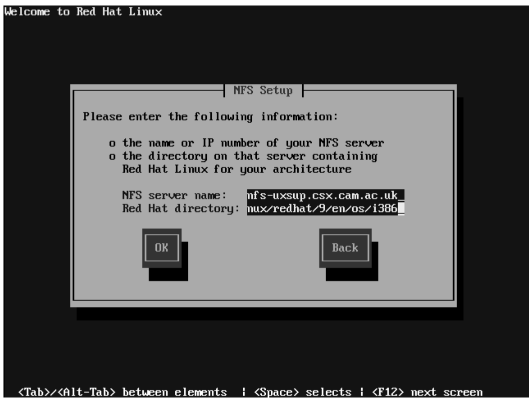


Figure 8: NFS set up

For this course:	NFS server name:	`nfs-uxsup.csx.cam.ac.uk`
	Red Hat directory:	`/linux/redhat/9/en/os/i386`

There will now be a short wait while the installation program is downloaded from the server.

Welcome to Red Hat

This is the GUI to the installation program. Throughout the installation you can use the mouse to select options and navigate between the screens. You can also select installation options by using the [Tab] and [Return] keys.

Cancelling the installation can occur any time before the *"About to install"* screen. There is no turning back after this screen because the installer will start writing packages to your local disk.

Figure 9: Welcome to Red Hat

For this course: Simply press [Return] to continue with the installation.

Hint: Be very careful to only use a single mouse click to select the *Next* button on the GUI. Double clicking the mouse button will result in the installation program skipping a screen. Sorry to sound pedantic but this is a very common mistake.

The graphical interface

Once the second stage of the installation program has been downloaded from the NFS server and the GUI started, you will be ask to configure your mouse.

If you don't know what mouse you have, choose one that is compatible with yours. Otherwise choose the *Generic* option. If you have a serial mouse you will need to choose the correct port and device that your serial mouse is connected to. The *Emulate 3 Buttons* option allows a 2 button mouse to emulate a 3 button mouse by clicking both mouse buttons at the same time to act as a middle button.

You can change the mouse configuration after the installation by using the `redhat-config-mouse` command in a shell prompt.

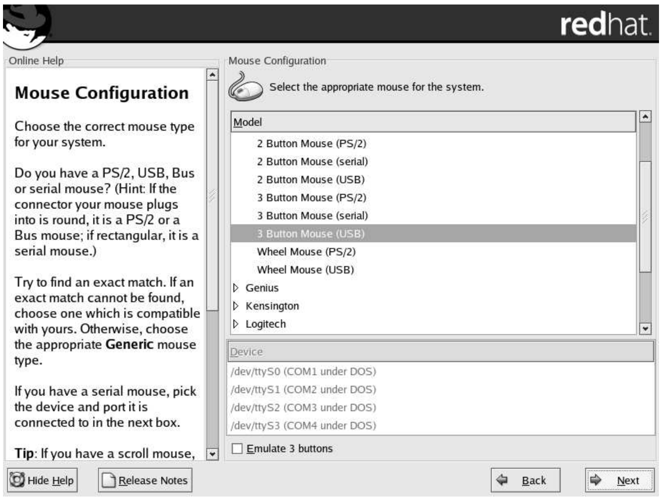


Figure 10: What mouse?

For this course: Accept the mouse that has been highlighted.

Installation type

On this screen you must decide what type of installation you require. Note that in figure 11 the “Custom” button is selected and all the others are unselected.

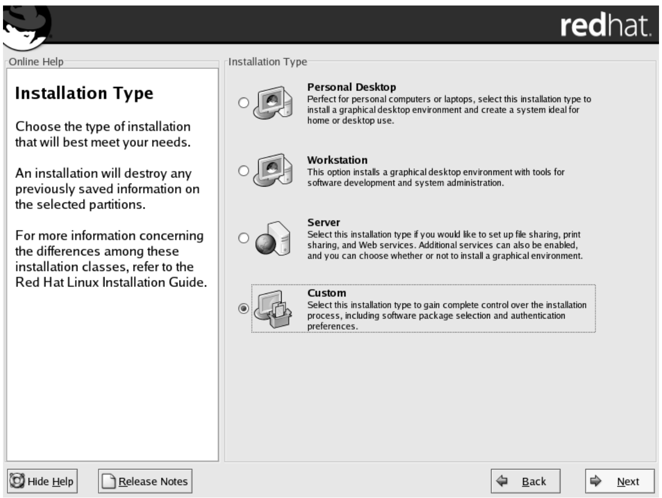


Figure 11: Installing Type

For this course: Choose the *Custom* installation option.

Further details about each installation class can be found in the Red Hat Installation Guide. The guide can be found at `www.redhat.com/docs/manuals/linux`.

Disk Partitioning

On this screen you have to select whether on not you wish to partition manually.

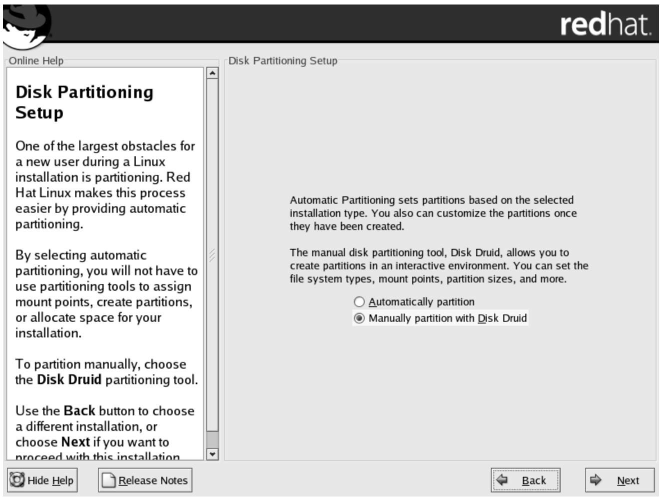


Figure 12: Disk Partitioning

For this course: Select the *Manually Partition with Disk Druid* option.

What's what in the main Disk Druid Window - Partitioning

Next, you must tell the installation program where to install the Red Hat software. This is achieved by defining mount points for one or more disk partitions into which the operating system is installed.

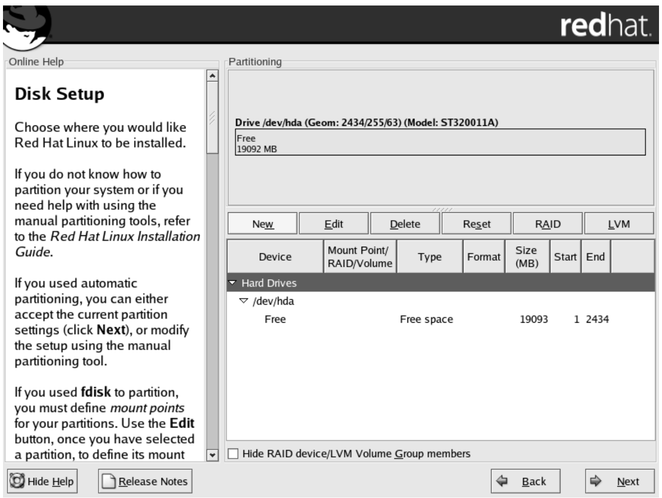


Figure 13: Partitioning

Controlling Disk Druid

You can control Disk Druid by clicking on the appropriate buttons with the mouse or by using keyboard short cuts.

Here is a description of each button:

New:	Used to add a new partition.
Edit:	Used to edit mount points and partition sizes.
Delete:	Used to delete the partition which is highlighted within the partition section table.
Reset:	This will reset the partition table to its original state. All changes will be lost.
RAID:	Used to provide redundancy to any or all disk partitions.
LVM:	Allows you to create an LVM logical volume.

An example of how to add a single partition

Here is an example of how to add a 100M partition called `/boot`.

First click on the *New* button in the main Disk Druid window. Then enter the appropriate details into the window that pops up. For this example enter `/boot` in the *Mount Point* field and 100 into the *Size (Megs)* field, then you simply need to click on the *Ok* button. See figure 14 for an illustration of this example.

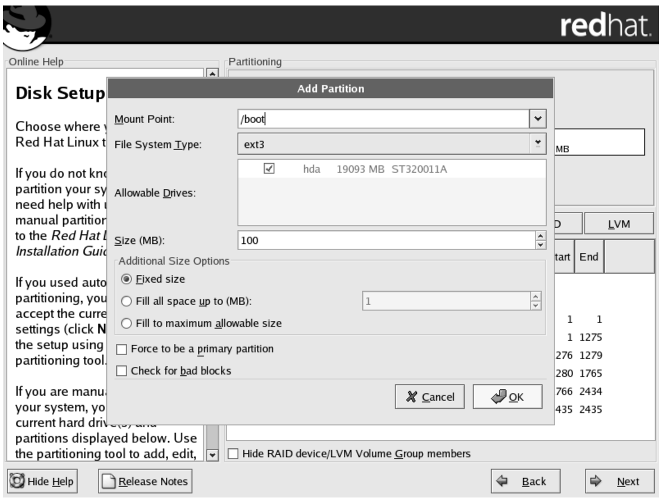


Figure 14: Partitioning

You can choose whether to keep a partition a fixed size or to allow it to ”grow” (fill up the rest of the disk) or to allow the partition to grow to a certain point.

If you select the *Fill to maximum allowable size* button on more than one partition then any additional free space will be shared out between the partitions.

The partition layout for this course

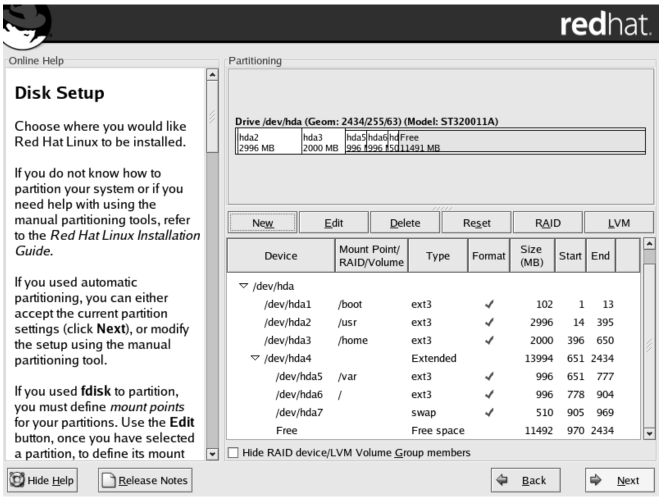


Figure 15: Partitioning

For this course: First delete all of the partitions shown.

Next, create the following partition layout:

100 MB	`/boot`
512 MB	swap
1000 MB	`/`
4000 MB	`/usr`
1000 MB	`/var`
2000 MB	`/home`

Make sure that none of the partitions has the *Fill to maximum allowable size* button selected. This deliberately leaves some space spare for us to play with during this course.

Hint: If you make a complete hash of the partitioning simply hit the *Reset* button. This restores the partition table to its original state.

Installing GRUB/LILO

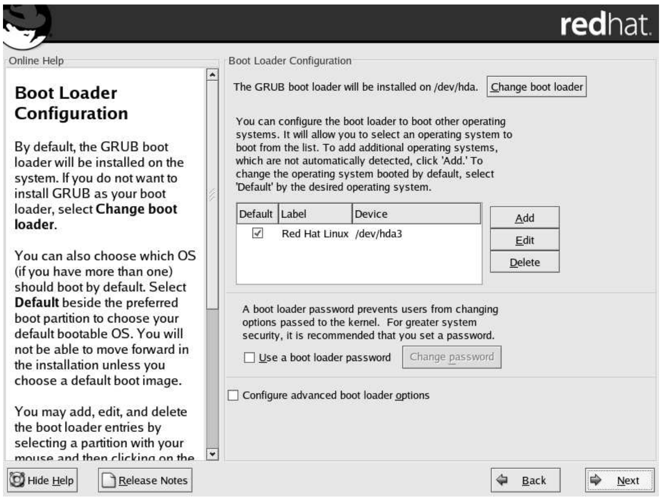


Figure 16: Installing LILO

For this course:

Do not change the default boot label.
Do not use a boot loader passwd.
Do not select the *Configure advanced boot loader options* button.

Network Configuration

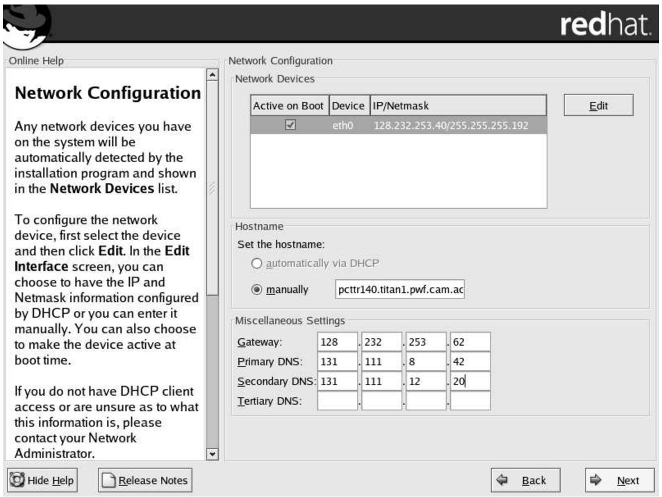


Figure 17: Network Configuration

For this course:
Do not select the *automatically via DHCP* button.
Add a secondary DNS server: `131.111.12.20`
Check the network configuration and click on the *Next* button.

Firewall Configuration

On this screen of the installation process we have the option of setting up a firewall to stop selected network services from being accessed.

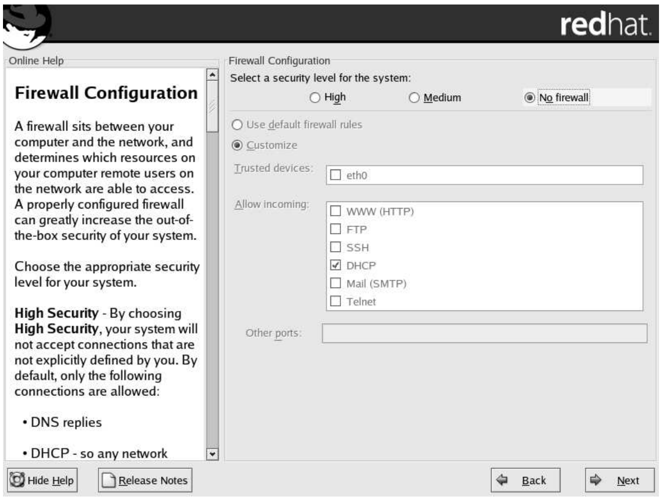


Figure 18: Firewall configuration

For this course: Select the *No firewall* option.

Language support selection

This is where we decide on the default language that is used once the installation has finished.

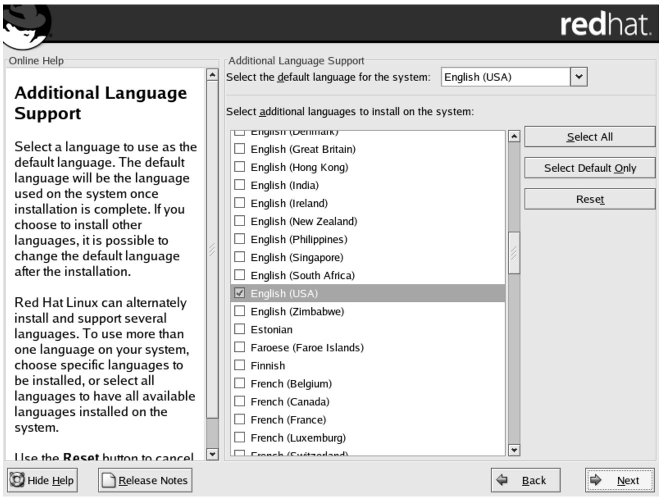


Figure 19: Language support selection

For this course: Accept the default language *English (USA)*

Time zone configuration

You can set your time zone by either selecting your computer's physical location or by an offset from the Universal Co-ordinated Time.

There are a number of ways you can select the timezone. You can either use the interactive map and click on the yellow dot which corresponds to your city or you can scroll through the list and choose your desired timezone.

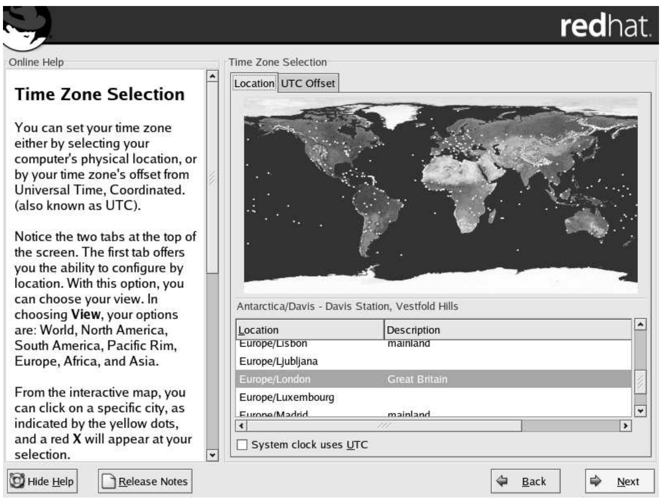


Figure 20: Time zone configuration

For this course: Set the Timezone to Europe/London.

Warning: If you are setting up a dual boot Windows/Linux system then it is probably best to set the system clock to use local time and not UTC. On the whole is it not a very good idea to have two different operating systems fighting over which one controls the system clock.

Setting the `root` password

On this screen you simply have to supply the password for the root account.

The root password you choose must be at least six characters long and is case sensitive. Make sure you choose a password which is easily remembered and has a good mix of numerals, alphanumeric characters, upper and lower case letters, and is not a dictionary word.

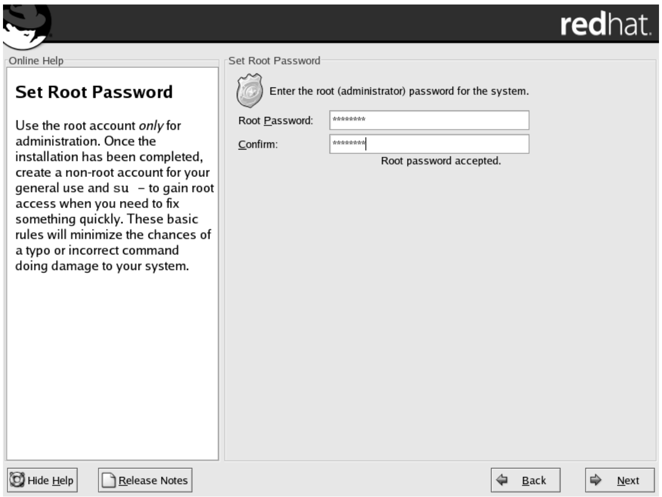


Figure 21: Root password

For this course: Set the root password to be the one shown on the overhead projector.

Authentication

MD5 is simply a different encryption algorithm from the old-style one and allows up to 256 characters instead of the normal eight letters or fewer. Shadow passwords are a mechanism which allows the encrypted passwords to be readable only by `root`.

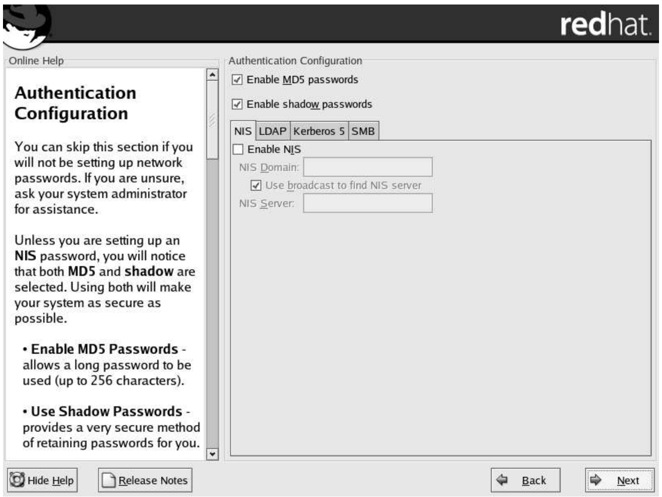


Figure 22: Authentication

For this course: Make sure that MD5 and shadow passwords are enabled and all the other methods are disabled.

Package group selection

This is the part of the installation where you get to choose which package groups of the Red Hat distribution you wish to install. Each package group is a bundle of packages. For example the kernel development package group contains various packages to install, compile and build new kernels. Once a package group has been selected it is then possible to select or deselect individual packages from that group by clicking on the *Select individual packages* button and advancing to the next screen.

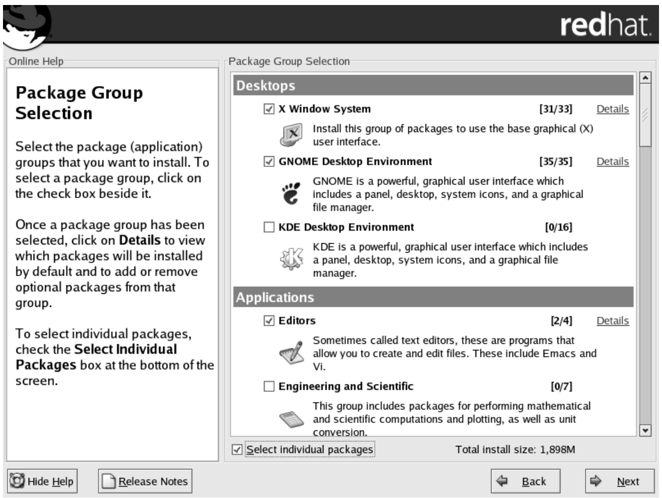


Figure 23: Package group selection

For this course:

Select the following package groups:	X Window System GNOME Desktop Enivronment Editors Graphical Internet Text-based Internet Office/Productivity Sound and Video Graphics Development Tools Kernel Development Administration Tools
Remove the following packages groups:	Printing Support

Make sure that the *Select individual packages* button is selected.

Selecting individual packages

On this screen you can add or delete individual packages from the package groups that you selected on the previous screen.

You can choose to view the individual packages in *Tree View* or *Flat View*.

The *Tree View* displays the packages grouped by application type.

The *Flat View* display the packages in an alphabetical listing on the right of the screen.

Using *Tree View*, you see a listing of package groups. When you expand this list (by double-clicking on the folder arrow beside a package group name) and pick one group, the list of packages in that group appears in the panel on the right. *Flat View* allows you to see all of the packages in an alphabetical listing on the right of the screen.

To select an individual package, click the checkbox beside the package name. A check mark in the box means that a package has been selected.

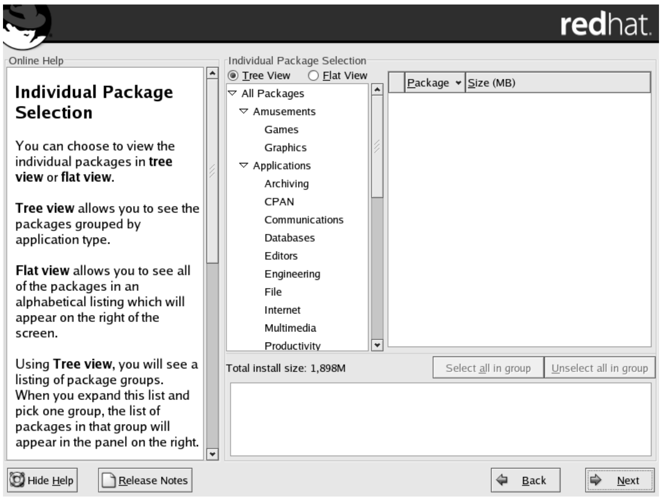


Figure 24: Selecting individual packages

For this course:

- Add the `anaconda` package:

  Hint - take a look in the *Applications* menu then in the *System* folder.

- Remove the `tcl` package:

  Hint - look in the *Development* menu then in the *Languages* folder.

Unresolved Dependencies

Some packages depend on other packages being installed, so to make sure that the system has all the packages it needs, it performs a dependency check every time you install or remove an individual package.

If there are any unresolved dependencies then the installer will list them and give you an opportunity to install the packages needed.

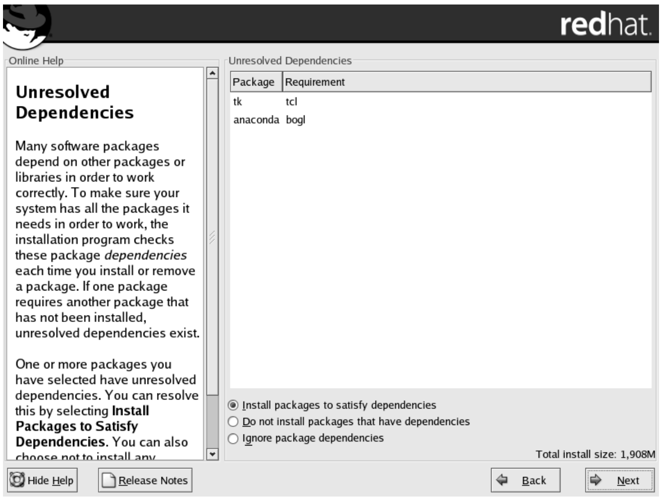


Figure 25: Unresolved Dependencies

For this course:

- Check that *Install packages to satify dependencies* is selected.
- Click on the *Next* button to satisify the dependencies.

About to Install

You should now see a screen preparing you for the installation of Red Hat Linux. If you would rather not continue with the installation process, this is your last opportunity to safely cancel the process and reboot your machine. To cancel this installation process, press your computer's Reset button or use the [Control]-[Alt]-[Delete] key combination to reboot your machine.

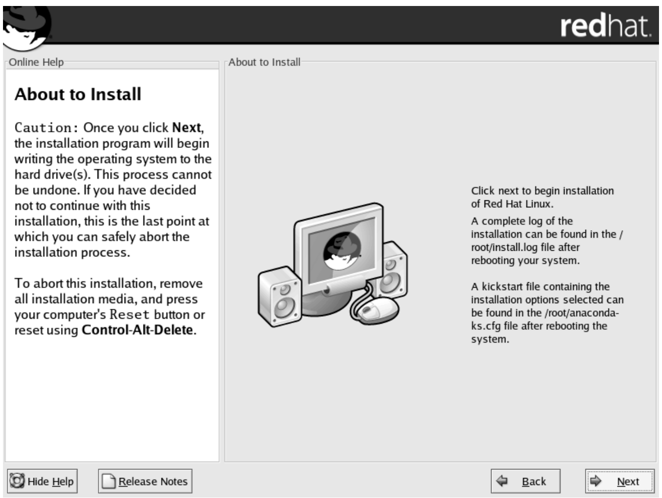


Figure 26: Are you ready to install?

For this course: Press the *Next* button.

Creating a boot disk

It is highly recommended that you create a boot diskette. If for some reason your system were not able to boot properly using GRUB, LILO, a boot diskette would enable you to properly boot your system.

Figure 27: Creating a boot disk

For this course: Insert a blank floppy and click on the *Next* button.

Installing X

At this point the installation program will probe your system in an attempt to determine which video card you have. If this probing fails then you would be presented with a list of video cards and monitors for you to select from.

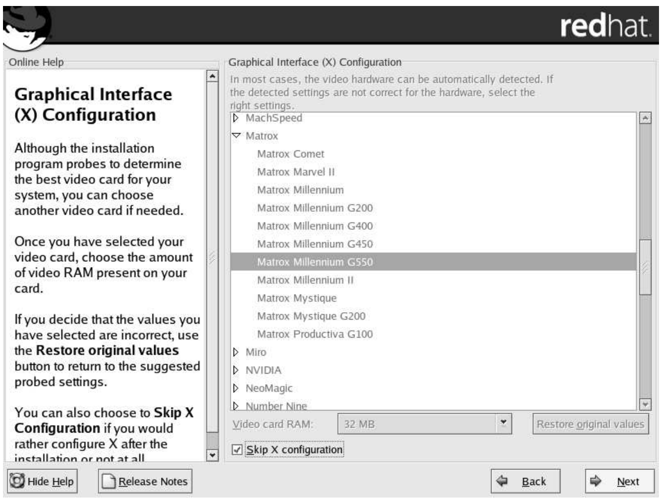


Figure 28: Installing X

For this course: Click on the *Skip X configuration* button because we are going to install X in the next section of the course.

Congratulations

Congratulations you have now installed Red Hat. Simply remove your floppy and reboot your system!

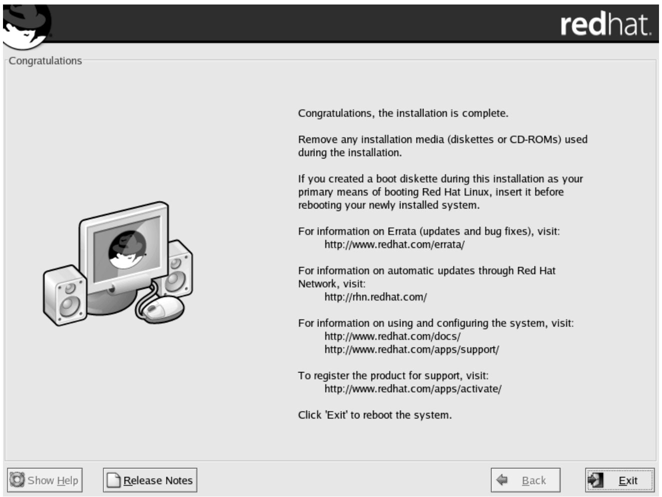


Figure 29: Congratulations

For this course: Press the *Exit* button.

2 XFree86

2.1 Introduction

This section describes how to configure `X`. Despite it normally working out of the box after an install, on these machines it does not. You will re-configure it from scratch since that will give you a much more thorough understanding of the configuration file than just using tools.

2.1.1 Graphics cards

This section attempts to be as general as possible, but assumes that the target hardware is vaguely modern, so for example the graphics card will support extensions added to the standards many years ago. Much of this section will apply regardless, but it will require more research if the behind the scenes magic that normally happens were to fail. If this were to happen then you are urged to get more modern hardware in order to run `X`. However, as these machines demonstrate, the state of the art with Linux lags behind the platform for which the hardware was developed. Thus with sufficiently recent hardware it can be a struggle to get a decent display.

2.1.2 Framebuffers and VESA mode

Linux has the concept of a kernel framebuffer. This is the kernel's view on the graphics hardware. It is possible via some boot time magic (in this case vga=795) to turn this feature on. It is possible to force `X` to work using this framebuffer in preference to the graphics card itself. This has the advantage of only requiring one aspect of the system to understand the graphics hardware. It also uses a more simplistic interface to the graphics hardware. The disadvantage is that it will not be as fast as a dedicated driver written for that particular card.

Similarly `X` has a VESA driver which uses a generic interface to the graphics hardware. Again in more awkward situations this driver will work when there is no dedicated hardware support, but at the expense of speed.

2.1.3 Some DDC magic

Before configuring `X` here is some background information to uncover some of the magic you will see. The biggest bit of magic in configuring graphics on a PC comes under the acronym DDC. This is "Display Data Channel" created by VESA (Video Electronics Standard Association) and is how the computer, graphics card, and monitor communicate with each other. Assuming all the parts in the chain (graphics card, cable, and monitor) are set up to use DDC then the operating system can calculate the capabilities of the display system.

In order to check that your computer has a functioning DDC setup follow the instructions in slide 22.

```
# /usr/sbin/ddcprobe

Videocard DDC probe results
Description:  Matrox G550
Memory (MB):  32

Monitor DDC probe results
ID: IVM4650
Name: AS4315
Horizontal Sync (kHZ): 30-82
Vertical Sync (HZ)  : 50-75
Width (mm): 340
Height(mm): 270
```

Slide 22: Exercise: Checking DDC

`ddcprobe` can fail in two main ways. If the graphics card does not support the appropriate extensions, you will see the final error message:

```
VESA BIOS Extensions not detected
```

If the monitor does not support DDC or if the monitor supports it but the cable connecting the graphics card with the monitor is missing a wire, the final error message after the graphics card has been probed would be:

```
EDID read failed.  (No DDC-capable monitor attached?)
```

2.1.4 Power Management

Monitor power management is handled by DPMS (Display Power Management Signaling). DPMS, also created by VESA, allows the computer to switch a monitor into various power saving states. All modern monitors will have this facility, and if in doubt it should be assumed that a monitor supports DPMS.

2.2 Configuring XFree86

Despite improvements with `XFree86` the graphics cards in these machines are still not completely supported, however things have become a lot easier over the intervening 12 months. However the sequence shown is the *easiest* one, so whilst the machine will work perfectly almost from the moment you finish using the GUI it would be a lot harder if a slightly different route had been chosen.

Also be warned that *all* the possible ways of configuring `X` have their own problems.

Configuration Tools

`redhat-config-xfree86`	A Red Hat tool
`xf86config`	The original tool from XFree86
`XFree86 -configure`	The newer tool from XFree86
Create your own	The good old Unix way

Additional tools

`ddcprobe`	Identifies graphics cards and monitors
`gpm`	Identifies mice
`X -probeonly`	Requires a reasonable `XF86Config` file. Tells you what the `X` server sees.

Slide 23: How to Configure XFree86

Red Hat provide `redhat-config-xfree86` provide for you to configure `X`. It isn't generic, so normally would not be taught on this course, however this particular combination of hardware is more awkward than most so for these purposes that is what we will use. Normally it would be suggested that you used `XFree86 -configure` and then copy the generated configuration file into place. After that the procedure is broadly the same.

Before `redhat-config-xfree86` can be used succesfully on this hardware a minimal amount of preparation is needed. The Matrox supplied drivers need to be installed in order for the graphics cards in these machines to be made to work with the attatched monitors. Normally you would search the web and find the appropriate drivers, in this case from `http://www.matrox.com/mga/`, but the course cannot afford to rely on their website working, so if you follow the instructions in exercise 24 then you will download the drivers from a local copy.

```
# cd /usr/src
# ftp ftp-uxsup.csx.cam.ac.uk
Connected to ftp-uxsup.csx.cam.ac.uk (131.111.8.10).
220 ProFTPD 1.2.2 Server (ProFTPD Unix Support Installation) [nurse.csi.cam.ac.uk]
Name (ftp-uxsup.csx.cam.ac.uk:root): ftp
331 Anonymous login ok, send your complete email address as your password.
Password: Your Email address@cam.ac.uk
230-This is the anonymous FTP server for the University of Cambridge Computing
 Service's Unix Support group.  Its standard name is ftp-uxsup.csx.cam.ac.uk.
 It is meant for use from within the .cam.ac.uk domain.

 Please report any problems to Unix Support (unix-support@ucs.cam.ac.uk).
 Our web pages can be found at http://www-uxsup.csx.cam.ac.uk/ .

230 Anonymous access granted, restrictions apply.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> cd /pub/linux/redhat/local_extras
250 CWD command successful.
ftp> get mgadrivers-3.0.tgz
local: mgadrivers-3.0.tgz remote: mgadrivers-3.0.tgz
227 Entering Passive Mode (131,111,8,10,212,120).
150 Opening BINARY mode data connection for mgadrivers-3.0.tgz (4549422 bytes).
226 Transfer complete.
2916194 bytes received in 1.04 secs (4.5e+03 Kbytes/sec)
ftp> quit
221 Goodbye.
```

Slide 24: Exercise: Fetching the Matrox drivers

Now install the drivers that you have just downloaded by following the instructions in exercise 25.

```
# tar -zxf mgadrivers-3.0.tgz
# cd mgadrivers
# ./install.sh
```

Answer “yes” to any questions once you have *read* them.

Slide 25: Exercise: Installing Matrox drivers

Now the `redhat-config-xfree86` command will be able to run, and produce a template file which can be used as a basis for configuring `X` more fully.

redhat-config-xfree86

This command may take quite a long time to start up, and when it does you will find that on these machines your mouse pointer is invisible. Your screen should look something like figure 30. If your mouse didn't work then you could use the tab and arrow keys instead.

- Press tab once to move to the Resolution menu.
- Press the space bar once to activate the menu.
- Press down arrow 4 times to highlight 1280x1024.
- Press the enter key to select.
- Press tab once to move to the Color Depth section.
- Press the space bar once to activate the menu.
- Press down arrow once to select "Millions of colors".
- Press [Alt]+[A] for the Advaned options.
- Press [Alt]+[E] to enable Hardware 3d Acceleration.
- Press [Alt]+[O] to end the configuration.
- Press [Alt]+[O] to end the program.

Slide 26: Exercise: redhat-config-xfree86

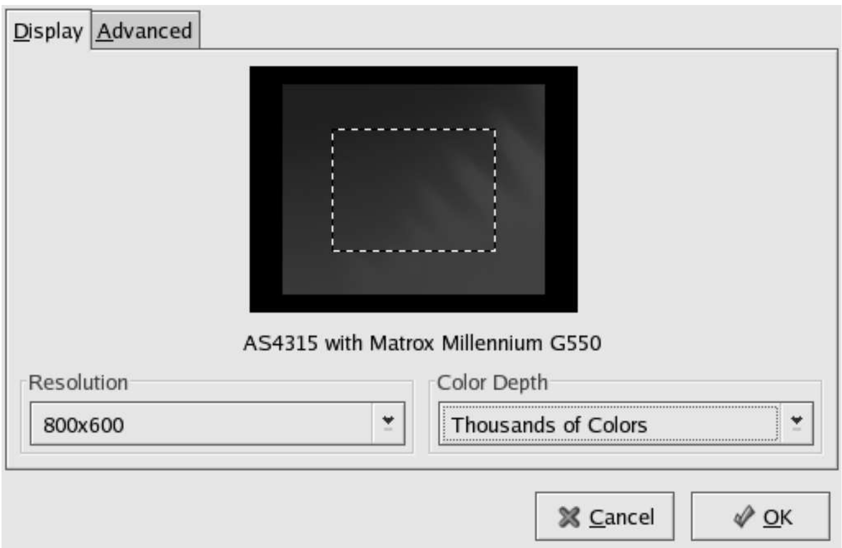


Figure 30: Running `redhat-config-xfree86` Display

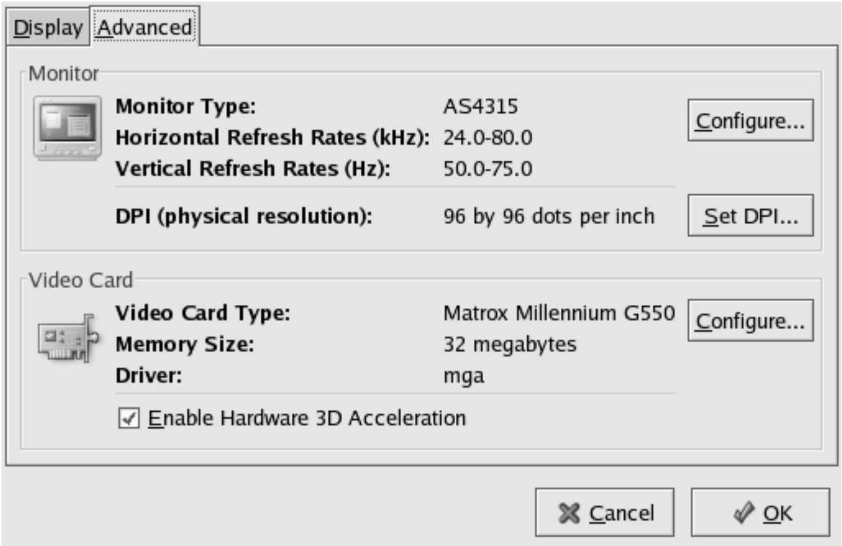


Figure 31: Running `redhat-config-xfree86` Advanced

Now you should have a configuration file `/etc/X11/XF86Config` which should allow you to start up `X`. Follow the instructions in slide 27 to check that this is the case. Once it has started you should be presented with a recognisable desktop.

In order to test the configuration type
\# **`startx`**
This may take up to 3 minutes due to the extreme nature of the interaction of the hardware combination in use.

In order to exit X and return to text mode you can either follow the instructions for that particular window manager, or you can force an abrupt exit by pressing Control-Alt-Backspace.

Slide 27: Exercise: Testing XF86Config configuration

There are several sections in the `/etc/X11/XF86Config` file generated by `redhat-config-xfree86`. All the possible sections are listed in slide 28. The contents of these sections will be largely self explanatory, although the syntax for changes will need some explaining.

•	`Section "ServerLayout"`	Binds all the other sections together.
•	`Section "Files"`	Lists details of 'files' that are used.
	`Section "Module"`	Lists modules to be loaded.
•	`Section "InputDevice"`	Keyboard and Mice.
•	`Section "Monitor"`	Details about the monitor.
	`Section "Modes"`	Details about the different monitor modes.
•	`Section "Device"`	Some details about the graphics card.
•	`Section "Screen"`	Links together the monitor and graphics card.
	`Section "DRI"`	Direct Rendering Infrastructure (DRI) configuration.
	`Section "ServerFlags"`	Sets server options.
	`Section "VideoAdaptor"`	Unknown. See `XF86Config`(5x)

Marked sections are explained, others are just listed for completeness.

Slide 28: XF86Config

2.2.1 Device

```
Section "Device"
        Identifier  "Videocard0"
        Driver      "mga"
        VendorName  "Videocard vendor"
        BoardName   "Matrox Millennium G550"
EndSection
```

Slide 29: Default Graphics card config

The Device section describes the graphics card. The default that these machines generate is shown in slide 29. The "Identifier" is a unique identifier within the file, in this case for a specific graphics card. The only other directive that is always needed is "Driver". Everything else can be deleted or commented out. The "VendorName" and "BoardName" are just strings which are useful for humans reading the configuration file. This is the section where you would specify that X should use VESA or the framebuffer by selecting the appropriate drivers.

- Identifier *"name"*
- Driver *"driver"*
- VendorName *"vender"*
- BoardName *"board name"*
  BusID *"bus-id"* (Specify one from several graphics cards)
- Screen *number*
  Chipset *"chipset"*
  ChipID *id*
  ChipRev *rev*
  Ramdac *"ramdac-type"*
  DacSpeed *speed*
  DacSpeed *speed-8 speed-16 speed-24 speed-32*
  Clocks *clock ...*
  ClockChip *"clockchip-type"*
- VideoRam *mem*
  BiosBase *baseaddress*
  MemBase *baseaddress*
  IOBase *baseaddress*
  TextClockFreq *freq*
- Options

Marked directives are explained, others are just listed for completeness.

Slide 30: Device directives

The “Option” keyword allows you to specify different modifiers to the behaviour. There are a large number of possible options for each driver. Previously the “HWcursor” option needed to be set to “no” in order to make the mouse pointer function. The current release of the Matrox drivers does not have this issue any more. This course cannot possibly hope to go through the options in anything like complete detail, instead we will direct you to the current documentation at `http://www.xfree86.org/current/`.

The “Screen” directive isn’t normally needed, but can be useful when the graphics card is able to drive more than one monitor, which in theory these cards can do.

The “VideoRam” directive allows you to set how much memory on the graphics card the X server will use. Normally the amount of memory on the graphics card will be autodetected and the full amount used. Sometimes it is necessary to specify though. Note that if you specify a particular amount then that is how much X will attempt to use. If this number is too high then the X server will probably crash the machine. If the number is too low then it won’t take advantage of the extra memory, thereby either reducing your maximum resolution or maximum number of colours, or it will stop some speed enhancing buffering from happening.

The current release of Red Hat configures the graphics card correctly so no alteration is needed to this section.

2.2.2 Mice

One of the most common problems with X is that it has failed to identify the mouse, in particular when you change the mouse device. This should become rarer as badly defined serial mice become rarer.

```
Section "InputDevice"
        Identifier  "Mouse0"
        Driver      "mouse"
        Option      "Protocol" "IMPS/2"
        Option      "Device" "/dev/input/mice"
        Option      "ZAxisMapping" "4 5"
        Option      "Emulate3Buttons" "no"
EndSection
```

Slide 31: Default Mouse Configuration

The default guessed configuration for mice on these machines is shown in slide 31. The Identifier can be any unique identifier string. The Driver is, fairly obviously, “mouse”. Then there are a number of options depending on the exact mouse being used, and the preferred settings for the mouse.

In order to write this section by hand you would first need to identify the interface being used. Serial mice plug into a serial port, which is one with either 9 pins or 25 pins male in a “D” shaped connector. PS/2 mice plug into a round DIN 6-pin female connector. USB devices generally have a flat 4 pin (male) USB connector. Once the interface has been identified you need to locate the appropriate entry in `/dev`. Almost without exception you will find that `/dev/mouse` has already been set up during the installation. If the link is not there then you are advised to set it up. In order to create this link you need to know what the device is. Alternatively, if you do not wish to use a link then you need to know what the device is in order to use that device rather than `/dev/mouse`. If it is a serial mouse then depending which port it is connected to it will be one of `/dev/ttyS0`, `/dev/ttyS1`, `/dev/ttyS2` or `/dev/ttyS3`. If it is a PS/2 mouse it will be `/dev/psaux` (also known as `/dev/psmouse`). If as here it is a USB mouse then the easiest choice is `/dev/input/mice`.

Once the interface has been organised the specific details for that mouse using that connector needs to be selected.

- Bus mouse

  Too rare to bother with

- Serial mouse
  - Most vaguely modern serial mice are auto-detected.
  - Most of the rest are or pretend to be “Microsoft” mice.
  - Protocol options include:

    Auto, Microsoft, GlidePoint, IntelliMouse (Wheel mice), Logictech, MMHittab, MMSeries, MouseMan, MouseSystems and ThinkingMouse

- PS/2 mouse
  - Most PS/2 mice should work with “auto”.
  - All PS/2 mice should be able to work as simple PS/2 mice.
  - Protocol options include:

    Auto, PS/2 and IMPS/2 (Wheel mice)

- USB mouse
  - These are new and in the author’s experience it seems that they appear as “ps2” mice.
  - Protocol options include:

    Auto, usb, ExplorerPS/2 (Wheel mice)

Slide 32: Mouse types under X

Other options exist, such as telling `X` how many buttons the mouse has, how to handle middle mouse button, and whether it is a wheel mouse (ZAxisMapping). More details can be found in `README.mouse` part of the distribution of `X` (`xc/programs/Xserver/hw/xfree86/doc/README.mouse`) which google should locate if you do not wish to download several megabytes of source code.

Additionally the generated configuration file demonstrates another trick that can be used in slide 33. It is possible to specify that if any USB mice are added that they can be used too. This is only relevant if a non-USB mouse is needed. On these machines which have USB mice it will support all the USB mice that you can attatch.

```
Section "InputDevice"
# If the normal CorePointer mouse is not a USB mouse then
# this input device can be used in AlwaysCore mode to let you
# also use USB mice at the same time.
        Identifier  "DevInputMice"
        Driver      "mouse"
        Option      "Protocol" "IMPS/2"
        Option      "Device" "/dev/input/mice"
        Option      "ZAxisMapping" "4 5"
        Option      "Emulate3Buttons" "no"
EndSection
```

Slide 33: Multiple Mice

2.2.3 Keyboards

Slide 34 shows the default keyboard setup from `redhat-config-xfree86`. As before, “Identifier” is a unique identifier for “InputDevices” within the file, and the “Driver” configures the type of device, in this case a keyboard.

```
Section "InputDevice"
# Specify which keyboard LEDs can be user-controlled (eg, with xset(1))
#       Option  "Xleds"         "1 2 3"

# To disable the XKEYBOARD extension, uncomment XkbDisable.
#       Option  "XkbDisable"

# To customise the XKB settings to suit your keyboard, modify the
# lines below (which are the defaults).  For example, for a non-U.S.
# keyboard, you will probably want to use:
#       Option  "XkbModel"      "pc102"
# If you have a US Microsoft Natural keyboard, you can use:
#       Option  "XkbModel"      "microsoft"
#
# Then to change the language, change the Layout setting.
# For example, a german layout can be obtained with:
#       Option  "XkbLayout"     "de"
# or:
#       Option  "XkbLayout"     "de"
#       Option  "XkbVariant"    "nodeadkeys"
#
# If you'd like to switch the positions of your capslock and
# control keys, use:
#       Option  "XkbOptions"    "ctrl:swapcaps"
# Or if you just want both to be control, use:
#       Option  "XkbOptions"    "ctrl:nocaps"
#
        Identifier  "Keyboard0"
        Driver      "keyboard"
        Option      "XkbRules" "xfree86"
        Option      "XkbModel" "pc105"
        Option      "XkbLayout" "gb"
EndSection
```

Slide 34: Default keyboard setup

There are two “XkbRules”, either “xfree86”, or “xfree98” for Japanese keyboards.

The default keyboard, as shown in slide 34, is “pc105” a 105 key PC keyboard (basically anything with the now standard “Windows^{TM}” keys. On older keyboards there were fewer keys and “pc101” should be used instead. The correct keyboard for these machines is “pc105”. Other options include “pc102”, “pc104” and “microsoft” (which covers their natural keyboard range).

For any given keyboard the layout will be normal in a specific country. The “XkbLayout” should be set to the country code for that country. For the UK the country code is “gb”.

“XkbOptions” can probably have lots of options, and somewhere they may even be documented. The only one which I know is in use by some people is “ctrl:swapcaps” which switches the capslock and control keys around.

Change the keyboard section of `/root/XF86Config.new` to look as below, by removing all the comments. This is purely a cosmetic change to make the file easier to read.

```
Section "InputDevice"
        Identifier  "Keyboard0"
        Driver      "keyboard"
        Option      "XkbRules"      "xfree86"
        Option      "XkbModel"      "pc105"
        Option      "XkbLayout"     "gb"
EndSection
```

Confirm that the `X` server still works by following the instructions in slide 27.

Slide 35: Exercise: Setting up the keyboard

It is worth noting that the keyboards in here have three additional keys “WWW”, “Mail” and “Search”. It is possible to attribute functionality to these keys; all you need to know is that they have keycodes “178”, “236” and “229” respectively. It is possible to identify the keycodes with `xev`. For example in order to make the “WWW” key print “w” when pressed all that is needed is, when running `X`, to type `modmap -e "keycode 178 = w"`. For more advanced functionality have a look at: `http://lineak.sourceforge.net/`.

See `http://portal.suse.com/sdb/en/2001/10/cg_spckeyboard.html` for details on how to achive the same sort of effect in text mode.

2.2.4 Files

```
Section "Files"
# RgbPath is the location of the RGB database.  Note, this is the name of the
# file minus the extension (like ".txt" or ".db").  There is normally
# no need to change the default.

# Multiple FontPath entries are allowed (they are concatenated together)
# By default, Red Hat 6.0 and later now use a font server independent of
# the X server to render fonts.

        RgbPath      "/usr/X11R6/lib/X11/rgb"
        FontPath     "unix/:7100"
EndSection
```

Slide 36: Default “Files” section

The Files section references all the static files that are useful to the `X` server. In fact it is slightly misnamed, since not everything it references is a file. Slide 37 shows the syntax for adding a reference to the font path, which can either be a font server (on the local machine or on a remote one) or a directory which contains fonts. The font path specified can be overridden by the user with `xset` when running `X`.

- Using font servers
  Either `FontPath "tcp/`*`hostname`*`:`*`portnumber`*`"`
  or `FontPath "unix/:`*`portnumber`*`"`
- Using directories with fonts
  Either `FontPath "`*`PathToFonts`*`"`
  or `FontPath "`*`PathToFonts`*`:unscaled"`
- An Example
  `FontPath "`*`/usr/X11R6/lib/X11/fonts/75dpi/`*`:unscaled"`
  `FontPath "unix/:7100"`

Slide 37: Setting the font path

The ":unscaled" directive shown in slide 37 allows the selection of fonts to prefer bitmapped fonts where the size matches the requested size, but if that fails then use scalable fonts, and if that fails to provide a match to fall back to scaling bitmap fonts (which never works very well). In the example the `X` server will use the local 75 DPI fonts if it doesn't have to scale them, otherwise it will ask the font server.

Due to the way that Red Hat have decided to make `X` work the configured FontPath is correct. It uses the font server which Red Hat run, and talk to this font server over a unix domain socket.

The Files section can also specify the RgbPath and the ModulePath. The former is the name of the file which specifies a small fraction of the total number of colours available (but without the file extension, since there are lots of different options for historical reasons). The latter affects where the `X` server looks for modules. Neither of these two options are needed since the `X` server has a default, and this default is correct for Red Hat.

Instead of installing Matrox's drivers using the provided install script it is also possible to install them by hand, and place them in a separate location, and then just add a reference to that location with the ModulePath directive.

2.2.5 Monitor

The Monitor section provides details about the monitor. The details listed in slide 38 are the based on the results of probing the monitor with DDC. The Identifier can be any unique identifier string which labels one of potentially many monitors within the configuration file. It is merely a label and the specific value has no significance. Similarly the VendorName and ModelName have no significance, except as identifiers.

```
Section "Monitor"
     Identifier   "Monitor0"
     VendorName   "Monitor Vendor"
     ModelName    "AS4315"
     DisplaySize  340        270
     HorizSync    30.0 - 82.0
     VertRefresh  50.0 - 75.0
     Option      "dpms"
EndSection
```

Slide 38: Default Monitor section

Where the DDC probe was successful this section does not need to list any technical details about the monitor on the grounds that it will be able to ask the monitor again when it starts up. You have already seen some of the information `X` can extract using DDC in section 2.1.4.

Where the probe isn't successful then these technical details would need to be specified. These details can normally be found in the monitor manual, or in many cases from `/usr/share/hwdata/MonitorsDB` although this often requires DDC to work correctly in order to identify the monitor. Also the World Wide Web can normally provide some useful details about monitors.

Make the monitor section in `/etc/X11/XF86Config` look like:

```
Section "Monitor"
     Identifier   "Monitor0"
     VendorName   "Iiyama"
     ModelName    "AS4315"
     Option       "dpms"
EndSection
```

Confirm that the `X` server still works by following the instructions in slide 27.

Slide 39: Exercise: Updating the monitor configuration

The DPMS option shown in slide 39 enables power management features, allowing `X` to power down the monitor when the machine is not being used from console.

There are other directives which can be used in the "Monitor" section, some of these are listed in slide 40. The DisplaySize is useful when the `X` server needs to attempt to apply WYSIWYG literally, it is specified number of millimeters horizontal and vertical.

Mode and ModeLine specify monitor timings, which allow you to adjust the exact position and size of the display on the monitor. `http://www.linuxdoc.org/HOWTO/XFree86-Video-Timings-HOWTO/` has more details as does the `XF86Config`(5x) manual page. If you wish to experiment with this then you are urged to look at the `xvidtune` command, which allows you to create the modeline interactively. The UseModes directive allows you to use a "Modes" section, which group some Modes and ModeLines together. The "Modes" section allows you to use these modes with multiple monitor. `XF86Config`(5x) has more information on this if you should need it. Normally you wouldn't need to play with this when using a digital connection between the graphics card and the monitor, as is the case here.

•	VendorName "*vendor*"	Optional monitor's manufacturer name
•	ModelName "*model*"	Optional monitor's model name
	HorizSync "*horizontal sync range*"	Optional horizontal sync range for monitor.
	VertRefresh "*vertical refresh range*"	Optional vertical refresh range for monitor.
•	DisplaySize	Specify the width and height of the monitor display.
	Gamma *gamma*	Adjust the gamma correction for the monitor.
	Gamma *red-gamma green-gamma blue-gamma*	Adjust the gamma correction for each colour.
•	UseModes	Specify a Modes section containing monitor timings
•	Mode	Specify a monitor timing here.
•	ModeLine	specify a monitor timing here, but more compactly.
•	Options	Specify a number of options, such as DPMs

Marked directives are explained, others are just listed for completeness.

Slide 40: Monitor section directives

2.2.6 Screen

Slide 41 shows the default state for the screen section. The purpose of the Screen section is to tie in the graphics card and monitor in a single section.

The "Identifier" is, as ever, a unique identifier of Screen sections. The "Device" references a graphics card via the unique identifier used in a Device section. Similarly the "Monitor" references a monitor via the unique identifier of a Monitor section. Thus a specific monitor and graphics card are linked together.

```
Section "Screen"
    Identifier "Screen0"
    Device     "Videocard0"
    Monitor    "Monitor0"
    DefaultDepth     24
    SubSection "Display"
        Depth     16
        Modes  "800x600" "640x480"
    EndSubSection
    SubSection "Display"
        Depth     24
        Modes  "1280x1024" "1280x960" "1152x864" "1024x768" "800x600" "640x480"
    EndSubSection
EndSection
```

Slide 41: Default Screen section

There needs to be a "Display" subsection which specified options for the bit depth that is used. The driver for the graphics card will have a default bit depth, commonly 8 (bits) which represents 256 colours. This can be overridden in the main "Screen" section with a "DefaultDepth" directive.

The directives in the Display section are listed in slide 42

- Depth *depth*
- Modes ”*mode-name*” ...

  FbBpp *bpp*

  Weight *red-weight green-weight blue-weight*
- Virtual *xdim ydim*
- ViewPort *x0 y0*

  Visual ”*visual-name*”

  Black *red green blue*

  White *red green blue*

  Options

Marked directives are explained, others are just listed for completeness.

Slide 42: Display Subsection

The “Depth” specifies the depth of the display. It is given in numbers of bits. 8 bits represents 256 colours. 16 bits means 65536 colours. 24 bits means 16.7 million colours. 32 bits also means 16.7 million colours although the extra 8 bits aren’t used for colour information (and often not for anything). The human eye struggles to perceive more than 16.7 million colours. See

`http://developer.gnome.org/feature/archive/xconcepts/x-concepts.html#VISUALS-CMAPS`

for a more detailed explanation.

The “Modes” directive lists all the modes which are valid at that bit depth. It is possible to specify more than one mode, although in my experience there is little point. The `X` server has a number of modes built into it, a probably exhaustive list for the builtin “Modes” to chose from is:

640x350	640x400	640x480	720x400
800x600	1024x768	1152x864	1024x768i
1280x960	1280x1024	1400x1050	1600x1200
1792x1344	1856x1392	1920x1440	

(Note that 1024x768i is an interlaced display, and should in general be avoided as it will flicker badly).

Additionally it is possible to specify further modes in the “Monitor” section, which can be referenced here.

The “Virtual” directive allows you to have a virtual desktop which is larger than the screen size, and is scrolled around by moving the mouse to the edge of the display. “Viewport” dictates the starting view onto the large virtual desktop. Both these options tend to be unnecessary since the equivalent functionality is handled by the window manager.

Before changing the configuration file it is worth noting that whilst the keywords in the `XF86Config` file are not case sensitive any filenames (which includes names of loaded modules) will be.

Set up the Screen section in `/root/XF86Config.new` to be:

```
Section "Screen"
  Identifier   "Screen0"
  Device       "Videocard0"
  Monitor      "Monitor0"
  DefaultDepth 24
  Subsection   "Display"
     Depth 24
     Modes "1280x1024"
  EndSubSection
EndSection
```

Confirm that the `X` server still works by following the instructions in slide 27.

Slide 43: Exercise: Setting up the Screen section

2.3 ServerLayout

The “ServerLayout” section binds the “Screen” and “InputDevice” sections together. Slide 44 shows the default for these machines. Whilst there is no way within the configuration file to refer to the Identifier in this section (unlike all the other identifiers in other sections) it is possible to have more than one “ServerLayout” section and refer to it on the command line by using the -layout option. However in most situations only one “ServerLayout” is appropriate for a given workstation.

The “Screen” directive has the format:

Screen *screen-num* ”*screen-id*” *position-information*

`XF86Config`(5x) has more information but under most circumstances the *screen-id* would be the “Identifier” of the appropriate screen section (such as `"Screen0"`), and the *screen-num* would be 0, and the positional information would be 0 0.

The keyboard and mouse are specified with an InputDevice entry each which references the Identifier for the relevant InputDevice section (in this case `"Mouse0"` and `"Keyboard0"`). It is possible to specify multiple devices, but only one can be the primary device. The Corepointer and CoreKeyboard flags mark the primary device of each type.

It is also possible to have Options directives, where any option permitted in the ServerFlags section can be specified. See `XF86Config`(5x) for more information.

```
Section "ServerLayout"
        Identifier     "XFree86 Configured"
        Screen      0  "Screen0" 0 0
        InputDevice    "Mouse0" "CorePointer"
        InputDevice    "Keyboard0" "CoreKeyboard"
EndSection
```

Slide 44: Default SeverLayout

2.4 Commercial X servers

Commercial X servers are rarely worth using under Linux unless for some reason you need to use a very obscure card. The cheapest graphics cards will probably provide adequate performance for most X use. If you do want to use a graphics intensive package and a specific graphics card, then it may be worth considering a commercial server. One common example of this is with laptops. If you do need to look at commercial servers then look at:

- `http://www.metrolink.com/`
- `http://www.acceleratedx.com/`

2.5 Window Managers

Now you should have a working `X` setup. Before you use it in earnest you will need to get to grips with the windowing environment. Largely the current collection of different windowing environments work in a similar fashion.

Red Hat currently defaults to using `Gnome` (which is technically a session manager and determines the look of the dekstop) alongside `Metacity` (which is a minimalist window manager allowing you to control the location and size of all the applications which are started on this X display). However there are plenty of others that you can choose from. I do not intend to go through them all since there are a huge number of possible ones, from the traditional if underfeatured `twm` through to `Enlightenment` which is infinitely configurable but has quite high system requirements.

This section will be less structured than the previous ones. I will let you become familiar with the more important portions of the Gnome desktop environment.

This section used to attempt to teach you how to configure the desktop environment for other people. However that is proving harder and harder as the configuration is becoming more and more scattered around various files many of which are written in markup languages like XML or are created using GUI development tools like Glade.

You should now create a new user, as in exercise 45, so that you can start from fresh.

First, create a non-root user:

```
# useradd username
# passwd username
```

Now switch to another virtual console with [Alt]+[F2], and log in as this new user.
No explanation is given of the commands above which will be covered elsewhere.

Slide 45: Exercise: Creating a new user

```
$ ls -la
total 28
drwx------ 2 user ... Dec 11 12:20 .
drwxr-xr-x 3 root ... Dec 11 12:20 ..
-rw-r--r-- 1 user ... Dec 11 12:20 .bash_logout
-rw-r--r-- 1 user ... Dec 11 12:20 .bash_profile
-rw-r--r-- 1 user ... Dec 11 12:20 .bashrc
-rw-r--r-- 1 user ... Dec 11 12:20 .emacs
-rw-r--r-- 1 user ... Dec 11 12:20 .gtkrc
```

Slide 46: Before

• Exit X, and log in as a new user	Start from fresh
• $ **`ls -la`**	See what is there at the moment
• $ **`startx`**	Start X
• rightclick on the desktop→New Terminal	Open up a terminal
• $ **`ls -la`**	See what is there now

Slide 47: Exercise: Gnome/Metacity config files

Having started X, as detailed in Slide 47, you will find yourself running a rather pretty *pair* of window/display managers, with various different windows and icons thrown in for good measure. As you can see in Slide 48 a fair few directories have been created by Metacity and Gnome.

```
$ ls -la
total 124
drwx------ 10 user ... Dec 11 12:31 .
drwxr-xr-x  3 root ... Dec 11 12:20 ..
-rw-r--r--  1 user ... Dec 11 12:20 .bash_logout
-rw-r--r--  1 user ... Dec 11 12:20 .bash_profile
-rw-r--r--  1 user ... Dec 11 12:20 .bashrc
-rw-r--r--  1 user ... Dec 11 12:20 .emacs
-rw-rw-r--  1 user ... Dec 11 12:31 .fonts.cache-1
drwx------  5 user ... Dec 11 12:31 .gconf
drwx------  3 user ... Dec 11 12:31 .gconfd
drwx------  5 user ... Dec 11 12:31 .gnome
drwx------  5 user ... Dec 11 12:31 .gnome2
drwx------  2 user ... Dec 11 12:30 .gnome2_private
drwxr-xr-x  2 user ... Dec 11 12:31 .gnome-desktop
-rw-r--r--  1 user ... Dec 11 12:20 .gtkrc
-rw-rw-r--  1 user ... Dec 11 12:31 .gtkrc-1.2-gnome2
-rw-------  1 user ... Dec 11 12:31 .ICEauthority
drwx------  3 user ... Dec 11 12:31 .metacity
drwxr-xr-x  3 user ... Dec 11 12:31 .nautilus
-rw-------  1 user ... Dec 11 12:31 .recently-used
-rw-------  1 user ... Dec 11 12:31 .rhn-applet.conf
-rw-------  1 user ... Dec 11 12:30 .Xauthority
```

Slide 48: After

2.6 Play time

Unlike most traditional window managers this system does not really provide you with a single configuration file that you, as a user, can manually edit. Instead it provides you with a reasonably comprehensive set of tools to edit settings graphically.

In order to familiarise yourself with the Gnome desktop environment and the Metacity window manager you should play with them with both the root user and a non root user. Play with any and all options that you can find, see what you can change. In particular do the things listed in Slide 49 and if you have time then try the things in Slide 50, the aim is to feel “at home” with the desktop environment.

- Log in as a user.
- Examine the Gnome preferences (in the main menu).
  Note that Metacity configuration is part of this.
  Hint: If you hover your mouse over a menu item you will see the "tool tips"
- Examine the System Settings (in the main menu).
- Change the background image.
- Configure the screensaver.
- Launch a (Gnome) terminal.
- Change the colours and fonts of Gnome Terminal and make the background transparent.
- Change which theme the current user is running.
- Add something to the menu.
  Hint: Rightclick somewhere on the open menu.
- Install a new font.
  To download a font type: wget ftp://ftp-uxsup.csx.cam.ac.uk/pub/linux/redhat/local_extras/arial.ttf
  Hint: Go to location *fonts:* in Nautilus
- Check that you can now set Arial as the font in Gnome Terminal.

Slide 49: Exercise: The Gnome desktop environment

- Use the file manager: you can move, copy or link things to the desktop or elsewhere, using all three mouse buttons.
- Investigate the panel; how you can move it, how you can move elements around on it with the middle mouse button. There are *lots* of other options involving clicking each of the mice buttons in different locations.
- Drag things, drop things.
- Experiment. Ask for help if you cannot do something, or if you do something and do not understand what is going on.
- Challenge: make your desktop as horrific as possible.

Slide 50: Exercise: The Gnome desktop environment (optional)

File Managers

`Nautilus` is the default Gnome file manager. Some of you may not like file managers, but they do serve a purpose, in this case the file manager is necessary if you want to have icons on the desktop.

2.7 External Documentation

There is a lot of documentation online, much of which may prove useful. More details connected with X can be found at the following locations:

- `http://www-uxsup.csx.cam.ac.uk/courses/` - Configuring X (This is probably out of date now)
- `http://the.earth.li/XFree86/`
- `http://www.europe.redhat.com/documentation/rhl9/rhl-rg-en-9/s1-x-clients.php3`

More details on the use and configuration of Gnome/Metacity can be found at:

- `http://www.gnome.org/start/2.4/`
- `http://www.europe.redhat.com/documentation/rhl9/rhl-gsg-en-9/`

2.8 Graphical Logins

It is perfectly possible to arrange that you default to having a graphical login. In order to this you should change the run level to 5, of which more in in the Boot sequence section.

3 The Red Hat Software Package System

3.1 Software management

- Is something installed?
- Where is it installed?
- Program A needs program B.
- Installing program A broke program B.
- Have I deleted all the files for program A?

Slide 51: Problems with software management

Why do we need software management?

Why can't we just copy a program's files into place? If each program consisted of a single file this might be feasible, but a program typically comes as a collection of files: perhaps more than one actual executable program file, perhaps some support libraries, often some graphics files for a GUI application and so on.

Given this, how do you go about checking that a program is correctly installed? You have to know, or have documented somewhere, where every file is and what it should contain. Then you have to laboriously check them all. This is not a realistic approach. Furthermore, if a program depends on some configuration files, is there a systematic way to locate these files? If you just copy files in they can go anywhere; can you remember where you put them all?

A common problem in software occurs when you install program A and find it doesn't work because it relies on program B. Worse still, you can install program C which appears to work perfectly happily but, sometime later, you find that program D doesn't work, because program C included a file that overwrote one of program D's files or clashes with program D in a more subtle manner.

Suppose you want to remove a program. Can you remember where all the various files were? Even if you can, removing them manually can be a real pain. Also, you might have installed another program which depends on this one. Removing program B might break program A.

The solution to this problem is to stop treating files as the basic unit of software and work at a higher level by treating collections of them ("packages") as the lumps to manipulate. This will require extra tools corresponding, for example, to `cp` (to install) and `rm` (to uninstall).

- Software naturally comes as collections of files:

  programs, configuration files, libraries, graphics

- "Packages" as the unit of software control—

  not individual files

- Basic control:

  adding, removing, upgrading

- Advanced control:

  Dependencies, conflicts

Slide 52: Concepts of package management

Introducing the `rpm` program

One of Red Hat's major contributions to the Linux community was their writing of the Red Hat Package Manager program, `rpm`. This program, which has gone through a number of versions is one of the two standard package management tools used on Linux and probably the more common of the two. The `rpm` program is used by Red Hat Linux, S.u.S.E., TurboLinux, Yellow Dog Linux, Caldera Open Linux and many others. The other package manager is `dpkg`, which is used by the Debian and Corel distributions. `dpkg` is far more featureful and the absence of these features can be keenly felt from time to time by `rpm` users trying to do subtle things. Unfortunately it is rather more complicated as a result. However, it deserves mention as it has supplied many ideas to recent versions of `rpm` and the maintainers of both systems have stated that they benefit from not working in a vacuum.

While Red Hat provides a single command to do package management, this means that it must have several options. However, the first option specifies how the command is to be used; querying an installed or uninstalled package, installing a new package, erasing (removing) an installed package, or upgrading an installed package. The command, plus its first option may be considered as four separate tools as the leading options cannot be used in combination and one of them must be specified.

- RedHat Package Manager
- *One* program: `rpm`
- *Many* options!
- Five major modes:

```
rpm --query ...
rpm --install ...
rpm --erase ...
rpm --upgrade ...
rpm --verify ...
```

Slide 53: Manipulating packages

3.2 Querying packages

Using the `--query` option

We'll start with the `--query` option. This makes no changes to the system but merely makes enquiries.

The simplest use is to query the installed system to find out what packages are installed: `rpm --query --all` (slide 54). Note that there are very many packages and this command's output would typically be redirected to a file, piped to `more` or a filtering program (`grep` is common, as in slide 55).

```
$ rpm --query --all
setup-2.5.25-1
bzip2-libs-1.0.2-8
e2fsprogs-1.32-6
glib-1.2.10-10
...
emacs-21.2-33
...
vorbis-tools-1.0-3
gnome-media-2.2.1.1-4
comps-9-0.20030313
```

Slide 54: What packages are installed?

```
$ rpm --query --all | grep XFree86
XFree86-Mesa-libGL-4.3.0-2
XFree86-xfs-4.3.0-2
XFree86-100dpi-fonts-4.3.0-2
XFree86-4.3.0-2
...
XFree86-xauth-4.3.0-2
XFree86-tools-4.3.0-2
```

Slide 55: What XFree86 packages are installed?

Package version numbers

NAME-VERSION-RELEASE

Package names come in three components, separated by hyphens. The name of the software package can contain hyphens, but the version numbers cannot. The last two hyphens are treated as delimiters; any earlier ones are treated as part of the package name. Consider, for example, `XFree86-75dpi-fonts-4.3.0-2`. Its name is "`XFree86-75dpi-fonts`", its version number is "`4.3.0`" and its release number is "`2`".

The version number corresponds to the version of the underlying software. The release number increments when patches are applied to the base software or if the package is rebuilt with, say, a different configuration file.

Enquiring of a single installed package

In addition to the `--all` option, we can query an individual package. Note that in the example shown in slide 54 we saw package `emacs-21.2-33`. We can query the `emacs` package to determine the version number of the package with the command `rpm --query emacs` (slide 56).

Fully specified package names

We can make the same enquiry of a package name, quoting its version and release number or just its version number. However, getting the numbers wrong will stop any package being displayed.

```
$ rpm --query emacs
emacs-21.2-33

$ rpm --query emacs-21.2-33
emacs-21.2-33

$ rpm --query emacs-21.2
emacs-21.2-33

$ rpm --query emacs-21.2-33
package emacs-21.2-33 is not installed
```

Slide 56: Enquiring of individual packages

Getting more information about a package

There is much more information stored in a package than two version numbers. The `--info` suboption allows us to determine this extra information.

```
$ rpm --query --info emacs
Name        : emacs            Relocations: (not relocateable)
Version     : 21.2                  Vendor: Red Hat, Inc.
Release     : 33                Build Date: Thu 20 Feb 2003 06:41:42 AM GMT
Install date: Wed 26 Nov 2003   Build Host: porky.devel.redhat.com
Group       : Apps/Editors      Source RPM: emacs-21.2-33.src.rpm
Size        : 31244106             License: GPL
Signature   : DSA/SHA1, Mon 24 Feb 2003, Key ID 219180cddb42a60e
Packager    : Red Hat, Inc. <http://bugzilla.redhat.com/bugzilla>
URL         : http://www.gnu.org/software/emacs/
Summary     : The GNU Emacs text editor.
Description :
Emacs is a powerful, customizable, self-documenting, modeless text
editor. Emacs contains special code editing features, a scripting
language (elisp), and the capability to read mail, news, and more
without leaving the editor....
```

Slide 57: Detailed information on a package

The information categories

Name: the name of the package

Version: the version of the underlying software

Release: the release of this packaging of the particular version

Install date: when the package was installed onto the system

Group: Software is clumped into groups and subgroups. For example, the `emacs` package is an editor which is placed in the "`Applications/Editors`"[1] category.

Size: the total size of the package's files in bytes

Signature: a securtity feature establishing that this package was created by Red Hat

Packager: the company or person who created this packaging of the base software, not necessarily the same person that maintains the software

URL: a web page about the product, often containing the full documentation

Summary: a one-line description of the package

Relocations: Some software is "relocatable"; it can be installed in more than one location. This is not such a piece of software. Very little is.

Vendor: the company that supplied this package or the distribution the package came in, typically the same as the "Packager"

Build date: when the package was built

Build host: the system the package was built on

Source RPM: Just as the compiled software is packaged, there is a means to package the source code (and this is an essential component of building a software package). We won't be considering building our own packages in this course, so we won't consider source packages either.

License: the licence under which the packaged software is provided

Description: a longer description of the packaged software than is possible in the "Summary" line

```
$ rpm --query --list emacs
/etc/skel/.emacs
/usr/bin/b2m
/usr/bin/ebrowse
...
/usr/share/man/man1/gctags.1.gz
/usr/share/man/man1/gfdl.1.gz
/usr/share/pixmaps/emacs.png
```

Slide 58: Listing the files in a package

[1] I've cheated on the slide by replacing the word "Applications" with the word "Apps" to get the text to fit on the slide.

```
$ rpm --query --file /usr/bin/emacsclient
emacs-21.2-33
```

Slide 59: Determining the package for a file

Connecting files & packages

It is possible with `rpm` to derive from a package the complete list of files it contains with the query suboption `--list`. It is also possible to determine which package a file comes from with the query suboption `--file`.

```
# mkdir /redhat
# mount -o ro nfs-uxsup.csx.cam.ac.uk:/linux/redhat /redhat
# cd /redhat
# ls
6.2  7.2  8.0  beta  contrib  enterprise    ls-lR.gz  rawhide
7.1  7.3  9    code  current  local_extras  preview   updates
# cd 9/en/os/i386/RedHat/RPMS/
# ls
4Suite-0.11.1-13.i386.rpm
...
zsh-4.0.6-5.i386.rpm
```

Slide 60: Exercise: Packages before they are installed

Packages before they are installed

Next we are going to look at uninstalled packages; ones that your system doesn't know about (yet). Don't worry about the `mount` command; just trust us for now.

A package comes as a single archive file with a conventional filename. This is the format in which they are distributed and stored prior to installation. The format of the conventional filename is

NAME-VERSION-RELEASE.ARCHITECTURE.rpm

The filename doesn't need to take this form. The `rpm` program ignores the file name and works purely with the file's contents which contain the various components of the conventional name (slide 62).

- i386, i486, i586, i686
- alpha
- ppc
- sparc, sparc64
- noarch

Slide 61: Architecture tags

Architecture tags

The various architectures can be classed into a few categories.

i386: This is the standard Intel architecture. Packages with this extension should not have programs that rely on 486 (or beyond) extensions. Typically, for applications at least, the standard compilers just generate 386 code that will work on the full range of Intel chips so this architecture tag is equally applicable for all flavours.

i486, i586, i686: Typically the only packages where optimisations for specific chips are employed are the packages containing the kernel. However, there is no reason why other packages should not be built for these more specific architectures. Numerical libraries would be an obvious example.

alpha: DEC's (then Compaq's, now made by Intel) AlphaAXP chip

ppc: Motorola's PowerPC chip

sparc: This is the SPARC chip, used mainly in Sun workstations. The original SPARC chip was a 32-bit chip and the new 64-bit chips are capable of running the old 32-bit binaries.

sparc64: As with the Intel series, the kernel is the only package typically compiled separately for the newer, 64-bit, versions of the chip.

noarch: The special tag "`noarch`" is used for packages of files that have no dependence of a specific architecture. These are commonly used for sets of image files, or fonts that can be shared between architectures.

```
$ pwd
/redhat/9/en/os/i386/RedHat/RPMS

$ rpm --query --package 4Suite-0.11.1-13.i386.rpm
warning: 4Suite-0.11.1-13.i386.rpm: V3 DSA signature: NOKEY, key ID db42a60e
4Suite-0.11.1-10

$ cp 4Suite-0.11.1-13.i386.rpm /tmp/fubar

$ rpm --query --package /tmp/fubar
warning: /tmp/fubar: V3 DSA signature: NOKEY, key ID db42a60e
4Suite-0.11.1-13
```

Slide 62: Exercise: Querying a package file

```
$ rpm --query --info --package 4Suite-0.11.1-13.i386.rpm
warning: 4Suite-0.11.1-13.i386.rpm: V3 DSA signature: NOKEY, key ID db42a60e
Name        : 4Suite           Relocations: (not relocateable)
Version     : 0.11.1                Vendor: Red Hat, Inc.
Release     : 13                Build Date: 07 Feb 2003 12:31:02
Install date: (not installed)    Build Host: sylvester.devel.redhat.com
Group       : Development/Libs   Source RPM: 4Suite-0.11.1-13.src.rpm
Size        : 8683034              License: Apacheish
Signature   : DSA/SHA1, Mon 24 Feb 2003, Key ID 219180cddb42a60e
Packager    : Red Hat, Inc. <http://bugzilla.redhat.com/bugzilla>
URL         : http://www.4suite.org/
Summary     : Python tools and libraries for XML processing and databases.
Description :
The 4Suite package ...
```

Slide 63: Exercise: Querying a package file for information

```
$ rpm --query --list --package 4Suite-0.11.1-13.i386.rpm
/usr/bin/4odb
/usr/bin/4rdf
...
/usr/share/doc/4Suite-0.11.1/profile/Rdf
/usr/share/doc/4Suite-0.11.1/profile/Rdf/large_complete.py
```

Slide 64: Exercise: Querying a package file for a file listing

Querying a package file

We can make queries of a package file (as opposed to an installed package) with the `--package` option (slide 62). As the name, version and release are typically part of the name a simple query is rarely useful.

We can repeat the `--info` (slide 63) and `--list` (slide 64) options, however, which is a more typical use.

Note that in the `--info` case, the installation date is given as "(not installed)". This is a result of enquiring of a package file. Even if that package had been installed, if you ask of a package file you always get the "(not installed)" answer.

Digital signature warning

Red Hat "sign" their package files to prove that they are genuine, untampered with, package files from Red Hat Linux Inc. The warning says that the system noted that they was indeed a signature on the package file but it had no way to verify that it was genuinely Red Hat's signature, or anyone else's for that matter.

Red Hat do ship the file that confirms the signature, ready for incorporation into the package management system. But we now face the bootstrapping issue of whether or not to trust this file. We shall deal with this problem in the time-honoured fashion of ignoring it and just trust the file.

```
# rpm --import /usr/share/doc/rpm-4.2/RPM-GPG-KEY
```

Slide 65: Loading Red Hat's signature file

Loading Red Hat's signature file

The rpm program has a --import option, which we won't meet again, that loads this signature file. This operation need only ever be run once. Once the key is in the database it is there for good and we should not receive any more of these warnings.

3.3 Installing packages

Installing packages

Now we have a handle on RPM files, we can start adding them to our system. We now drop the --query option and replace it with the --install option.

While all the queries we have done to date could have been done by any user, installations must be done by the superuser.

As is typically the case on Unix systems, successful operation is completely silent, as shown on slide 66.

The package has been installed. Try it; it's very silly.

```
# rpm --install xsnow-1.42-10.i386.rpm
#
```

Slide 66: Installing a package

BUG WARNING AND WORKAROUND

There is an intermittent bug in the current versions of rpm which causes it to lock up under certain circumstances. Because it is intermittent it is very hard to pin down and fix. As a result, Red Hat have no patch for it yet. The symptoms are that commands which update the package database hang. Control-C does not kill them and nor does kill -TERM. This paragraph gives the workaround.

1. Identify the rpm process
2. Kill it
3. Change to the /var/lib/rpm directory
4. Run db_recover -c

```
# ps -ef | grep rpm
root      3853  3717  3 14:26 pts/0    00:00:00 rpm --install xsnow-1.42-10.i386.rpm
root      3855  3717  0 14:26 pts/0    00:00:00 grep rpm
# kill -KILL 3853
# cd /var/lib/rpm
# db_recover -c
```

Package dependencies

Now we'll examine a failed attempt to install a package. Suppose we want to install the DVI previewing program ("xdvi"). This comes in a package tetex-xdvi-1.0.7-66.

However, an attempt to install it fails with a huge "failed dependencies" error (slide 67). The first line is thae statement of an explicit package-on-package dependency. To use the `tetex-xdvi` package at version `1.0.7` you must have the `tetex-fonts` package at the same version. This is followed by a long list of shared libraries the package needs. One of the features of the Red Hat package management system is that when executable binaries are packaged up the packaging program determines what libraries those programs need. If the libraries aren't in the package itself it records their names as "dependencies". If the dependencies aren't satisfied (i.e. if the libraries aren't present) then the package won't be installed.

It is possible to override `rpm` by crushing its objections underfoot. There is a `--nodeps` suboption which means "ignore the dependency problems". We're not going to use this until later. In the mean time we will resolve the dependencies.

```
# rpm --query tetex-xdvi
package tetex-xdvi is not installed

# rpm --install tetex-xdvi-1.0.7-66.i386.rpm
error: failed dependencies:
        libmd5.so.0   is needed by tetex-xdvi-1.0.7-66
        libwwwapp.so.0   is needed by tetex-xdvi-1.0.7-66
        libwwwcache.so.0   is needed by tetex-xdvi-1.0.7-66
        ...
        tetex-fonts = 1.0.7 is needed by tetex-xdvi-1.0.7-66
# rpm --query tetex-xdvi
package tetex-xdvi is not installed
```

Slide 67: Failing to install `tetex-xdvi-1.0.7-66`

Looking for the libraries manually

Packages containing libraries are often named after their principal library, so we look for these names in the set of package files (slide 68).

Of these, the package file `w3c-libwww-5.4.0-4.i386.rpm` looks most promising so we examine the list of its contents (slide 69). We find that it actually contains *all* the libraries the `tetex-xdvi` package needs. This is not uncommon; if a set of libraries are typically used together it's sensible to combine them in a single package. The real question is why the libraries aren't in the program's package. The reason is that other programs could use these libraries. For example, all the programs in the package file `w3c-libwww-apps-5.4.0-4.i386.rpm` use them too.

```
# ls *libwww*
perl-libwww-perl-5.65-6.noarch.rpm
w3c-libwww-5.4.0-4.i386.rpm
w3c-libwww-apps-5.4.0-4.i386.rpm
w3c-libwww-devel-5.4.0-4.i386.rpm
```

Slide 68: Looking for the `libwww` libraries

```
# rpm --query --list --package w3c-libwww-5.4.0-4.i386.rpm
/usr/lib/libmd5.so.0
/usr/lib/libmd5.so.0.1.0
/usr/lib/libwwwapp.so.0
/usr/lib/libwwwapp.so.0.1.0
/usr/lib/libwwwcache.so.0
/usr/lib/libwwwcache.so.0.1.0
...
```

Slide 69: Finding the `libwww` libraries

- `rpmdb-redhat`
- A copy of the RPM database for *all* packages from the Red Hat Linux distribution
- Lets `rpm` know about uninstalled packages *from Red Hat Linux*
- Causes the `rpm` to offer suggested resolutions to dependencies

```
# rpm --install rpmdb-redhat-9-0.20030313.i386.rpm
```

Slide 70: The RPM database package

The RPM database package

Searching manually is a bit hit-and-miss and there is a better way. Red Hat ship a database of *all* their packages and what they contain. This allows the `rpm` system to recommend other uninstalled packages to resolve dependencies.

Note that this will only work for packages from Red Hat. Third party packages will not benefit from this.

```
# rpm --query tetex-xdvi
package tetex-xdvi is not installed

# rpm --install tetex-xdvi-1.0.7-66.i386.rpm
error: failed dependencies:
        libmd5.so.0   is needed by tetex-xdvi-1.0.7-66
        libwwwapp.so.0   is needed by tetex-xdvi-1.0.7-66
        libwwwcache.so.0   is needed by tetex-xdvi-1.0.7-66
        ...
        tetex-fonts = 1.0.7 is needed by tetex-xdvi-1.0.7-66
    Suggested resolutions:
        tetex-fonts-1.0.7-66.i386.rpm
        w3c-libwww-5.4.0-4.i386.rpm
# rpm --query tetex-xdvi
package tetex-xdvi is not installed
```

Slide 71: Failing to install `tetex-xdvi-1.0.7-66` again

Successful installation

Now we need to resolve the dependency. We could, if we wanted to, first install the `tetex-fonts` and `w3c-libwww` packages (so long as they, in turn, have no unresolved dependencies) and then the `tetex-xdvi` package. In practice however, we don't need to do this, and a more typical use is to install all three packages at once. Note that it does not matter in what order the package files appear on the command line.

```
# rpm --query tetex-xdvi
package tetex-xdvi is not installed

# rpm --install tetex-xdvi-1.0.7-66.i386.rpm \
  tetex-fonts-1.0.7-66.i386.rpm w3c-libwww-5.4.0-4.i386.rpm

# rpm --query tetex-xdvi
tetex-xdvi-1.0.7-66
```

Slide 72: Successful installation of `tetex-xdvi-1.0.7-66`

3.4 Dependencies

Introduction to dependencies

Now we've seen the concept of dependencies in play, let's look at them in more detail. Clearly packages can require certain properties of the system and other packages can provide those properties when they are installed.

- Requirement
- Provision

Slide 73: Concepts of dependencies

Requirements

We will start by considering what requirements a package may have.

Shared libraries: We have already seen that a package may rely on some shared libraries for its programs to work. This is easy for the package system to cope with. When a package is being assembled, all the programs involved are listed and checked for their dependencies on shared libraries. Any shared libraries that are not provided by the package itself are added to a list also contained in the package. When the package comes to be installed a check is done to see if the required shared libraries are provided either on the system already or within a package being added at the same time.

Shells: If a package contains one or more shell scripts then it is easy to extract from them a list of the required shells at package creation time. The concept of "shell" can be taken quite loosely here to mean anything that follows the "`#!`" on the first line: `/bin/sh`, `/usr/bin/perl` etc. Again, when a package comes to be installed a check is done to see if the required shells are already installed or are being installed at the same time.

Packages: A package may require the system to have some functionality that is provided by another package. It is possible to add a requirement on another package at package creation time. For example, a number of different packages might want a set of commonly used GIFs that appear in web pages ("previous page", "next page", "up" etc.) so could explicitly depend on the package ("`indexhtml`") that provides them. Note that this is much cleaner than listing a dependency on individual files. It is also much closer to the model of treating packages rather than files as the units of installation.

Facilities: Suppose a package requires a mail transport agent (MTA) on the system for its programs to work properly. It could contain an explicit dependency on the `sendmail` package. However, there are MTAs other than `sendmail`; `exim` and `qmail` are two common examples. Instead of depending on an explicit package, it is possible to depend on a "facility" provided by a package. The `sendmail` package provides a facility called "`smtpdaemon`" as should the `exim` and `qmail` packages. The package that requires there to be an MTA can then depend on the facility `smtpdaemon` and have its dependency satisfied by any one of the three packages providing it.

Files: The most primitive dependency of all is a dependency on an explicit, individual file. This is deprecated and dying out. However, the package that requires an SMTP daemon might require it to be located at `/usr/lib/sendmail`. Under these circumstances a dependency on that file name might be reasonable.

Red Hat want to move towards libraries, shells and facilities as the used dependencies. Unfortunately they started with just packages. This has led to a large number of older packages that only quote their package dependencies. Facilities and packages both live in the same "namespace" as a consequence and many alternative MTAs offer the facility "`sendmail`" as a workround for those packages that depend explicitly on the package "`sendmail`".

- Shared libraries
- Shells
- Packages
- Facilities
- Files

Slide 74: Possible requirements

Determining dependencies

We need to be able to determine what dependencies a package has *before* we attempt to install it. The `--query` has a `--requires` suboption that provides precisely this function.

```
$ rpm --query --requires --package ruby-1.6.8-5.i386.rpm
libc.so.6
libc.so.6(GLIBC_2.0)
libcrypt.so.1
libdl.so.2
libm.so.6
libruby.so.1.6
rpmlib(CompressedFileNames) <= 3.0.4-1
rpmlib(PayloadFilesHavePrefix) <= 4.0-1
ruby-libs = 1.6.8-5
```

Slide 75: Using `rpm --query --requires`

Depending on a particular version of a package

Note that if we look at the requirements of the `ruby` package we see that it depends on package `ruby-libs` at version `1.6.8`, release `5`.

Offering facilities

So now we know what a package needs. We also need to know what it provides. For this, `rpm` offers the `--provides` suboption as the converse of `--requires`.

```
$ rpm --query --provides --package ruby-libs-1.6.8-5.i386.rpm
curses.so
dbm.so
digest.so
etc.so
...
sha2.so
socket.so
syslog.so
ruby-libs = 1.6.8-5
```

Slide 76: Using `rpm --query --provides`

- Install `MySQL-python-0.9.1-6.i386.rpm`
- You will need to install extra packages

N.B. Some of those extra packages will need extras of their own!

Slide 77: Exercise: Resolving dependencies

Source RPMs

The `rpm` program is also used to make packages, and the original source code, together with makefiles and configurations is stored in a "source RPM". It is possible for a single source RPM to generate more than one binary RPM and under these circumstances it is quite common for the libraries to be referred to by the name of the package containing them because it is being built at the same time as the program package depending on them.

3.5 Removing packages

Erasing packages

The term for removing a package with `rpm` is "erasing" and it has the `--erase` option to do this.

As ever, dependency checking may cause an erasure to fail. Just as we had packages fail to install because something they relied on was missing, we can have packages fail to erase because something relies on them. In the case of a dependency on shared libraries being provided by the package being erased both the library and the dependent package are listed.

```
# cd /root

# rpm --query xsnow
xsnow-1.42-10

# rpm --erase xsnow

# rpm --query xsnow
package xsnow is not installed
```

Slide 78: Erasing an installed package

```
# rpm --erase tetex-fonts
error: removing these packages would break dependencies:
        tetex-fonts = 1.0.7 is needed by tetex-xdvi-1.0.7-66
```

Slide 79: Failing to erase a package

```
# rpm --erase tetex-fonts tetex-xdvi
```

Slide 80: Successfully erasing packages

3.6 Overriding dependency checks

Why?

The system administrator is sometimes in the position of knowing better than the package management system just what needs to be done. Having dependency checks block package installation or erasure can be useful as a safety net but from time to time needs to be overridden.

For example, suppose we wanted to test out a new set of `tetex-fonts` fonts. We really just want to erase the `tetex-fonts` package and install our own test fonts (which probably aren't even packaged at the moment).

The `rpm` program has the suboption `--nodeps` to turn off dependency checking on installation and erasure (and upgrades which we will see later).

Anecdote

For example: There was a printing system that had got screwed up. Erasing the `lpr` package and reinstalling it worked fine. I didn't want to bother with reinstalling the dependent packages which were all OK.

```
# rpm --erase ghostscript-fonts
error: Failed dependencies:
  ghostscript-fonts is needed by (installed) ghostscript-7.05-32
  ghostscript-fonts is needed by (installed) gnome-print-0.37-4
  ghostscript-fonts is needed by (installed) libgnomeprint22-2.2.1.1-3
  ghostscript-fonts is needed by (installed) libgnomeprint-1.116.0-6
# rpm --erase --nodeps ghostscript-fonts

# rpm --query ghostscript-fonts
package ghostscript-fonts is not installed
# cd /mnt/redhat/9/en/os/i386/RedHat/RPMS
# rpm --install ghostscript-fonts-5.50-9.noarch.rpm
```

Slide 81: Erasing and then replacing a depended-on package

3.7 Upgrading packages

- New version of package
- Erase old and install new?
- Want to keep configuration
- RPM labels configuration files
- These are not changed by upgrades

Slide 82: What is upgrading?

What is upgrading?

An upgrade is subtly different from just erasing the old version of a package and installing a new one. A package contains a variety of types of files, some of which the system administrator is expected to change. These are configuration files and control how the system functions. If the old package were just erased then the configuration files would be lost and, on installation of the new package, they would have to be re-edited.

However, RPM packages have a flag in their databases that labels files as being configuration files. On upgrade these files are not replaced with the new package's default versions.

Other files that have been modified under the old package but which were not labelled as configuration files are saved in a file of the same name with `.rpmsave` appended.

```
# rpm --query --all | grep cups
qtcups-2.0-15
cups-libs-1.1.17-13
cups-1.1.17-13

# cd /redhat/updates/9/en/os/i386
# rpm --upgrade cups-1.1.17-13.3.0.3.i386.rpm \
                cups-libs-1.1.17-13.3.0.3.i386.rpm

# rpm --query --all | grep cups
qtcups-2.0-15
cups-1.1.17-13.3.0.3
cups-libs-1.1.17-13.3.0.3
```

Slide 83: A simple upgrade—CUPS

Why an upgrade might fail

Upgrades rarely fail. The most common reason for them to fail is that the dependencies have changed and an extra dependency has arisen in the new version of a package. A package can also depend on a specific version of another package, requiring that two or more packages be upgraded at the same time.

An alternative reason is that the system administrator is actually attempting a downgrade. The `--upgrade` option will normally reject an attempt to upgrade to an older version. There is an additional option, however, called `--oldpackage` that turns off this defence and allows a downgrade to occur.

- Upgrade may introduce bugs
- New version of software may be too different
- May want to illustrate old behaviour

Slide 84: Downgrading

```
# cd /redhat/9/en/os/i386/RedHat/RPMS
# rpm --query --all | grep cups
qtcups-2.0-15
cups-1.1.17-13.3.0.3
cups-libs-1.1.17-13.3.0.3

# rpm --upgrade cups-1.1.17-13.i386.rpm \
                cups-libs-1.1.17-13.i386.rpm
  package cups-libs-1.1.17-13.3.0.3 (which is newer than
    cups-libs-1.1.17-13) is already installed
  package cups-1.1.17-13.3.0.3 (which is newer than
    cups-1.1.17-13) is already installed
```

Slide 85: Failing to downgrade

```
# rpm --upgrade --oldpackage cups-1.1.17-13.i386.rpm \
             cups-libs-1.1.17-13.i386.rpm
#
```

Slide 86: Forcing a downgrade with `--oldpackage`

- Upgrades will do installs
- Freshening upgrades only if the package is installed
- Permits more use of wildcards

```
# rpm --upgrade cups-*
```

Slide 87: Freshening—upgrading only if necessary

Freshening—upgrading only if necessary

The `--upgrade` option has a dark side. If you upgrade a package file for a package that is not already installed then it will be installed. This is why we could not just run

```
# rpm --upgrade cups-*
```

and had to list the files we wanted to upgrade from. In fact, `--upgrade` can be used anywhere `--install` is, though not vice versa.

There is an alternative: `--freshen`. This option will run an upgrade so long as the package being upgraded is already installed.

```
# cd updates/9/en/os/i386
# rpm --freshen cups-*
```

Slide 88: Freshening `cups`

3.8 Checking a package

- Content corruption
- Permission or ownership changes
- Deletion

Slide 89: What can go wrong with a package?

- System administrator error
- Operating system error
- Hardware error
- Enemy action

Slide 90: Why can a package go wrong?

Packages going wrong

Every now and then a package will spontaneously break. Everyone with the relevant access will deny having changed anything but something has clearly changed.

The root cause of the change of behaviour can be a number of things. Sometimes the operating system itself gets confused. This can manifest itself in a whole variety of ways. However, more typically, a change has been made to one of the files within the package itself or a package it depends on. Either the data (file content) or metadata (file ownership, permissions etc.) gets changed or files just go missing. This can be due to system administrator finger trouble[2] or glitches in the system (hardware or software). The least common cause, but the one everyone dreads, is enemy action; a cracker has broken into your system and is doing serious damage.

For this reason, `rpm` provides a mode to check the integrity of installed files against the database. If files within the package have been edited or deleted then it will flag them. If it does not then you will need to look elsewhere, but can at least eliminate this package from your enquiries if you don't have a security breach.

In the case of a security breach this is not adequate on its own. Trojaned programs are often delivered as an RPM (see, I *told* you RPM was commonly used) and installed as a forced upgrade. Under these circumstances the RPM testing mechanism won't work because the (trojaned) files match the (trojaned) RPM database entries exactly.

```
# rpm --query --file /etc/xinetd.d/rsync
rsync-2.5.5-4
# rpm --verify rsync
# vi /etc/xinetd.d/rsync
# rpm --verify rsync
S.5....T c /etc/xinetd.d/rsync
```

S: The file has changed size.

5: The file's MD5 checksum has changed.

The contents of the file must have changed.

T: The timestamp on the file has changed.

c: It's a configuration file.

So all these changes are quite possibly innocent.

Slide 91: Verifying a package

[2]We're system administrators; we don't make mistakes. They're *system errors*.

The `--verify` option

The `rpm` program provides a mode, triggered by the `--verify` option, to test all the files within a package. Note that not all files belong in packages (e.g. `/etc/hosts`). The most common examples are PID files, lock files and log files created by programs as they run rather than at installation.

There is also a `--file` suboption. This does not check just the integrity of the file but of the package the file belongs to. If you have plenty of time there is also a `--all` suboption which tests the integrity of every installed package.

However, bear in mind that there can be well over a hundred files in a system that do not belong to a package. These are mainly log files or files created by programs on the fly.

Verification codes

```
SM5DLUGT c filename
```

Each character indicates a mismatch between the RPM database and reality. If there is no mismatch then a full stop is printed instead. Files can also be flagged as “missing” and warnings are also generated for unfulfilled dependencies.

S: Size

M: Mode (permissions)

5: MD5 checksum

D: Device major and minor numbers

L: symbolic Link target

U: owning User

G: owning Group

T: modification Time

c: Configuration file

Slide 92: Verification codes

- Checking a file

```
# rpm --verify --file /etc/xinetd.d/rsync
S.5....T c /etc/xinetd.d/telnet
```

- Actually checking the package of a file

```
# chmod u-w /usr/bin/rsync
# rpm --verify --file /etc/xinetd.d/rsync
S.5....T c /etc/xinetd.d/rsync
.M......   /usr/bin/rsync
```

Slide 93: Verifying a file's package

```
# rpm --verify --all
S.5....T c /etc/printcap
S.5....T c /etc/hotplug/usb.usermap
S.5....T c /etc/sysconfig/pcmcia
...
```

Slide 94: Verifying everything

3.9 Postinstall scripts

- Not everything can be done with files
- Run a script after installation
- In general: (post/pre)-(install/erase) scripts
- Example:

```
# tail -2 /etc/passwd
ntp:x:38:38::/etc/ntp:/sbin/nologin
gdm:x:42:42::/var/gdm:/sbin/nologin

# cd /redhat/9/en/os/i386/RedHat/RPMS
# rpm --install httpd-2.0.40-21.i386.rpm
# tail -2 /etc/passwd
gdm:x:42:42::/var/gdm:/sbin/nologin
apache:x:48:48:Apache:/var/www:/sbin/nologin
```

Slide 95: Not just a collection of files

Why do we need postinstall scripts?

Not everything associated with a package can be shipped as files. For a good example, consider the `httpd` daemon. It needs to have a user to run as. The `httpd` package can't just ship an `/etc/passwd` file and overwrite the existing one; it has to run the `useradd` command somehow if the userid it wants doesn't already exist.

This is where postinstall scripts come in.

- Package: `httpd-2.0.40-21`
- Location: `/redhat/9/en/os/i386/RedHat/RPMS/httpd-2.0.40-21.i386.rpm`
- Postinstall: Add the `apache` user.

Slide 96: Postinstall script example

3.10 Miscellaneous notes

- Documentation components
- Multiple versions of kernel packages

Slide 97: Miscellaneous notes

Documentation

It is standard practice with RPMs to include the documentation for a package (other than its manual pages) in the directory `/usr/share/doc/`*`package-M-N`*`/`. Some packages have extensive documentation in this directory, others just have the `README` file from the package's source code.

Multiple versions of the kernel

Normally, it would be impossible to have more than one version of a software package installed simultaneously. The files would clash, after all. However, in the case of the kernel, the version is built into the filenames. Installing multiple copies of the kernel (and being able to choose which to boot from) is covered by a later section of this course.

3.11 Building software from source code

- There is no RPM
- There is no RPM *yet*
- You want to tinker
- You have a better compiler

Slide 98: Why build from source code?

Why build from source code?

There is no RPM. Not all software comes as RPMs, though you might be surprised at just how much does. Take a look in the `/redhat/contrib/` directory to see just how much people other than at Red Hat have packaged up.

You want to tinker. If you know the relevant language (almost always C) you might want to play with the source code and build your own special version.

You have a better compiler. Perhaps your compiler is better than the one used by the packager. It may produce faster or less buggy machine code.

- `README`
- `INSTALL`
- `configure`

Slide 99: Bits of a source tree

Bits of a source tree

`README`: This is the first file to read. It typically describes what the program is and does. Sometimes it also contains instructions for building the program, but these are commonly stored in the following file.

`INSTALL`: How to build the program. With many modern programs, and all the programs that meet the FSF's programming standards there is a standard procedure. The procedure is to run the `configure` script which builds a `Makefile`. Then you run `make` to build the software and `make install` to install it.

`configure`: This program determines all the options to the program, including its eventual installation location and defaults. It is a large and complex shell script but the set of available options is printed out with "`./configure --help`".

In the Red Hat model, non-packaged software should be installed in `/usr/local` and not in the main `/bin`, `/usr` or `/etc` directories. This is typically the default for `configure`.

- Mount the GNU distributions as `/gnu`
- Unpack `/gnu/nano/nano-1.0.9.tar.gz` in `/var/tmp`
- Configure it
- Build it
- Install it
- Full details in notes

Slide 100: Exercise: Building `nano`—GNU `pico`

Detail for building `nano`

Do the mounting as `root`:

```
# mkdir /gnu
# mount -o ro nfs-uxsup.csx.cam.ac.uk:/gnu /gnu
```

Do the building as an ordinary user:

```
$ cd /var/tmp
$ ls /gnu/nano
$ tar -xzvf /gnu/nano/nano-1.2.1.tar.gz
nano-1.2.1/
nano-1.2.1/po/
nano-1.2.1/po/Makefile.in.in
nano-1.2.1/po/Makevars
...
$ cd nano-1.2.1
$ ./configure
checking for a BSD-compatible install... /usr/bin/install -c
checking whether build environment is sane... yes
checking for gawk... gawk
...
$ make
make  all-recursive
make[1]: Entering directory '/var/tmp/nano-1.2.1'
Making all in po
...
```

And do the installation as `root`:

```
# cd /var/tmp/nano-1.2.1
# make install
Making install in po
make[1]: Entering directory '/var/tmp/nano-1.2.1/po'
...
```

And as an ordinary user, try out the new editor:

```
$ nano
```

Comments

In this example, we copy the source tree over from the Unix Support NFS server. Don't worry about the `mount` magic; it will be explained later in the course. The source tree has been sorted as a single file by the `tar` program (hence the `.tar` suffix) and the single file has been compressed with the `gzip` program (hence the `.gz` suffix). It can be unpacked with `tar` on its own as it knows how to handle compressed files. We then enter the source tree. Note that only the mounting and installation phase need be done as `root`. All the building can, and therefore should, be done as an ordinary user.

- Red Hat doesn't ship with a typing tutor
- Build and install GNU `gtypist` version 2.7 (in `/usr/local`)

Slide 101: Exercise: Building `gtypist`

More practice

Feel free to build other programs from the `/gnu/` directory. Please keep the installations to `/usr/local`. You may well find that they don't all build cleanly.

4 Boot sequence

4.1 Introduction

- BIOS
- GRUB
- Kernel startup
- `init`
- `rc` scripts

Slide 102: System startup overview

System startup overview

This section aims to give you a working knowledge of what goes on between your pressing the power switch on the front of your computer and its being a fully-functional Web server (or whatever you're planning on using it as).

In an ideal world, of course, most of what I'm going to tell you should be unnecessary. BIOSes and bootloaders, at least, should only be of interest to OS hackers and people trying to be clever. In this universe, though, and especially on a platform as baroque as a PC clone, a rough understanding of how things work is more-or-less essential.

4.2 BIOS – the Basic Input/Output System

Introduction

The BIOS of a PC is a piece of software that's built-in to the machine, usually by being stored in a flash ROM on the motherboard.

The BIOS has effectively two purposes. The first is the one we're directly concerned with here: taking the machine from its initial power-on state to the point where it can run GRUB. The second purpose, and the source of the BIOS's name, is to provide a set of rudimentary device drivers for some of the hardware in the machine, so that bootloaders, and OSes that don't know any better, have slightly easier lives.

- Set up hardware (motherboard and expansion cards)
- Test hardware
- Start bootloader
- Provide services to bootloader and OS

Slide 103: BIOS responsibilities

BIOS startup

I'm not going to cover what the BIOS does at startup in detail, on the grounds that this is a course about Linux, and not PCs in general. Nonetheless, there follows a brief description of the subject.

The first responsibility of the BIOS is to set up various aspects of the motherboard chipset so that the machine will behave like a normal PC. Modern PCs have a reasonably flexible and sensible hardware architecture, but when the operating system starts, it expects the system to look more-or-less like it belongs in 1983. This includes things like setting the PCI interrupt routing to look like an ISA bus, turning on the A20 gate (so the machine appears to have 1 Mb of memory) and so forth. When Linux starts, it will, of course, reverse a fair chunk of this brain-damage, but it's all necessary for backward compatibility.

Having set up the motherboard, the BIOS will call upon the BIOS on any expansion cards to initialise. Usually this only means the video card, which sets itself up to look like a VGA card, and provides a few software routines for writing to the screen (The original PC BIOS predated VGA, so it needed a separate BIOS).

- BIOS stores information in non-volatile RAM
- Configuration program built into BIOS

Slide 104: BIOS configuration

BIOS configuration

The BIOS stores the information it needs for starting up in battery-backed-up memory on the motherboard, informally referred to as "CMOS". The contents of this memory is what the "BIOS setup" program (usually built into the BIOS these days) modifies.

The Debian GNU/Linux installation manual[3] has some hints on BIOS configuration for running Linux.

BIOS booting

The next phase of the BIOS's job is to start up the operating system. How it does this is slightly variable, but it usually consists of looking around the system for a bootable disc, loading one sector (512 bytes), known as the "boot sector" from it into a particular address in memory and starting execution there. Historically, the disc chosen was the first floppy drive (if it had a disc in it), or the first hard disc if that failed. Modern BIOSes have support for a wider range of boot devices (notably including CD-ROMs), and for disabling certain devices (so the students can't boot the machines from floppy).

You might think that the BIOS's job is finished once it loads the boot sector, but it isn't. Because the boot sector (like bootloaders in general) is so small, it doesn't contain any code to control the PC's hardware. Instead, the BIOS provides facilities for using its code (which already has to exist to get this far) to access the hardware. Some primitive operating systems (notably MS-DOS) always use these BIOS drivers.

- Interfaces grow less quickly than hardware
- Very old BIOSes have a 64 MB RAM limit (e.g. `mem=128M` may help)
- Very old BIOSes have a 1024-cylinder (8.4 GB) disc limit
- Old BIOSes have a 32 GB disc limit

Slide 105: BIOS limits

[3] `http://www.uk.debian.org/releases/woody/i386/ch-preparing#s3.8`

BIOS limits

Because the PC BIOS interface dates (ultimately) from 1983, it tends to contain assumptions about the scale of computer hardware which aren't valid any more. It seems that generally, when one of these limits is reached, PC manufacturers come up with a means of extending the interface which is good for a few more years, before that in turn runs out of steam.

The two main fields in which this is seen are in disc and memory sizes. While the PC BIOS has an interface for determining the size of system memory, it has an upper limit on the size it can report (apparently at 64 Mb). There are newer BIOS interfaces for determining memory size, but on old machines with lots of RAM it may be necessary to explicitly tell Linux how much memory you've got. The Linux BootPrompt-HowTo has some information on this[4].

The disc-size limitation is in a way more serious. The PC BIOS has, at various points, had limits on the size of disc it can address. As with memory, these limits have increased over the years, but a large disc in an old machine is liable to be problematic. This is a serious problem because the Linux bootloader has to use the BIOS to access the disc, so any part of the disc that will be used by the bootloader must be within the part of the disc the BIOS can address. As mentioned in the installation section, this is generally worked around by keeping a small partition (traditionally called `/boot`) that is within the BIOS-addressable range and contains everything necessary to load the kernel.

4.3 GNU GRUB – The Grand Unified Bootloader

- The GRand Unified Bootloader
- Ludicrously over-featured
- Can also use LILO

Slide 106: GNU GRUB

GNU GRUB – The Grand Unified Bootloader

The usual bootloader for Red Hat Linux 9 is GNU GRUB. Its job in our case is to copy the Linux kernel from disk into RAM and start it executing, though it's capable of far more than that.

- BIOS (or another loader) loads boot block containing `stage1`.
- `stage1` loads first block of `stage2`, whose location is burned in.
- First block of `stage2` loads rest of `stage2` using burned-in block list.
- `stage2` loads configuration file `/boot/grub/grub.conf`.
- `stage2` loads kernel.

Slide 107: Typical GRUB boot process

Typical GRUB boot process

Before delving into the complexities of how GRUB is configured, I'll describe how it works in the usual case. This

[4] `http://metalab.unc.edu/mdw/HOWTO/BootPrompt-HOWTO-3.html#ss3.3`

doesn't cover all the details of what GRUB can do, but it's a start.

GRUB comes in two parts, called `stage1` and `stage2`. `stage1` is a very small program which fits into a 512-byte disc sector with room to spare for partition tables and a small amount of configuration information. `stage2` is a lot bigger, and lives in a file in the file system. It handles the complexities of interacting with the user and actually loading Linux.

GRUB's `stage1` has to be loaded by something else. In the case of a default Red Hat installation, `stage1` is stored in the master boot record (MBR) of the hard disc, and is loaded by the BIOS. In some other systems, there's something else in the MBR, and it takes responsibility for loading GRUB's `stage1`. However the first stage gets loaded, it brings with it the rest of the relevant disc sector, which contains enough information to allow `stage1` to find `stage2`, and possibly the partition table for the disc.

Once `stage1` is loaded, it takes a look around, prints "`GRUB`" on the console and then loads the first block of `stage2`. The physical location of this block is recorded in `stage1` when it's installed. The reason for only loading the first block of `stage2` is that `stage1` is too small to contain code to load any more. This first block then contains code to load the rest of `stage2`, along with a list of the disk blocks to load it from.

`stage2` is the part of GRUB that does all the hard work. It reads its configuration file, `/boot/grub/grub.conf`, and uses this to generate a menu of possible things to boot. The default one of these will probably a Linux kernel, and after ten seconds it will load the kernel from the file name specified in `grub.conf`. Note that `stage2`, unlike the earlier stages, can read Linux filesystems directly, so it doesn't have to remember which disk blocks to load files from.

`grub.conf`

Let's take a look at the `/boot/grub/grub.conf` on a typical fresh Red Hat 9 system and see what it says.

```
# grub.conf generated by anaconda
#
# Note that you do not have to rerun grub after making changes to this file
# NOTICE:  You have a /boot partition.  This means that
#          all kernel and initrd paths are relative to /boot/, eg.
#          root (hd0,0)
#          kernel /vmlinuz-version ro root=/dev/hda6
#          initrd /initrd-version.img
#boot=/dev/hda
default=0
timeout=10
splashimage=(hd0,0)/grub/splash.xpm.gz
title Red Hat Linux (2.4.20-8)
        root (hd0,0)
        kernel /vmlinuz-2.4.20-8 ro root=LABEL=/ hdc=ide-scsi
        initrd /initrd-2.4.20-8.img
```

- `default=0`
  - Select the first option by default
- `timeout=10`
  - Boot the default option after ten seconds if nothing happens
- `splashimage=(hd0,0)/grub/splash.xpm.gz`
  - Load this image onto the background of the display
  - `(hd0,0)` is what GRUB calls `/dev/hda1`, which is `/boot`

Slide 108: `grub.conf`, global settings

`grub.conf`, global settings

`default=0 timeout=10 splashimage=(hd0,0)/grub/splash.xpm.gz`

These lines control the overall behaviour of GRUB. They tell it to display a pretty picture behind the main menu, to use the first entry in the menu as the default, and to choose that option if the user doesn't press a key within ten seconds.

- `title Red Hat Linux (2.4.20-8)`
- `        root (hd0,0)`
  - Sets *GRUB's* root partition
- `        kernel /vmlinuz-2.4.20-8 ro root=LABEL=/ hdc=ide-scsi`
- `        initrd /initrd-2.4.20-8.img`

Slide 109: `grub.conf`, menu entries

`grub.conf`, menu entries

The remainder of `grub.conf` consists of a list of menu entries for GRUB to offer and for each entry, a list of commands that GRUB should execute when that entry is chosen. In the default Red Hat case, there's only one suchg entry, and it's labelled "`Red Hat Linux (2.4.20-8)`".

"`root (hd0,0)`" set's GRUB's idea of the root partition. This is distinct from Linux's idea of the root partition, and in this case the partition it refers to is actually `/dev/hda1`, which Linux mounts on `/boot`. All of the filenames that GRUB uses in the rest of the entry definition are relative to GRUB's root partition, and hence to `/boot` in Linux's view of the world.

The "`kernel`" command tells GRUB which operating system kernel to load. Since the filename is relative to GRUB's root directory, this actually means `/boot/vmlinuz-2.4.20-8`. The rest of the line specifies parameters to be passed to the kernel, telling it to mount its root directory read-only from the partition with a label of '`/`', and to use the `ide-scsi` driver to handle `/dev/hdc` (the CD-RW drive).

The "`initrd`" command tells GRUB to load an intial RAM disk to go along with the kernel. Again, the filename is relative to GRUB's root directory, and again, I'll explain what this is used for in a little while.

1. Edit grub.conf to change the initial timeout to five seconds.
2. Reboot to test your change.

Slide 110: Exercise: Modifying grub.conf

- Menu for choosing entries from grub.conf
- ‘c’ for a text prompt
- ‘e’ to edit commands for an existing entry
- ‘a’ to just add options

Slide 111: GRUB menu and prompt

The GRUB menu and prompt

At boot, GRUB presents a menu with entries from grub.conf, and selecting one of these is just a case of using the up and down arrow keys to select the right one and pression [RETURN]. To do anything more interesting, you’ll need to use one of the other options described at the bottom of the screen.

Pressing ‘c’ gives you a prompt at which it’s possible to type any of the commands that can appear in grub.conf. This is useful if you want to boot something completely different from what’s on the menus, but this is rarely the case. More likely, you’ll want to slightly vary one of the existing menu entries, by changing the filename to load the kernel from, or adding kernel parameters. You can edit the commands for the current menu entry by pressing ‘e’, which or you can add options to its kernel command using ‘a’.

- *n* Start in runlevel *n*
- single Start in single-user mode
- emergency Start in emergency mode
- init=/bin/sh if even emergency mode doesn’t work
- root=/dev/*xxx* Mount root from /dev/*xxx*

Slide 112: Useful kernel parameters

A few useful kernel parameters

Most kernel parameters are only useful in obscure circumstances, so you’re probably best off reading either the LILO manual or the documentation for the bit of the kernel you’re trying to change the behaviour of. The Linux BootPrompt-HowTo[5] is another good source. There are a few that are worth knowing for when the machine won’t boot normally, and these are listed on slide 112.

Supplying a number as a parameter (e.g. **2**) tells init which runlevel to change into on startup, overriding the initdefault setting in /etc/inittab. Supplying single or emergency will cause differing levels of single-user startup. Both of them should ask for a root password (for some reason, single doesn’t in Red Hat Linux) and

[5] http://metalab.unc.edu/mdw/HOWTO/BootPrompt-HOWTO.html

then dump you in a root shell – the difference is that `emergency` doesn't run `/etc/rc.d/rc.sysinit` first, so any non-root filesystems won't be mounted and the root filesystem will be mounted read-only, so running **`mount -a`** will be necessary before doing almost anything. In either case logging out of the root shell will cause the system to complete a normal multi-user startup.

Booting with `init=/bin/sh` takes single-user booting even further, and doesn't start `init` at all, and starts a root shell immediately without asking for a password. This isn't a good idea unless an emergency boot fails.

1. Reboot your machine
2. Press 'a' at the boot menu.
3. Try booting with one of the options given above, and see what it does.

Slide 113: Exercise: Kernel parameters

4.4 Kernel startup

- Emitted all the time, but mostly at startup
- Logged through `klogd` and `syslogd`
- Saved in `/var/log/messages` with everything else
- Last 8k or so accessible through `dmesg`(8)
- Boot messages saved in `/var/log/dmesg`

Slide 114: Kernel messages

Kernel messages

The Linux kernel is a very verbose beast, generating (on my test machine) 154 lines of output while starting up. Most of these can be ignored until you need them. The boot-time messages are merely a special case of kernel messages in general, which are logged by `klogd`. There will be more details on log files in the next section. It's possible to see the last few messages output by the kernel by running `dmesg`, but if the system's been running a long time, this may no longer include the boot messages. Because of this, Red Hat systems save a copy of the boot messages in `/var/log/dmesg`. Note, though, that this file won't contain any messages output during the later stages of system startup, notably those produced by loading Ethernet drivers.

1. Look at `/var/log/dmesg` in your favourite file viewer.
2. See if you can understand any of the messages.

Slide 115: Exercise: Looking at the boot messages

- specified by GRUB `initrd` command
- mounted as root and `/linuxrc` run
- loads kernel modules needed to mount root
- can appear at `/initrd` after boot

Slide 116: Initial RAM disc

Initial RAM disc

Once the kernel has finished starting up, it attempts to mount the root filesystem. In a normal Unix system, this would be the real root filesystem from disc, but Red Hat Linux has a bit of subtlety[6] here to allow it to load the kernel modules it needs to mount the root filesystem without them being compiled into the kernel. In our case, these are the modules necessary to use the Third Extended File System (ext3), which is the format used by Red Hat Linux for its root file system. It's also useful in cases where the root file system is on a SCSI disk, since each SCSI host adaptor needs a different driver, and building them all into the kernel would make it rather large[7]. When GRUB loads the kernel, it also loads the initial RAM disc, which is a file containing a small compressed filesystem. There's some magic in the kernel which temporarily mounts this as the root filesystem, runs `/linuxrc` from it, and only when that's finished mounts the real root filesystem.

If there's a directory called `/initrd` in the real root filesystem, the initial RAM disk gets remounted there when the real root is mounted. Red Hat's start-up scripts unmount it fairly early, but if you boot the system in `emergency` mode, you can take a look at it.

The initial RAM disk is created by `mkinitrd`(8), which automatically works out what to put in it based on your current kernel configuration. `mkinitrd` will be explained further in a later section of the course.

1. Reboot in `emergency` mode.
2. Look at the contents of the initial RAM disk in `/initrd`.

Slide 117: Exercise: Seeing the initial RAM disk

4.5 `init`

- First user process started (PID 1)
- Parent of all other processes
- Styled after System V `init`
- Configured through `/etc/inittab`

Slide 118: `init`(8)

`init`(8)

When `init` starts, it's the only process running (ignoring some processes running in the kernel), and has a process

[6] Actually, it's a horrible hack.

[7] Nonetheless, this is what Debian do.

ID of 1. If the system's going to do anything else, it's up to `init` to cause it to happen.

The version of `init` that Linux uses these days is modelled after the System V version, and differs from the older BSD version in many ways. This need not concern you unless you're likely to have to deal with BSD-like systems (or have come from a BSD environment), but it does at least explain why the RPM package containing `init` is called `SysVinit`.

- one of 0, 1, 2, 3, 4, 5, 6
- system state, characterised by a set of services
- maintained (and only known about) by `init`

Slide 119: Runlevels

Runlevels

`init`'s design is based around the concept of "runlevels". A runlevel is a state the system can be in, and is characterised by the set of system services that are available in that state. `init`'s main job is to handle changes in runlevel. This includes first entering a runlevel when the system starts, and switching to runlevel 0 or 6 just before it stops.

The runlevels available are numbered 0 through 6. 0 and 6 are special: entering runlevel 0 instructs the system to shut down, while entering runlevel 6 instructs it to reboot. Runlevel 1 is conventionally used for "single-user mode", in which the system has as little running as possible. This mode is useful for doing particularly awkward bits of reconfiguration (like shuffling filesystems between discs), which aren't really feasible with everything running. Runlevels 2 through 5 are available for your use, and will be explained a little more in Section 7.

- Contains all `init`'s configuration
- Line-oriented
- Third colon-separated field is the important one

Slide 120: `/etc/inittab`

/etc/inittab

`init`'s operation is entirely configured by the `/etc/inittab` file. In addition to comments (beginning with `#`) and blank lines, it contains lines with five fields separated by colons. In a piece of impressive obfuscation, the third field is the main one for determining the meaning of each line. The file format is fully documented in `inittab`(5).

The standard Red Hat `/etc/inittab` contains, amongst other junk, the following:

```
id:3:initdefault

si::sysinit:/etc/rc.d/rc.sysinit

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

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

x:5:respawn:/etc/X11/prefdm -nodaemon
```

`inittab` details

So, we've got five kinds of line, `initdefault`, `sysinit`, `wait`, `once` and `respawn`. `initdefault` indicates which runlevel `init` should switch to on startup. In this case, this is runlevel 3.

- Run at startup
- Mounts `/proc`
- Sets system clock
- Sets up keyboard and text screen
- Sets up USB devices
- Checks and mounts local filesystems
- Enables swap on partitions
- Enables quotas
- Cleans up
- Enables swapping on files

Slide 121: `/etc/rc.d/rc.sysinit`

`/etc/rc.d/rc.sysinit`

The `sysinit` line tells `init` to run the `/etc/rc.d/rc.sysinit` script when it first starts. This script does everything you see happening between the "`Welcome to Red Hat Linux`" and "`Entering runlevel: 3`" messages, notably setting the system clock, setting up the screen and keyboard, mounting and checking filesystems and enabling swapping.

`wait` entries and `rc`

The various entries labelled "`wait`" tell `init` what command to run when it enters a given runlevel. In this case, it runs `/etc/rc.d/rc` and passes it the runlevel on the command line. `rc`, not to be confused with the Plan 9 shell of the same name, is responsible for running the scripts that start and stop system services.

```
K20rstatd -> ../init.d/rstatd
...
S99linuxconf -> ../init.d/linuxconf
S99local -> ../rc.local
```

- `K`ill and `S`tart scripts
- Called in numerical order in each group
- Called with arguments `stop` and `start` respectively

Slide 122: `rcn.d` scripts

`rcn.d` scripts

These scripts are stored in various directories under `/etc/rc.d`[8], named `rc0.d` through `rc6.d`, one for each runlevel. Each of these directories contains a collection of symbolic links to files in `/etc/rc.d/init.d`, with slightly special names. For instance, on a newly-installed Red Hat system, `/etc/rc.d/rc2.d` contains:

```
K03rhnsd -> ../init.d/rhnsd
K05atd -> ../init.d/atd
K05saslauthd -> ../init.d/saslauthd
K20nfs -> ../init.d/nfs
K24irda -> ../init.d/irda
K44rawdevices -> ../init.d/rawdevices
K50xinetd -> ../init.d/xinetd
K72autofs -> ../init.d/autofs
K74ntpd -> ../init.d/ntpd
K75netfs -> ../init.d/netfs
K86nfslock -> ../init.d/nfslock
K87portmap -> ../init.d/portmap
K95firstboot -> ../init.d/firstboot
K95kudzu -> ../init.d/kudzu
S08iptables -> ../init.d/iptables
S09isdn -> ../init.d/isdn
S10network -> ../init.d/network
S12syslog -> ../init.d/syslog
S17keytable -> ../init.d/keytable
S20random -> ../init.d/random
S24pcmcia -> ../init.d/pcmcia
S26apmd -> ../init.d/apmd
S55sshd -> ../init.d/sshd
S80sendmail -> ../init.d/sendmail
S85gpm -> ../init.d/gpm
S90crond -> ../init.d/crond
S90cups -> ../init.d/cups
S90xfs -> ../init.d/xfs
S95anacron -> ../init.d/anacron
S99local -> ../rc.local
```

[8] On almost any other system using a System V `init`, they would be in `/etc`, as would `rc` and `init.d`

Each link's name begins with three magic characters. The first one is either `S` or `K`, indicating a service to be either started or killed when `rc` enters this runlevel. The next two characters indicate the order in which the scripts should be run. When `rc` enters a new runlevel, it first runs the `K`ill scripts in numerical order, then the `S`tart scripts in numerical order.[9]

Note that both `S` and `K` links point into the same directory. If you check the links for various other runlevels, you'll find that scripts pointed to by `K` links in some are pointed to by `S` links in others. In order that the scripts can tell whether they're meant to start or kill their respective services, `rc` passes each script an argument of `start` or `stop`, depending on what it wants to the script to do.

I shan't attempt to explain the precise purpose of all the `init.d` scripts. Most of them just start or stop a single daemon with the same name as the script. A few do special things, notably the network setup script `/etc/rc.d/init.d/network`, which will be explained in the network concepts section.

- Press [I] when you're told to
- `Start service` *foo* `(Y)es/(N)o/(C)ontinue? [Y]`
- [Y] – start the service
- [N] – don't start the service
- [C] – start this and all subsequent

Slide 123: Interactive startup

Interactive startup

Permenantly enabling and disabling services will be covered in the section on service configuration, but it's worth mentioning here the "Interactive startup" option. When a Red Hat Linux system starts, it displays a message:

```
Press 'I' to enter interactive startup.
```

If you do as it says (before the "`Entering non-interactive startup`" message), `rc` will prompt you before running each `S`tart script. Choosing [Y]es or [N]o at one of these prompts does the obvious thing. Choosing [C]ontinue causes `rc` to run the rest of the scripts without asking.

1. Reboot your machine
2. Press [I] at the right moment
3. Start your system up without Sendmail running

Slide 124: Exercise: Interactive startup

[9] There's an obvious bogosity here, in that there's no guarantee that the kill links in the new run level correspond to the start links in the old runlevel. The Right Thing would have been for the kill scripts to have been associated with the old runlevel, not the new one, but there's too much tradition around now to fix this, and it works by and large well enough.

- Run when wait entries are finished
- Restarted if/when they die
- mingetty provides console logins
- prefdm runs xdm, gdm or kdm, which provide X logins

Slide 125: respawn entries

respawn entries

The remaining interesting lines in inittab are those with the respawn action. These tell init to run the program specified at the end of the line once it's run all of the wait entries for the runlevel. When the program does, init immediately restarts it. All of the respawn entries for a runlevel are run at once.

The main use of respawn entries is to run interactive login programs. The first batch of six entries run mingetty on the first six virtual consoles. mingetty provides the login prompts you see on the consoles, and invokes login for you. The final one runs prefdm, which is (indirectly) responsible for automatically starting the X server and a graphical login program.

1. Edit /etc/inittab to add a mingetty on tty8
2. Ask init to re-read /etc/inittab:
   # **telinit q**
3. Check that you can now login on virtual console 8 (use [Ctrl] + [Alt] + [F8])
4. Remove the line from inittab and telinit q

Slide 126: Exercise: Changing the number of text consoles

More secure consoles

Note that if you want to enable logins on virtual consoles above tty11, you'll need to edit /etc/securetty to include them. The root user is only allowed to log in on terminals mentioned in this file (though only certain login mechanisms homour this), and by default it only contains tty1–tty11.

4.6 Changing runlevels

```
# runlevel
N 3
# telinit 2
#
```

Meanwhile, on the console:

```
INIT: Switching to runlevel: 2
Stopping atd:                                         [  OK  ]
Stopping xinetd:                                      [  OK  ]
Starting NFS statd:                                   [  OK  ]
Stopping portmapper:                                  [  OK  ]
Starting pcmcia:                                      [  OK  ]
```

Slide 127: `telinit` and `runlevel`

`telinit` and `runlevel`

Having discovered what different runlevels do, you may now wish to know how to get into ones other than the default. This is a fairly simple process: Just run (as root) `telinit`[10], giving it the runlevel you want to change to. The command `runlevel`[11] can be used to find out which runlevel the system's in (and which one it was in before the last change).

1. Kill off any existing X server.
2. Find out what runlevel your machine is currently in.
3. Change to runlevel 2. This stops various network services.
4. Change to runlevel 5. This starts the X server.
5. Change back to the original runlevel.
6. Change the default runlevel to 5 and reboot.

Slide 128: Exercise: Changing runlevel

Shutting down

A running Linux system contains quite a lot of state, so shutting it down is at least a slightly delicate process. At the very least, it's necessary to ensure that any important filesystem data in RAM has been written to disk. Some system processes also keep a fair amount of state which they'd like to save for the next time they're started, and on any large system, there are likely to be users who need to be warned that their system will be going away.

[10] or just `init`

[11] `runlevel` is a Linuxism. The nearest System V comes is `who -r`.

`/sbin/shutdown` [`-t` *sec*] [`-arkhncfF`] *time* [*warning-message*]

- Default is to go single-user
- `-h` Halt system
- `-r` Reboot system
- `-c` Cancel shutdown
- *time* When to shut down (e.g. `now`, *hh*`:`*mm*, +*mm*)

Slide 129: `shutdown`(8)

`shutdown`(8)

The usual command for shutting down a Linux system is `shutdown`. In addition to actually asking `init` to do the dirty work, `shutdown` also handles warning the users in advance, and shutting down at a pre-determined time, if this is the kind of thing you do.

`shutdown` takes a variety of flags, of which the most useful are `-h` and `-r`, which tell `shutdown` to halt or reboot the machine respectively. If neither of these is supplied, `shutdown` switches the system to single-user mode (runlevel 1), which isn't usually what you want. The first (non-option) argument to `shutdown` is the time at which to shut down. This can be specified either as a 24-hour time of day (like "`17:13`"), a number of minutes from now ("`+42`") or just "`now`", if you don't want to wait. Any remaining arguments are an excuse for `shutdown` to display in its warning message to the users.

After you've set off a shutdown, you can cancel it using `shutdown -c`, which does what you'd expect.

`/sbin/halt` [`-nwdfip`]
`/sbin/reboot` [`-nwdfi`]
`/sbin/poweroff` [`-nwdfi`]

- `halt -p` is equivalent to `poweroff`
- All other flags are specialised.
- These are the versions in `/sbin`, not `/usr/bin`.

```
WARNING: could not determine runlevel - doing soft halt
  (it's better to use shutdown instead of halt from the command line)
```

Slide 130: `halt`(8), `reboot`(8), `poweroff`(8)

`halt`, `reboot` and `poweroff`

These programs have a strange dual nature. Originally, and still if the system is in runlevel 0 or 6 (which usually means they've been called by `init`), they instruct the kernel to halt, reboot or power down the system. Obviously the last of these is only possible if your hardware and kernel support it. When called from a normal runlevel, they just invoke `shutdown` with appropriate arguments. `shutdown` then calls `telinit`, which causes `init` to switch to runlevel 0 or 6 and call `rc`, which ends up running `/etc/rc.d/rc0.d/S01halt` or `/etc/rc.d/rc6.d/S01reboot`, which finally calls `halt` or `reboot` again, this time in the correct runlevel.

Incidentally:

- `poweroff` and `halt` both call `shutdown` in the same way, and thus both end up turning off the power.
- If `halt`, `reboot` or `poweroff` can't determine what runlevel the system is in, they assume 0 or 6 and scold you for not using `shutdown` instead.
- There also exist versions of these programs in `/usr/bin` as part of the `usermode` package.

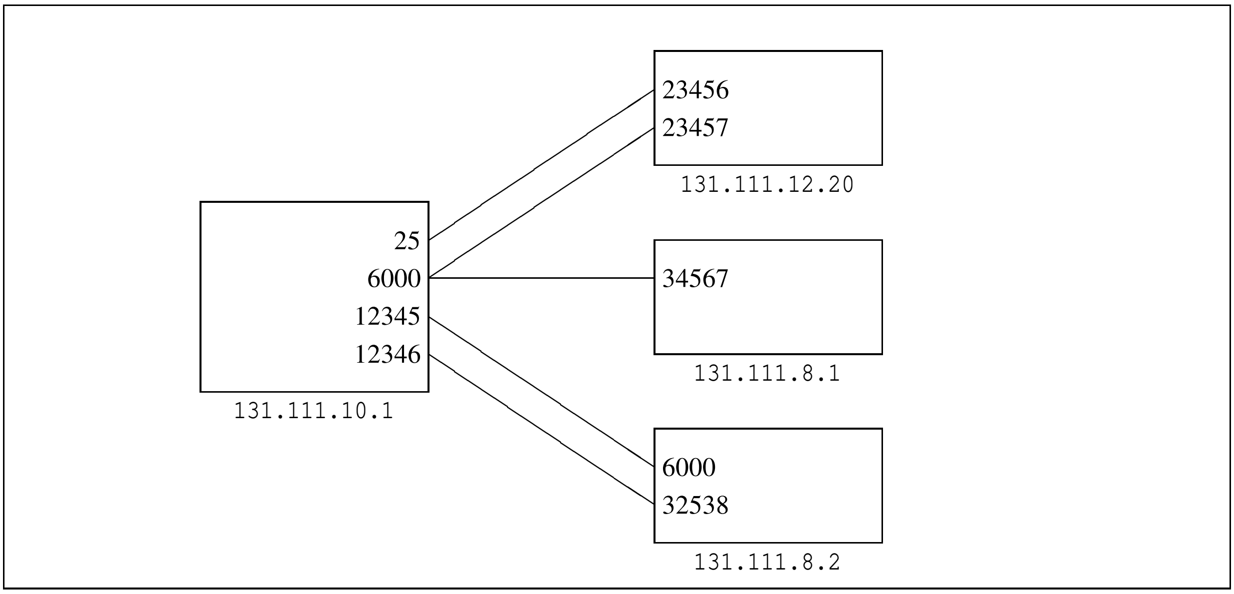
1. Reboot your machine using `shutdown`
2. Work out how to get `halt` not to run `shutdown`

Slide 131: Exercise: `shutdown` and friends

5 Network concepts

Network connections

This part of the course is designed to illustrate the networking aspect of the operating system: what the concepts are and how you set it up. It will discuss the Internet Protocol (IP) exclusively together with the lower level protocol it uses for local communication. Linux can speak other networking protocols: IPv6, Novell's IPX, Apple's EtherTalk etc., but IP is by far the most common and all we will talk about here.



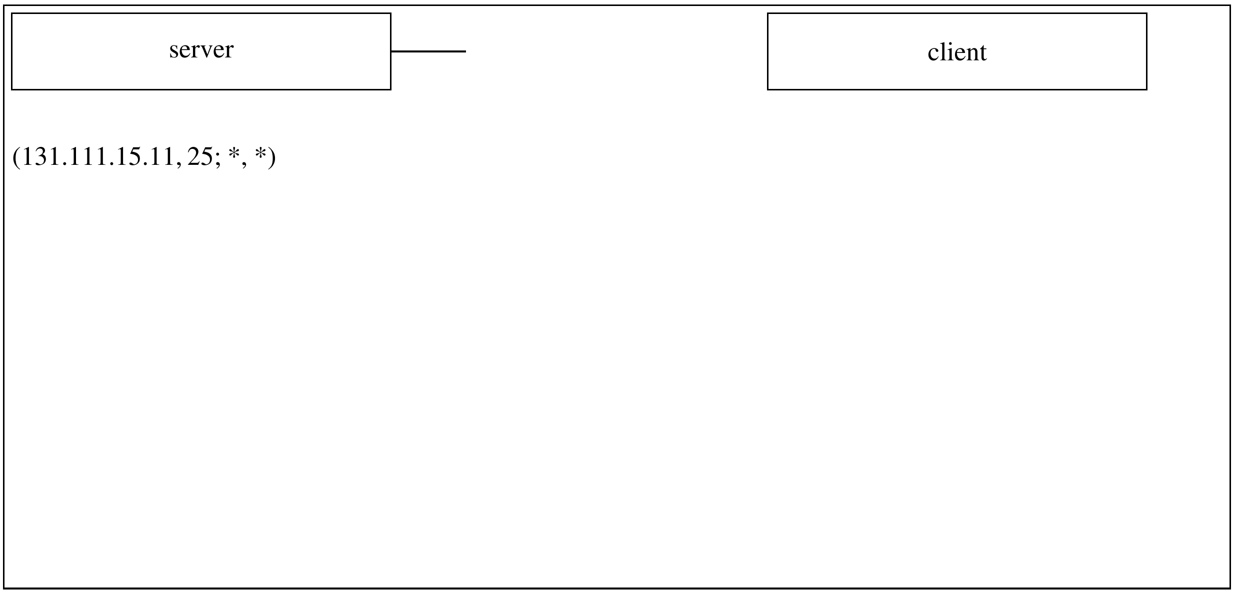
Slide 132: Internet network connections

IP networking

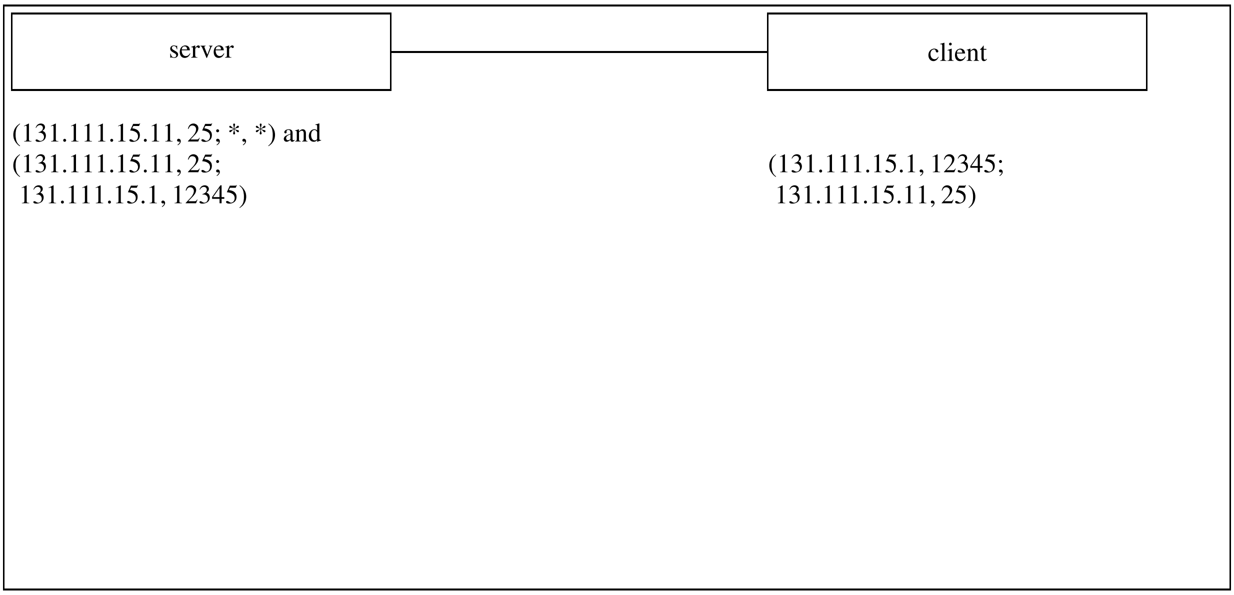
A presence on the Internet is identified by an "IP address". A good analogy is that the IP address of a machine is its phone number. It is possible for a machine to have multiple IP addresses but we won't discuss that in this course. We will deal only with the most common case of one network address per machine.

Along with a machine's phone number (IP address) comes a set of "port numbers" which are similarly analogous to telephone extension numbers. An IP network connection is uniquely defined by the "quad" of the IP address and port number at either end. Clearly the two machines at either end of the connection must agree about what ports are involved.

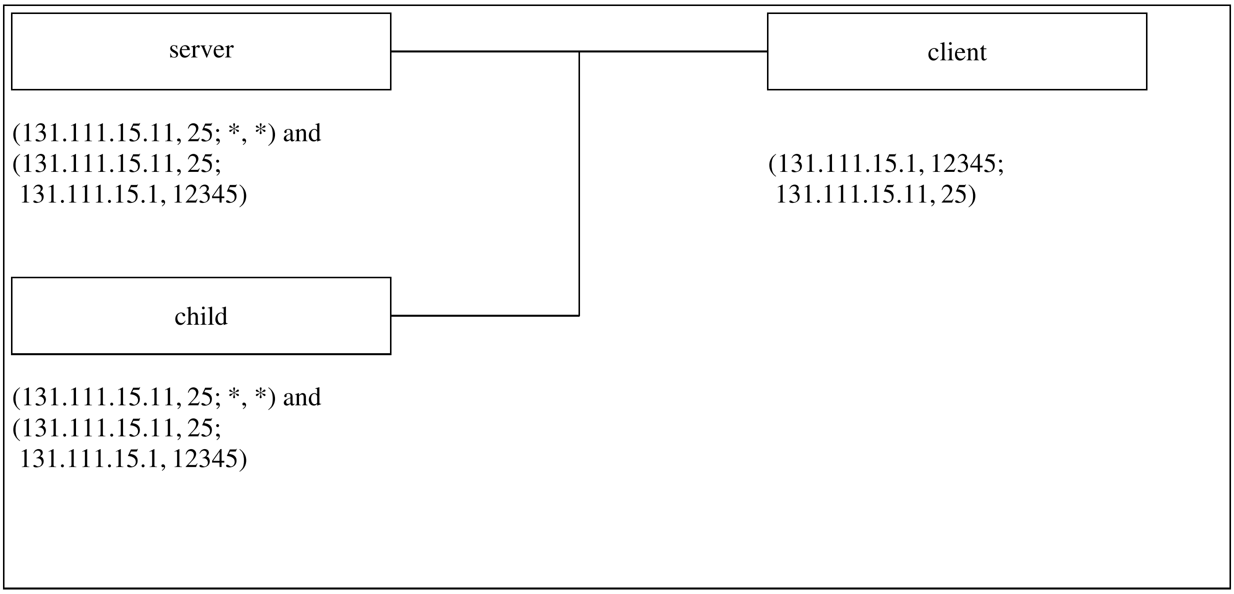
Note that a single port on a single IP address can conduct communications with more than one other end, so long as they have different (IP address, port number) pairs.



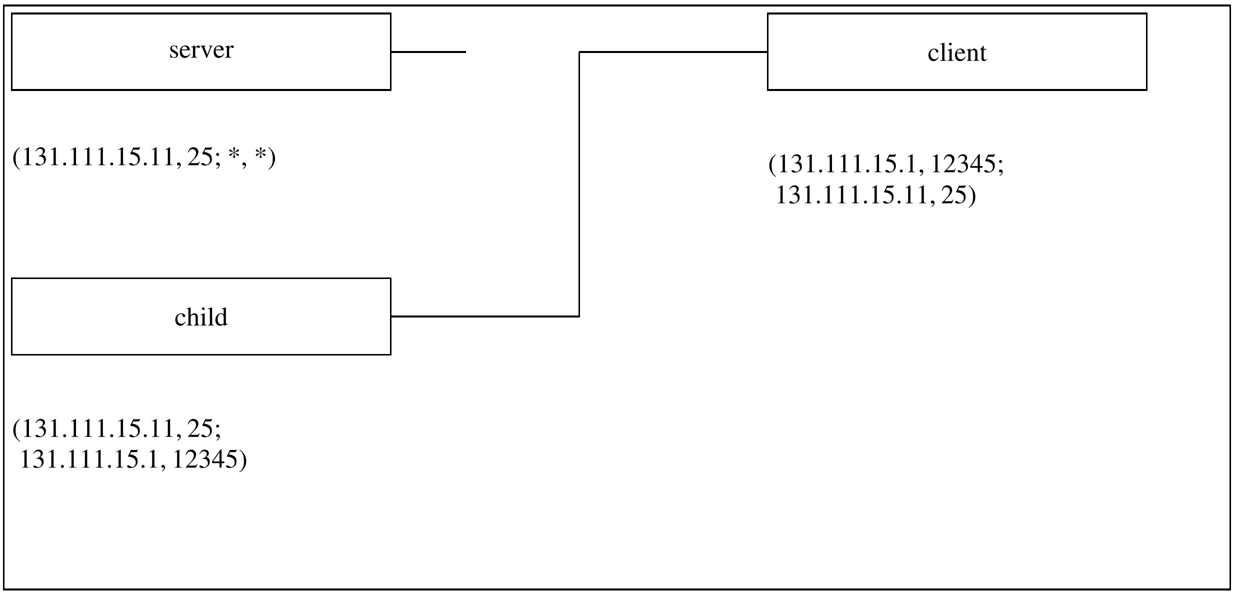
Slide 133: Setting up a network connection—1



Slide 134: Setting up a network connection—2



Slide 135: Setting up a network connection—3



Slide 136: Setting up a network connection—4

Network listeners

Network connections can be set up by prior agreement at both ends to determine port numbers. However, this is not what happens in practice[12] and the common way network connections are set up is (somewhat simplified) as follows.

[12] The only common exception occurs in the data transfer component of the file transfer protocol (FTP) which is widely derided for being an exception.

One system, the “server”, sets up something that is almost, but not quite, a network connection. The local ends are its own IP address and a port number, but the two remote elements of the quad are set to values that mean “anything”. No actual data is emitted by the machine at this point, but one end of a connection exists on a machine. This is called a “listener”. (Slide 133.)

Another system, the “client”, selects a (typically) random local port number and sets up a connection to the port on the server. Note that it has to know the server port number in advance and we will return to this issue soon. Because the server already has half of a connection established the connection attempt succeeds. Conversely, if the server had not been running a listener on that port the connection attempt would have been refused. (Slide 134.)

Once the client has established a connection the server has two quads in hand: the “half defined” listener quad and a fully defined quad carrying the server’s IP address and port and the client’s IP address and port. At this point the Unix process responsible for this listener “forks”. This means that it splits into two copies of itself that are identical save for one of them knowing it is the parent and the other knowing it is the child. No data should be sent from the client while the network connection is in this state. (Slide 135.)

The parent then closes the connection corresponding to the fully defined quad and the child closes the connection corresponding to the half defined quad. The parent then goes back to waiting for more incoming connections and the child carries on with the actual business of communicating with the client. The child server typically starts by sending some data to the client so that the client now knows it is safe to send data itself. Because the parent and child are separate processes, no amount of catastrophic failure in the child should be able to influence the behaviour of the parent’s listener. (Slide 136.)

TCP & UDP

I have been rather glib in my use of the expression “network connection”. Not all protocols use true connections. TCP (“transmission control protocol”) defines a full connection between the two machines. Delivery of data from one end of the connection to the other is guaranteed (by retransmission if necessary) and either end knows if the other has closed the connection. UDP (“user datagram protocol”) does not bother with the overhead of a connection. It consists simply of packets of data sent from one machine’s port to another with no guarantee of arrival. It is up to the user, or rather the user’s program, to cope with dropped packets. While TCP and UDP both use port numbers there is no overlap between them. The TCP ports and the UDP ports are completely unrelated.

All protocols that deal with large amounts of data use TCP because the guaranteed delivery becomes important and no protocol author wants to write their own retransmission algorithm when one already exists. However, there is a price for this facility; it takes more system resources to set up, maintain and tear down a TCP connection than it does simply to hurl a UDP packet into the void. UDP is typically limited to simple tasks within a subnet. It is very common for all UDP to be blocked at firewalls or other boundaries.

HTTP (the Web), FTP (file transfer) and SMTP (e-mail) are all examples of TCP protocols; they transmit potentially large amounts of data across a wide area network (WAN)—the entire Internet. DNS (IP name service) and NTP (time keeping) both use UDP because they exchange small amounts of data over typically short hops.

There do exist other protocols above IP. ICMP (Internet Control Message Protcol) is the best known “other”. This lacks the concept of port number and instead has a type number to determine what sort of control it is. The best known tool that uses this protocol is `ping` which sends the “ECHO_REQUEST” ICMP datagram and waits for the corresponding “ECHO_RESPONSE” datagram from the target machine.

```
pop2           109/tcp        pop-2   postoffice      # POP version 2
pop3           110/tcp        pop-3          # POP version 3
sunrpc         111/tcp        portmapper     # RPC 4.0 portmapper TCP
sunrpc         111/udp        portmapper     # RPC 4.0 portmapper UDP
auth           113/tcp        authentication tap ident
sftp           115/tcp
nntp           119/tcp        readnews untp  # USENET News Transfer Protocol
ntp            123/udp                       # Network Time Protocol
```

Slide 137: Extracts from `/etc/services`

Well known ports

This still leaves open the question of how the client knows to what port on the server it should attempt to connect. The "classic" Internet services listen on "well known ports". These are listed in the file `/etc/services`. The format of this file is fairly simple to understand. The first word on the line is the official name of the service. The second is the port and the protocol. Often there is both a TCP and UDP entry. The port has been assigned for both but only one is typically used. (The DNS is an exception to this; the UDP port is used for simple lookups and the TCP port for "zone transfers".) Next come any aliases for the service. The hash character ("#") introduces any comments which run to the end of the line. This file is used, for example, to determine that to make a TCP connection to the POP3 service on a server, connection should be made to TCP port 110.

The `portmapper`

Not all permanent network services listen on well known ports. Some use a different scheme where they listen on an arbitrary port and register this port with a service, called the `portmapper`, which does listen on a well known port (number 111). This scheme is used by the "RPC" (remote procedure call) services. When an RPC client wants to connect to an RPC service it first connects to the `portmapper` service, queries it for the port number of the service it is actually interested in and then connects to the service it wants. RPC services can be run over TCP or UDP and the portmapper listens for queries both by UDP and TCP.

The two most famous RPC services are NIS and NFS. Much will be said of NFS later in the course.

```
$ rpcinfo -p localhost
   program vers proto   port  service
    100000    2   tcp    111  portmapper
    100000    2   udp    111  portmapper
    100024    1   udp  32768  status
    100024    1   tcp  32768  status
$ rpcinfo -p thera.csi.cam.ac.uk
   program vers proto   port  service
    100000    3   tcp    111  portmapper
    100000    2   tcp    111  portmapper
 ...
    100024    1   udp  32776  status
    100024    1   tcp  32772  status
```

Slide 138: Using `rpcinfo`

`rpcinfo`

The `portmapper` service can be queried to find what services are registered using the `rpcinfo` program. Note that this program can be applied to remote sytems as well as the local one. Slide 138 gives an example of its output for a simple Red Hat system where the only registered RPC service is `portmapper` itself and of its output for a Solaris 2 system that makes rather more vigorous use of RPC. Note that the same RPC service can be offered over TCP and UDP and at a variety of versions too.

Examining network connections

The principal tools for examining the networking state of a system are `ifconfig` and `netstat`.

```
$ ifconfig
eth0  Link encap:Ethernet  HWaddr 00:A0:24:33:97:85
      inet addr:131.111.15.11  Bcast:131.111.15.255  Mask:255.255.255.0
      UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
      RX packets:2207385 errors:0 dropped:0 overruns:0 frame:0
      TX packets:53979 errors:0 dropped:0 overruns:0 carrier:0
      collisions:1 txqueuelen:100
      Interrupt:10 Base address:0x300

lo    Link encap:Local Loopback
      inet addr:127.0.0.1  Mask:255.0.0.0
      UP LOOPBACK RUNNING  MTU:3924  Metric:1
      RX packets:1663 errors:0 dropped:0 overruns:0 frame:0
      TX packets:1663 errors:0 dropped:0 overruns:0 carrier:0
      collisions:0 txqueuelen:0
```

Slide 139: Example `ifconfig` output

ifconfig

The `ifconfig` program details the state of the system's networking interfaces. The output of this command is mostly irrelevant at the level of this course. However, one line allows you to check the network setup against the parameters passed to you by the University IP Register. The IP address, broadcast address and netmask are all specified on one line in the output for the ethernet interface and should match what your machine was assigned.

```
# netstat --all --programs
Proto  Local Address  Foreign Address  State       PID/Program name
tcp    *:6011         *:*              LISTEN      7739/sshd
tcp    draig.csi:ssh  ghoul.csi:1009   ESTABLISHED 7739/sshd
tcp    draig.csi:6010 draig.csi:1025   ESTABLISHED 854/sshd
tcp    draig.csi:1025 draig.csi:6010   ESTABLISHED 858/xterm
tcp    *:6010         *:*              LISTEN      854/sshd
tcp    draig.csi:ssh  ghoul.csi:1015   ESTABLISHED 854/sshd
tcp    *:ssh          *:*              LISTEN      546/sshd
tcp    *:auth         *:*              LISTEN      441/
tcp    *:959          *:*              LISTEN      356/
tcp    *:sunrpc       *:*              LISTEN      304/portmap
udp    *:799          *:*                          -
udp    *:800          *:*                          -
```

Slide 140: Example `netstat` output (some columns dropped)

`netstat`

The standard Unix `netstat` command gives information about the network connections known by a system. The Linux version has some extensions for identifying what processes are responsible for them.

The command "`netstat --all --programs`" causes `netstat` to display all the network connection information it has (including some that we aren't interested in here) together with the process (PID) and program name responsible for it. If you run this as `root` then all processes that can be identified are shown. If it is run as a user then only that user's processes are identified.

The first column identifies the protocol in use. The second and third identify the four elements of the connection's quad. Having an asterisk for the local IP name means that the listener is listening on all interfaces. The fourth column identifies the status of a network connection. There is no status for UDP listeners. The last column (missing unless the `--programs` option is used) identifies the process ID and program responsible for the connection. Note that this cannot always be identified. If there is no PID then the network connection is due to a kernel module rather than an extrenal program. (The Network File System (NFS) is run from the kernel.)

Ethernet

We will now take a brief detour from pure IP to examine how IP is implmented on a typical network. The local area network[13] (LAN) is provided by ethernet. This acts, in its simplest incarnation, as a broadcast service with each machine shouting out loud and every other machine on the same ethernet being able to hear it.

It's possible, and likely on a busy LAN, that two machines may try to shout at the same time. This is called a "collision" and the procedure for dealing with it is that all colliding systems back off for a randomly chosen period of time before trying again. This retry is repeated with an increasing average amount of time before each retry.

The other consequence of this simple model is that each machine on the LAN will hear every other machine. Each packet starts by declaring which machine it is from and which it is to. Typically every machine will ignore packets not meant for it. This ignoring is typically done in the ethernet card itself to avoid wasting CPU in the operating system. It is possible to put an ethernet card in "promiscuous" mode where it will continue listening (and passing data up to the operating system) even if the message is not meant for it.

Because of the simplicity of the protocol and the small number of machines on the LAN (a few hundreds at most) it is sufficient to give each machine a unique "ethernet address". This is burnt into the ethernet card by the manufacturer and is *globally* unique but only locally useful. The ethernet address is six bytes long and is usually written as six hexadecimal numbers separated by colons, e.g. `08:00:20:11:E3:37`.

From LAN to WAN

The ethernet addressing scheme is fine for LAN work, but not for anything larger. While the ethernet address is globally unique, unless it is on the LAN there is no way to determine where a card is, given just its ethernet address.

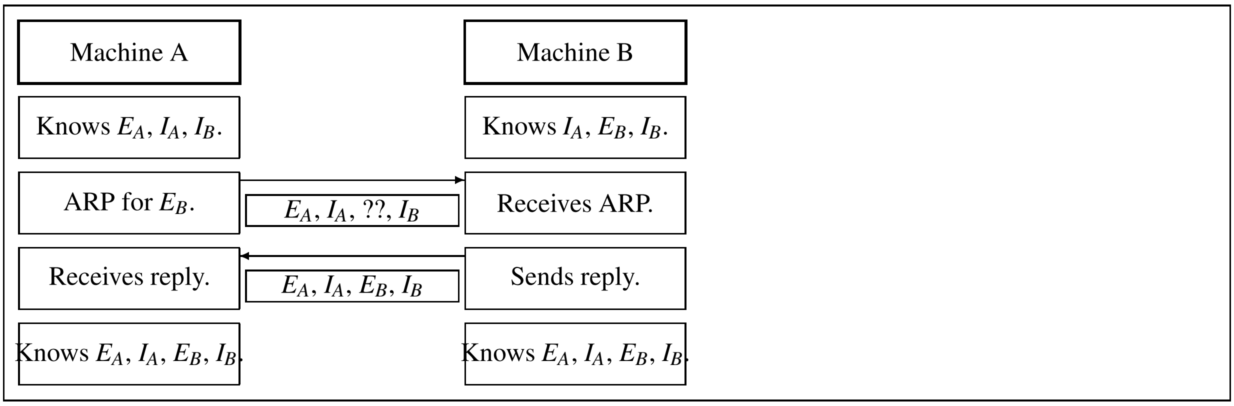
To get round this, we need a global addressing system where the addresses are specified in terms of what network the machine is on and not just when the ethernet card was made and in whose factory.

We do this by implementing the Internet Protocol above the ethernet. When a system communicates via IP over ethernet to another system on the same piece of ethernet it sends ethernet messages that start with the ethernet addresses of the source and target systems. These are followed by a flag saying that this ethernet packet is carrying IP data. This is followed by the source and target IP addresses and a flag saying what sort of IP connection it is (TCP, UDP, etc.).

[13] In our case the LAN is this room and the room downstairs.

ethernet target	`08:00:20:11:E3:37`
ethernet source	`00:90:92:7D:C0:00`
ethernet type	IP
IP source	`131.111.10.245`
IP target	`131.111.10.84`
IP type	TCP
TCP source port	12345
TCP destination port	21
DATA	

Slide 141: A TCP packet over ethernet (simplified)



Slide 142: The address resolution protocol

The address resolution protocol

We are starting with the assumption that we know the IP address of the target machine. How do we determine its ethernet address? This is handled by the address resolution protocol (ARP).

If the machine with ethernet address E_A and IP address I_A wants to communicate with the machine with IP address I_B it sends out an ethernet packet whose ethernet source address is E_A and whose target ethernet address is a "broadcast" address meaning that the packet is of concern to every machine on the ethernet. The ethernet type flag identifies it as an ARP packet and it carries the IP source and destination addresses as if it were a normal IP packet. When the machine with IP address I_B receives this packet, it sends back a similar packet but with its ethernet address, E_B, filled in and typically records the data it has derived about the initiating machine. Normal communication can now start.

The lookup from IP address to ethernet address is only kept for a few minutes after it is last used in a table called the "MAC[14] table" or the "ARP cache". To get at it directly use the `arp` command.

[14] medium access controller

```
$ arp -a
route-cent-8.cam.ac.uk (131.111.11.62) at 00:90:92:7D:C0:00 [ether] on eth0
griffin.csi.cam.ac.uk (131.111.10.84) at 08:00:20:11:E3:37 [ether] on eth0
ses002.csi.cam.ac.uk (131.111.11.158) at 00:01:30:43:DA:00 [ether] on eth0
mothra.csi.cam.ac.uk (131.111.11.156) at 08:00:20:CF:83:BE [ether] on eth0
cookie.csi.cam.ac.uk (131.111.11.77) at 00:C0:4F:68:12:D1 [ether] on eth0
```

Slide 143: Examining the ARP cache

Routing

Returning to the IP network, so far we have only considered networking from the perspective of the two ends. We can now discuss how the packets of data get from A to B.

If the two machines that want to communicate are on the same piece of ethernet then we have already seen how the ARP sets up communication between them. To contact a more remote system, though, some mechanism is required to get the data packets off the current network and onto another. This is called "routing".

```
$ netstat --numeric --route
Kernel IP routing table
Destination     Gateway         Genmask         Iface
131.111.15.0    0.0.0.0         255.255.254.0   eth0
127.0.0.0       0.0.0.0         255.0.0.0       lo
0.0.0.0         131.111.15.62   0.0.0.0         eth0
```

Slide 144: A system's routing table (some columns dropped)

The routing table

The IP component of a Unix kernel carries within it a "routing table". This is a lookup table that says "to get a packet to network X send it to address Y". Address Y (the router) must be a local address (i.e. one we can access without further routing). In Cambridge, which has a very simple network topology, the typical routing is for there to be a single "default" routing line which means that if the remote address is not on the local network then send it to the "default router". An IP packet for an IP address on network X is sent with the ethernet address for the router as its destination ethernet address. The router will receive this packet and then (hopefully) forward it. A LAN on the Internet will require one or more routers connecting it to the outside world.

The routing table in use on a system can be seen with the command "`netstat --numeric --route`" [15]. If we consider the routing table for a machine with IP address `131.111.15.11` shown in slide 144, we see that the second line refers to sending packets to itself via the internal "loopback" interface. In this case no explicit routing is required so the "gateway" field is left blank. The first and third lines refer to packets being sent to other systems.

[15] The `--numeric` option causes `netstat` to use numeric addresses rather than names in the output. For routing tables this is typically what you want.

- `131.111.8.42` masked by `255.255.255.255` gives `131.111.8.42`
- `131.111.8.42` masked by `255.255.255.0` gives `131.111.8.0`
- `131.111.8.42` masked by `255.255.254.0` gives `131.111.8.0`
- `131.111.8.42` masked by `255.255.0.0` gives `131.111.0.0`
- `131.111.8.42` masked by `255.0.0.0` gives `131.0.0.0`
- `131.111.8.42` masked by `0.0.0.0` gives `0.0.0.0`

Slide 145: Examples of masking

Line	Destination		Genmask		Result		Entry
1	131.111.8.42	&	255.255.254.0	→	131.111.8.0	≠	131.111.15.0
2	131.111.8.42	&	255.0.0.0	→	131.0.0.0	≠	127.0.0.0
3	131.111.8.42	&	0.0.0.0	→	0.0.0.0	=	0.0.0.0

Slide 146: Routing to `131.111.8.42`

The routing table

The principle for routing a packet is best shown by example. To understand the example you first need to understand "masking" as shown in slide 145. Now, suppose a packet was destined for `131.111.8.42`, and this is a DNS server so plenty of packets will be. The tests illustrated in slide 146 are carried out:

Similarly, for line 1 a genmask of `255.255.254.0` applied to `131.111.8.42` gives `131.111.8.0` which is not the same as the destination column entry of `131.111.15.0` and for line 2 a genmask of `255.0.0.0` applied to `131.111.8.42` gives `131.0.0.0` which is not the same as the destination entry of `127.0.0.0`. So both lines are skipped.

Finally, the genmask in line 3 is `0.0.0.0` which, when applied to the destination address `131.111.8.42` gives `0.0.0.0`. This *does* match the destination entry and hence this is the line of the routing table that is applied to the packet. Note that the genmask of `0.0.0.0` will cause this line to apply to *all* packets that it is applied to. This is the *default* rule.

If the gateway column entry is blank (i.e. `0.0.0.0`) then the packet is handled without any special routing and is sent out from the interface listed in the "iface" column. Interface `eth0` is the zeroth (i.e. first) ethernet interface and interface `lo` is the loopback, internal interface.

If the gateway is not blank then the packet, while still destined for its original address is sent to the gateway address by the ethernet system. The receiving system (hopefully a router) should know what to do with it.

Within the CUDN almost all routing tables look similar to this, as each departmental network has a single point of contact with the CS core routers. In these circumstances the default router is the only piece of extra information the operating system needs to know for complete Internet access. The default router address is added manually, typically at installation.

For more complex environments other mechanisms exist for building the routing tables. Routers can "advertise" themselves on the local networks detailing what they can route to, and a daemon called `routed` can listen for these advertisements and build a routing table dynamically from the information obtained.

Configuration programs

So far we have examined the concepts of networking and the tools to examine them once they have been set up. Finally, we are going to examine the tools used during set up. We need to configure two aspects of networking: the interface and the routing table.

The `ifconfig` program is used to configure the IP interface of the system and the `route` program is used to manipulate the routing table. In practice these commands are rarely run manually but are run at boot time based on information derived from configuration files.

Do not run these commands in the class!

```
# ifconfig eth0 netmask 255.255.255.0 131.111.15.96
# route add default gw 131.111.15.62 eth0
```

Slide 147: Examples of `ifconfig` and `route`

Boot time configuration

As mentioned in the section covering the boot sequence, the network setup script is `/etc/init.d/network`. For a typical machine with an Ethernet connection on the CUDN, using a fixed IP address, the relevant data files are `/etc/sysconfig/network`, which specifies the basic configuration, and the files in `/etc/sysconfig/network-scripts` relevant to the connection type, in this case `ifcfg-eth0` and `ifcfg-lo`.

`NETWORKING`	"`yes`" or "`no`"—Should networking even be turned on at boot?
`HOSTNAME`	This should be the fully qualified domain name (FQDN).
`GATEWAY`	This should be the default router for the network. It is used as the address in the call to `route` that sets up the default routing. (See slide 147.)
`GATEWAYDEV`	This should be the netwoek interface (almost always `eth0`) used for the default route. (See slide 147.
`NISDOMAIN`	Typically missing in a standalone system. Missing for the purposes of this course.

The setup script brings up the interfaces by calling `ifup` with the relevant interface name, and `ifup` in turn calls `ifconfig` and `route`. The file `/etc/sysconfig/network` is used for the networking aspects of the system itself and the `ifcfg-eth0` file is used for the interface specific parts.

```
NETWORKING=yes
HOSTNAME=pc801.oms.pwf.cam.ac.uk
GATEWAY=131.111.15.62
```

Slide 148: A typical `/etc/sysconfig/network`

```
DEVICE=eth0
BOOTPROTO=static
BROADCAST=131.111.15.255
IPADDR=131.111.15.11
NETMASK=255.255.254.0
NETWORK=131.111.14.0
ONBOOT=yes
```

Slide 149: A typical `/etc/sysconfig/network-scripts/ifcfg-eth0`

- The Domain Name Service
- Fully qualified domain names
- Types of record

Slide 150: The DNS

DNS names

What we have described so far is sufficient to understand networks so long as you are happy using IP addresses for systems. We should now indicate how we will add names to this setup.

The Domain Name Service (DNS) is a distributed database of name servers. However, at this stage we only need to understand how to be a DNS client, what information the DNS contains and how to get at it.

- Address records
- Fully Qualified Domain Name (FQDN)
- FQDN⟶IP Address

```
pcttr101.titan1.pwf.cam.ac.uk. 86400 IN A       128.232.253.1
```

Slide 151: A records

```
pcttr101.titan1.pwf.cam.ac.uk. 86400 IN A       128.232.253.1
```

pcttr101.titan1.pwf.cam.ac.uk.: Fully Qualified Domain Name

86400: How long this record should last

IN: It's an Internet address

A: It's an A record

128.232.253.1: The IP address

Slide 152: An A record

A records

The most important records that the DNS contains are the *address records*, or A records. These take a name and give an IP address. These are sufficient to introduce us to some of the peculiarities of the DNS.

We'll start by looking at a simple A record in the DNS.

pcttr101.titan1.pwf.cam.ac.uk.: The record contains a fully qualified domain name (FQDN). The DNS does not work with partial names. The mapping from the short machine name `pcttr101` into the trial FQDN `pcttr101.titan1.pwf.cam.ac.uk` must be done by the client; it is not done by the DNS itself. The dot on the end of the name is just part of the DNS' internal machinery indicating that the name is complete.

86400: The next record is a "time to live". This gives, in seconds, how long this record is valid after it is received. 86400 seconds is one day.

IN: The DNS can carry much more than just Internet information, but doesn't. It is entirely dominated by Internet records but this key shows its origins by marking this record as being an Internet record.

A: There are various types of Internet record in the DNS. The next field identifies which one this is. an "A" indicates that it is an address record.

128.232.253.1: The last field in the record gives the IP address corresponding to the FQDN on the Internet. We will see below how the DNS copes with names that have more than one IP address.

```
hermes.cam.ac.uk.       86400   IN      A       131.111.8.70
hermes.cam.ac.uk.       86400   IN      A       131.111.8.77
hermes.cam.ac.uk.       86400   IN      A       131.111.8.57
hermes.cam.ac.uk.       86400   IN      A       131.111.8.67
red.csi.cam.ac.uk.      86400   IN      A       131.111.8.70
orange.csi.cam.ac.uk.   86400   IN      A       131.111.8.77
green.csi.cam.ac.uk.    86400   IN      A       131.111.8.57
yellow.csi.cam.ac.uk.   86400   IN      A       131.111.8.67
```

Slide 153: Multiple A records

Multiple A records

It is quite possible to have multiple A records with different IP addresses for the same FQDN or different FQDNs with the same IP address.

- `/etc/nsswitch.conf`

  ```
  hosts:      files nisplus nis dns
  ```

- `/etc/resolv.conf`

  ```
  nameserver 131.111.8.42
  nameserver 131.111.12.20
  ```

Slide 154: Looking up a FQDN

Looking up a FQDN: `/etc/nsswitch.conf`

If a command is issued using a FQDN then the system needs to resolve that into an address.

First the system must determine where to look up the name. This includes the decision on whether to even use the DNS at all. The system file that governs this is `/etc/nsswitch.conf`. It has one line for each service the system might need to do lookups for. The line starting "`host:`" determines how host lookups are done. It is followed by a series of keywords: "`files nisplus nis dns`". Each of these corresponds to a lookup service.

files: Use `/etc/hosts`

nis: Use the Network Information Service (NIS or YP) - Not used in this course.

nisplus: Use the successor to NIS, NIS+ - Not used. Ever.

dns: Use the DNS

The FQDN is first looked up in the `/etc/hosts` file. If it is not there then the NIS service would be used if it was provided. But it isn't so the NIS+ service would be asked if it was provided. It's not. And never will be; NIS+ has never taken off outside a few pure-Solaris environments. Finally the DNS will be consulted.

```
# Do not remove the following line, or various programs
# that require network functionality will fail.
127.0.0.1       localhost.localdomain   localhost
128.232.253.1   pcttr101.titan1.pwf.cam.ac.uk
```

- One address
- Many names

Slide 155: Format of `/etc/hosts`

Format of `/etc/hosts`

The `/etc/hosts` file gives a series of names corresponding to an IP address with one address per line. If a name is being looked for it must match exactly. If the name corresponding to an IP address is being looked for then the first name in the line will be used. For this reason the fully qualified domain name should always be the first entry in the list of names.

Looking up a FQDN in the DNS: `/etc/resolv.conf`

Once the decision to use the DNS has been made the system needs to determine which DNS server(s) to use. This is determined by looking in the `/etc/resolv.conf` file (NB the missing "e" on "resolve"). Within this file will be a "`nameserver`" line which identifies the server to use. If there are multiple lines then they are used in order. In our case, we have two nameserver lines:

```
nameserver 131.111.8.42
nameserver 131.111.12.20
```

so the system will first ask `131.111.8.42` to give an IP address for the FQDN and if it does not respond then it will ask `131.111.12.20`. Note that a server sends back an answer that the name does not exist in the DNS then subsequent servers won't be asked. This is purely a resilience feature against dead servers.

- `/etc/resolv.conf`

  ```
  search titan1.pwf.cam.ac.uk
  ```

Slide 156: Handling incomplete names `search titan1.pwf.cam.ac.uk`

1. Query `pcttr101` as a FQDN — Fail
2. Query `pcttr101.titan1.pwf.cam.ac.uk` as a FQDN — Succeed

Slide 157: Handling incomplete names: `pcttr101`

`search titan1.pwf.cam.ac.uk`

1. Query `pcttr101.titan1.pwf` as a FQDN — Fail
2. Query `pcttr101.titan1.pwf.titan1.pwf.cam.ac.uk` as a FQDN — Fail
3. Query `pcttr101.titan1.pwf.pwf.cam.ac.uk` as a FQDN — Fail
4. Query `pcttr101.titan1.pwf.cam.ac.uk` as a FQDN — Succeed

Slide 158: Handling incomplete names: `pcttr101.titan1.pwf`

Handling incomplete names

But what happens if the name to be resolved is not fully qualified? If an unqualified name (e.g. `pcttr101`) or partially qualified name (e.g. `pcttr101.titan1`) appears as such in the `/etc/hosts` file then the `files` entry in the `/etc/nsswitch.conf` file will cause it to be resolved locally. Otherwise, the DNS client library turn to the `/etc/resolv.conf` file and looks for the `search` keyword. This instructs it on how to handle incomplete names.

The approach is to take the domain name following the `search` keyword and to use it and its parent domains as attempted bolt-ons to the offered name. So, if we have the line

```
search titan1.pwf.cam.ac.uk
```

in the `/etc/resolv.conf` file and the partially qualified name `pcttr101.titan1` needs to be resolved then the following procedure is followed:

1. A request is made for the partially qualified name, `pcttr101.titan1`. This is *always* the first step and ensures that FQDNs never get compromised by errors in the search configuration. This will fail.
2. The entire domain given to the `search` keyword is tacked on the end of the given name to create the trial FQDN `pcttr101.titan1.pwf.titan1.pwf.cam.ac.uk` and the DNS is queried for this. This will fail too.
3. The leading component of the domain is stripped off and what is left (`pwf.cam.ac.uk`) is added to the offered name to create `pcttr101.titan1.pwf.pwf.cam.ac.uk` which will also fail.
4. This stripping off will be repeated so that `cam.ac.uk` is added to give `pcttr101.titan1.pwf.cam.ac.uk` whose DNS lookup as a FQDN succeeds. The process stops here.

In general, the stripping off of layers of the added domain continues until there are two elements left to add (`ac.uk`). A single element (`uk`, in this case) is never added.

Comments in the `/etc/resolv.conf` file

Unlike almost all other system configuration files, “#” is *not* the comment character. The semicolon, “`;`”, is.

CNAME: Aliases

MX: Email domains

TXT: Text comments

Slide 159: Some other common record types in the DNS

Other record types in the DNS

To date we have worked soley with A records. We now need to address a few more.

```
www-uxsup.csx.cam.ac.uk. 86400   IN      CNAME   nurse.csi.cam.ac.uk.
```

Slide 160: CNAME Records

CNAME Records

Canonical name (CNAME) records permit the look up from an “alias” to a “real” name which in turn can be looked up through an A record to get an IP address. These are typically used to identify services which might move from machine to machine.

Note that the target of a CNAME record is required to be resolvable in an A record and not another CNAME record, though this requirement is not often enforced. It does avoid of loops, though.

```
hermes.cam.ac.uk.       86400   IN      MX      5 green.csi.cam.ac.uk.
hermes.cam.ac.uk.       86400   IN      MX      5 orange.csi.cam.ac.uk.
hermes.cam.ac.uk.       86400   IN      MX      5 yellow.csi.cam.ac.uk.
hermes.cam.ac.uk.       86400   IN      MX      7 ppsw.cam.ac.uk.
hermes.cam.ac.uk.       86400   IN      MX      5 red.csi.cam.ac.uk.
```

Slide 161: MX Records

MX Records

Most of the DNS is concerned with individuals hosts or services referring to individual hosts. Email is an exception. The MX record is typically defined to identify the hosts email should be sent to for a *domain*. Note that multiple entries often exist (so that email can still be received if a single host goes down) and that they have priorities.

In the example shown, the machines `green`, `orange`, `yellow` and `red` all have priority 5. This is higher than the priority 7 of `ppsw`. If email is to be sent to an address `@hermes.cam.ac.uk` then an email system will try to send it to these hosts first. If none of them can be contacted to receive it then the sender will fall back to the lower priority system `ppsw`.

Note that the target of an MX record is required to be resolvable in an A record and not a CNAME record, though this requirement is not often enforced.

- Tree structure
- Domains
- Zones

Slide 162: Structure of DNS

- `titan1.pwf.cam.ac.uk`
- `pwf.cam.ac.uk`
- `cam.ac.uk`
- `ac.uk`
- `uk`

Slide 163: Domain structure

- `cam.ac.uk`
- `ac.uk`
- `uk`
- `.`

Slide 164: Zone structure

The structure of the DNS

The DNS is a tree structure, with nodes called "domains". So the machine `ttr101.titan1.pwf.cam.ac.uk` is within the domain `titan1.pwf.cam.ac.uk` which is within the domain `pwf.cam.ac.uk` which is ... within the domain `uk`.

The DNS is not all held in a single place. It is split into "zones" which are the administrative units of the DNS. Each zone is a part of the DNS corresponding to a domain which is delegated to an institution (for example `cam.ac.uk` is delegated to the University Computing Service) minus those parts of the domain that have been further delegated (for example `cl.cam.ac.uk` is delegated to the Computer Laboratory). So the hostname `ttr101.titan1.pwf.cam.ac.uk` is within the `cam.ac.uk` domain and the `cam.ac.uk` zone, but the name `shep.cl.cam.ac.uk` is in the `cam.ac.uk` domain but the `cl.cam.ac.uk` zone.

A name lives in many domains (all the domains "above" it) but exactly one zone.

Zones are administered from one or more servers which are "authoritative" for the zone. These are the definitative sources of information regarding names in the zone but their answers are often cached in other namesservers thoughout the internet. This is why the records all have times to live wired into them; they specify how long a record may be cached by a server.

```
57.8.111.131.in-addr.arpa. 86400 IN     PTR     green.csi.cam.ac.uk.
```

- Not automatically generated
- Can be independently controlled

Slide 165: PTR records

PTR records — Getting names from addresses

There is no "reverse look up" in the DNS; there is no way to ask the question "what A records have this IP address?". And yet we need a way to identify the name of a system that is making a network connection where all we start with is the IP address of the system connecting.

The lookup from IP address to name is done by an *independent* set of records called PTR Records.

```
57.8.111.131.in-addr.arpa. 86400 IN     PTR     green.csi.cam.ac.uk.
```

`in-addr.arpa`: A DNS domain devoted to PTR records

`111.131`: A subdomain delegated to Cambridge

`57.8`: Identifiying a specific address

Slide 166: PTR records — Structure of key

Structure of PTR records

When a block of IP addresses (for example `131.111.0.0/16`) is allocated to an institution, which is a distinct action from delegating a name block of the DNS (e.g. `cam.ac.uk`), a subdomain of the `in-addr.arpa` domain (`111.131.in-addr.arpa` in this case) *is* delegated with it as a zone. This domain is designed for the setting up of PTR records.

The key for a PTR record consists of the address reversed under `in-addr.arpa` with the key of an A record as the corresponding value.

```
cam.ac.uk.  86400  IN  NS  chimaera.csx.cam.ac.uk.
```

cam.ac.uk. The zone

86400 Time to live of the record

IN It's an internet record

NS It's a name server record

chimaera.csx.cam.ac.uk FQDN of the server, which must be an A record

Slide 167: NS Records

NS Records

A name server (NS) record identifies the name of a server which is authoritative for a zone. Note that the IP address of the server is not contained in the record.

1. Client requests `uk.redhat.com` from `131.111.8.42` (`chimaera.csx.cam.ac.uk`)
2. `chimaera` knows the NS records for the `.` zone
3. `chimaera` selects one of them (`I.ROOT-SERVERS.NET`, say)
4. `chimaera` asks `I.ROOT-SERVERS.NET` for `uk.redhat.com`
5. `I.ROOT-SERVERS.NET` sends back the NS records for the `com` zone to `chimaera`
6. `chimaera` selects one of them (`K.GTLD-SERVERS.NET`, say)
7. `chimaera` asks `K.GTLD-SERVERS.NET` for `uk.redhat.com`
8. `K.GTLD-SERVERS.NET` sends back the NS records for the `redhat.com` zone to `chimaera`
9. `chimaera` selects one of them (`NS2.redhat.com`, say)
10. `chimaera` asks `NS2.redhat.com` for `www.redhat.com`
11. `NS2.redhat.com` sends back the A record to `chimaera`
12. `chimaera` passes the record back to the client

Slide 168: Chronology of a lookup

Chronology of a lookup

We will now consider the details of exactly how a name is looked up in the DNS. We will start with the client requesting `uk.redhat.com` from `131.111.8.42` (`chimaera.csx.cam.ac.uk`). It knows to contact this server from its `/etc/resolv.conf` file.

```
.                       204780  IN      NS      A.ROOT-SERVERS.NET.
.                       204780  IN      NS      B.ROOT-SERVERS.NET.
 ...
.                       204780  IN      NS      L.ROOT-SERVERS.NET.
.                       204780  IN      NS      M.ROOT-SERVERS.NET.

A.ROOT-SERVERS.NET.     559491  IN      A       198.41.0.4
B.ROOT-SERVERS.NET.     559491  IN      A       128.9.0.107
 ...
L.ROOT-SERVERS.NET.     559491  IN      A       198.32.64.12
M.ROOT-SERVERS.NET.     559491  IN      A       202.12.27.33
```

Slide 169: Top level records

Stage one—The `.` zone

Every name server has wired into it the names and IP addresses of the top-level name servers. These are updated regularly just in case they have changed or had entries added and must not be allowed to go missing or stale.

`chimaera` selects one of them at random or based on some system for locating the “nearest”. In our example we will assume it uses `I.ROOT-SERVERS.NET`. If it cannot connect to this server it will try another, of course.

`chimaera` asks `I.ROOT-SERVERS.NET` for `uk.redhat.com`.

```
com.                   172800  IN      NS      A.GTLD-SERVERS.NET.
com.                   172800  IN      NS      B.GTLD-SERVERS.NET.
 ...
com.                   172800  IN      NS      L.GTLD-SERVERS.NET.
com.                   172800  IN      NS      M.GTLD-SERVERS.NET.

A.GTLD-SERVERS.NET.    172800  IN      A       192.5.6.30
B.GTLD-SERVERS.NET.    172800  IN      A       192.33.14.30
 ...
L.GTLD-SERVERS.NET.    172800  IN      A       192.41.162.30
M.GTLD-SERVERS.NET.    172800  IN      A       192.55.83.30
```

Slide 170: Stage two—the `.com` zone

Stage two—the .com zone

`I.ROOT-SERVERS.NET` sends back the NS records for the `com` zone to `chimaera` instead of the answer to the question asked. This is called a "referral". It will typically also send back "additional information" that the querier might find useful such as the A records for the hosts named in the NS records. These records are usually returned in a random order to guard against those systems that always select the first (working) record given. `chimaera` selects one of these nameservers. Again this is either at random or according to some procedure for identifying the "nearest". In our example we will assume it uses `K.GTLD-SERVERS.NET`.

`chimaera` asks `K.GTLD-SERVERS.NET` for `uk.redhat.com`.

```
redhat.com.            172800  IN      NS      NS1.redhat.com.
redhat.com.            172800  IN      NS      NS2.redhat.com.
redhat.com.            172800  IN      NS      NS3.redhat.com.

NS1.redhat.com.        172800  IN      A       66.187.233.210
NS2.redhat.com.        172800  IN      A       66.77.185.41
NS3.redhat.com.        172800  IN      A       66.187.229.10
```

Slide 171: Stage three—the `.redhat.com` zone

Stage three—the .redhat.com zone

`K.GTLD-SERVERS.NET` sends back the NS records for the `redhat.com` zone to `chimaera` together with the corresponding A records as a referral.

`chimaera` selects one of them, again at random. In our case we will assume that `NS2.redhat.com` is picked and is contactable.

`chimaera` asks `NS2.redhat.com` for `uk.redhat.com`.

```
uk.redhat.com.          60      IN      A       193.103.254.132
```

Slide 172: Stage four—the answer

Stage four—the answer

`NS2.redhat.com` sends back the A record to `chimaera` and `chimaera` passes the answer to the client

`dig`: "domain information groper"

There is a tool for allowing detailed analysis of the DNS called `dig`. `dig` is typically run with the following arguments:

```
$ dig @server FQDN type [options]
```

server: the name or IP address of the DNS server to use. If you use the name then this will have to be looked up first. If no server is given then `dig` runs through the servers quoted in `/etc/resolv.conf`.

FQDN: the fully qualified domain name whose records are to be looked for.

type: the type of record: A, CNAME, MX, etc. If none is given then A records and CNAME records will be looked for.

options: typically none are used. In this course we will use the `+norecursive` option to turn off the default behaviour of starting and `.` and working all the way. This causes `dig` to request thast the server only do the first phase of the query, allowing us to follow through the steps manually.

For example, to get an A record from `chimaera` for `uk.redhat.com` run the command

```
$ dig @131.111.8.42 uk.redhat.com A
; <<>> DiG 9.2.1 <<>> @131.111.8.42 uk.redhat.com A
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 4932
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 3, ADDITIONAL: 3

;; QUESTION SECTION:
;uk.redhat.com.                 IN      A

;; ANSWER SECTION:
uk.redhat.com.          60      IN      A       193.103.254.132

;; AUTHORITY SECTION:
uk.redhat.com.          86400   IN      NS      ns2.redhat.com.
uk.redhat.com.          86400   IN      NS      ns3.redhat.com.
uk.redhat.com.          86400   IN      NS      ns1.redhat.com.

;; ADDITIONAL SECTION:
ns1.redhat.com.         294     IN      A       66.187.233.210
ns2.redhat.com.         594     IN      A       66.77.185.41
```

```
ns3.redhat.com.         594     IN      A       66.187.229.10

;; Query time: 157 msec
;; SERVER: 131.111.8.42#53(131.111.8.42)
;; WHEN: Thu Nov 28 12:14:15 2002
;; MSG SIZE  rcvd: 149
```

```
; <<>> DiG 9.2.1 <<>> @131.111.8.42 uk.redhat.com A
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 4932
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 3, ADDITIONAL: 3

;; QUESTION SECTION:
;uk.redhat.com.                 IN      A
```

Slide 173: `dig`'s output—The top

`dig`'s output—The top

The first line identifies the version of `dig` and the arguments passed

The rest of the first block describes the answer. It was a query, it did not return an error and the query has been assigned an ID of 4932. The interesting flag is `rd` which mean "recursiion desired". The server will follow up any referrals that are passed back. There was one query which returned one answering record, three records establising the authority of the answer (see below) and three records of additional information.

The second block simply lays out the question in the form of an incomplete record.

```
;; ANSWER SECTION:
uk.redhat.com.          60      IN      A       193.103.254.132
```

Slide 174: `dig`'s output—The answer section

`dig`'s output—The answer section

This bit is easy to understand. This is the answering record. If more than one record was found more than one would have been listed here.

```
;; AUTHORITY SECTION:
uk.redhat.com.          86400   IN      NS      ns2.redhat.com.
uk.redhat.com.          86400   IN      NS      ns3.redhat.com.
uk.redhat.com.          86400   IN      NS      ns1.redhat.com.
```

Slide 175: `dig`'s output—The authority section

`dig`'s output—The authority section

This section quotes the name servers that are authoritative for the zone that the record is in. These are the servers that ultimately claim the truth of the record in the answer section.

```
;; ADDITIONAL SECTION:
ns1.redhat.com.         294     IN      A       66.187.233.210
ns2.redhat.com.         594     IN      A       66.77.185.41
ns3.redhat.com.         594     IN      A       66.187.229.10
```

Slide 176: `dig`'s output—The additional information

`dig`'s output—The additional information

This is the additional useful information sent back with the answer. Typically it only contains the information that came back with the last phase of the query.

```
;; Query time: 157 msec
;; SERVER: 131.111.8.42#53(131.111.8.42)
;; WHEN: Thu Nov 28 12:14:15 2002
;; MSG SIZE  rcvd: 149
```

Slide 177: `dig`'s output—The tail

`dig`'s output—The tail

This sums up the entire query. The whole shenanigans took 157 milliseconds using port 53 on server 131.111.8.42.

- Use `dig`
- Use the `+norecusive` option
- Look up the PTR record for `1.253.232.128.in-addr.arpa`
- Start with one of the *root* name servers

  (You may assume you know their IP addresses)

  \# **`dig @`*`address`*` 1.253.232.128.in-addr.arpa PTR +norecursive`**
- Keep a record of every lookup you have to do
- How many do you have to do?

A.ROOT-SERVERS.NET	198.41.0.4	B.ROOT-SERVERS.NET	128.9.0.107
C.ROOT-SERVERS.NET	192.33.4.12	D.ROOT-SERVERS.NET	128.8.10.90
E.ROOT-SERVERS.NET	192.203.230.10	F.ROOT-SERVERS.NET	192.5.5.241
G.ROOT-SERVERS.NET	192.112.36.4	H.ROOT-SERVERS.NET	128.63.2.53
I.ROOT-SERVERS.NET	192.36.148.17	J.ROOT-SERVERS.NET	192.58.128.30
K.ROOT-SERVERS.NET	193.0.14.129	L.ROOT-SERVERS.NET	198.32.64.12
M.ROOT-SERVERS.NET	202.12.27.33		

Slide 178: Exercise: Looking up a PTR record

6 The kernel

6.1 /proc: The window on the kernel

```
$ ls /proc
1    4    apm         cpuinfo     ...
2    413  bus         devices     ...
3    422  cmdline     dma         ...
...
```

Slide 179: Looking in /proc

The proc directory structure

Within the Linux file system is a filesystem whose files do not correspond to any "real" files of data but instead provide simple access to the information held by the kernel.

The /proc directory tree contains two sorts of data: data about the system and data about user processes. Each of the directories with a numerical name carries data about the process with that number as its process id. We will address the per-process directories later. For now we will consider all the others, which carry data about the base operating system and the machine it is running on.

A word of warning: while the files are presented as such, they are not really files and this can confuse tools like ls. Each file is listed as having zero length, but still has content.

```
$ ls -l /proc/cmdline
-r--r--r--  1 root root  0 Nov 10 16:35 /proc/cmdline
$ cat /proc/cmdline
ro root=LABEL=/ hdc=ide-scsi
```

Slide 180: Confusing ls

The system /proc files

The manual page proc(5) will gives details on the structures in your system's /proc. The most useful are listed here:

cmdline The line the kernel was started with.

cpuinfo CPU information. The exact set of lines depends on the make of chip.

devices A list of all the devices under /dev and the corresponding major device numbers. These are the numerical values the kernel uses internally to identify devices. Note that this does not tell you what *hardware* devices are attached. See the /proc/pci entry for that.

filesystems The types of file system the kernel knows about. Rarely useful unless things are going seriously wrong.

interrupts On a PC this lists what the kernel thinks is on each interrupt. This can be very useful if you are trying to track down IRQ clashes.

loadavg The system load average, as used by the `uptime` program.

meminfo Memory allocation, as used by the `free` program.

modules A list of all the modules loaded by the system.

net/ A directory of entries referring to the kernel's networking features. See the networking concepts section of the course for details.

net/arp The ARP table.

net/route The routing table with IP addresses in hexadecimal format.

pci This lists every PCI device on the system, together with an identifying string and some technical details.

sys/ This directory tree contains a large number of files whose names are kernel parameters. Reading these files gives the value of the kernel parameter (e.g. `/proc/sys/fs/file-max` gives the maximum number of files that the system can have open at once.) Some of these files can be written to (by `root`) to change the value while the system is running.

```
# hostname
# echo surname > /proc/sys/kernel/hostname
# hostname
# echo original > /proc/sys/kernel/hostname
```

Slide 181: Exercise: Playing with `/proc`

The /proc/*PID*/ files

The per-process entries under `/proc` are typically far more useful than the system-wide entries.

cmdline: The command line used to launch the command. This has no terminating newline character, so may get confused with your prompt.

cwd: A "magic symlink" that points to the process' current working directory.

environ: The process' environment variables. These are unfortunately all run together in a way that is hard to mechanically unpick.

exe: Another "magic symlink" pointing to the program corresponding to the process.

fd/: This lists all the various files the process has open with symlinks pointing to each. Files of the form `/dev/pts/`*N* are login sessions' "pseudoterminals".

root: This points to the "root" of the process. This is how "high" up the file system the process could go if it continually change directory to **..**, its current parent directory. For almost all processes it is `/` but this can be changed with the `chroot`(1) command.

maps: This reveals where various binaries (especially shared libraries) are allocated in the process' memory map and which segments of process memory the process can use as "scratch space".

status: The status of the process as illustrated by `ps`.

1. Start up an X terminal as a user.
2. Start another as `root`.
3. Determine the user's shell's process id. (`echo $$`)
4. Move `root` to the corresponding `/proc` subdirectory.
5. # **`ls -l`**
6. $ **`tty`**
7. # **`ls -l fd`**
8. # **`echo boo > fd/1`**
9. $ **`cd /tmp`**
10. # **`ls -l exe`**
11. $ **`exec /bin/csh`**
12. # **`ls -l exe`**.

Slide 182: Exercise: Using `/proc/`*`PID`*

6.2 Kernel upgrades

Upgrading the kernel

We've seen in an earlier section how the Red Hat kernel is split into sections and how the GRUB needs to know about some of them. In this section we are going to look at upgrading these components as there are some extra wrinkles over and above a usual package upgrade.

- `/boot/`
- `/lib/modules/`*`version`*`/`
- `/boot/grub/grub.conf`

Slide 183: Files relevant to Red Hat's kernel

Kernel packages

First let's look at the packages that form the installed kernel. The first thing we discover is that we have an unnecessary package: `kernel-source`. We're not about to build kernels in this course nor do we need PCMCIA cards and we might as well simplify the upgrade procedure as much as possible.

```
# rpm --query --all | grep kernel-
kernel-pcmcia-cs-3.1.31-13
kernel-source-2.4.20-8
kernel-2.4.20-8
# rpm --erase kernel-source kernel-pcmcia-cs
```

Slide 184: Packages for Red Hat's kernel

```
# arch -k
i686
# mount -o ro nfs-uxsup.csx.cam.ac.uk:/linux/redhat /redhat
# cd /redhat/updates/9/en/os/i686
# ls kernel-*
kernel-2.4.20-24.9.i686.rpm
kernel-bigmem-2.4.20-24.9.i686.rpm
kernel-smp-2.4.20-24.9.i686.rpm
```

Slide 185: Upgrading Red Hat's kernel—1

```
# cd /redhat/updates/9/en/os/i386
# ls kernel-*
kernel-2.4.20-24.9.i386.rpm
kernel-BOOT-2.4.20-24.9.i386.rpm
kernel-doc-2.4.20-24.9.i386.rpm
kernel-source-2.4.20-24.9.i386.rpm
```

Slide 186: Upgrading Red Hat's kernel—2

Locating the relevant packages

You will also recall from the RPM section that the kernel packages are some of the few where there are different versions for i386, i486 and so on. We can determine the architecture of our system with the command "`arch -k`".

Red Hat have started to split their updates according to kernel architecture. The result is a confusing mess of directories, but the most recent versions of the packages that depend on kernel architecture have moved to a directory of their own, called `i686` in our case. The header files, however, are generic across all Intel platforms so are in the `i386` directory.

We must identify just the kernel packages that match our installed set and select the relevant kernel architecture where there is a choice. The "`-smp-`" set are the "Symmetric Multi-Processing" kernels. These are only suitable for systems with multiple CPUs. The "`-bigmem-`" kernels are compiled with support for more than 4GB of RAM and need not concern us.

If we examine the packages we see that the actual kernel package has the kernel version wired into every file path. This is to support the installation of multiple versions of the kernel. We will exploit this feature to install an extra kernel *alongside* the original. We also note that there can be dependencies between the kernel and other packages though they will not affect us here.

- And finally!

```
# cd /redhat/updates/9/en/os/i686
# rpm --install kernel-2.4.20-24.9.i686.rpm
```

Slide 187: Upgrading Red Hat's kernel—3

Installing the new RPMs

Note that we do *not* do an upgrade. We are installing a new kernel alongside the existing kernel. An upgrade would have left us without the old kernel.

`/boot`: Files added to. Symbolic links changed.

`/lib/modules`: New subdirectory.

`/boot/grub/grub.conf`: Manual change needed.

Slide 188: Booting off the new kernel

Changes made

The installation has made several additions to the software installed. The "new" kernel sits alongside the original in `/boot` and `/lib/modules`, but the three symbolic links in `/boot` have been changed to point to the most recently installed kernel:

- `vmlinuz`
- `module-info`
- `System.map`

Note that if we reboot at this stage we still get the old kernel because the GRUB has the old kernel's parameters wired into it as the default. (See `/boot/grub/grub.conf`.) It has the line `default=1` which refers to the second entry in the file (the first being number 0, of course) which boots the old kernel. We neeed to change this.

```
default=1
timeout=10
splashimage=(hd0,0)/grub/splash.xpm.gz
title Red Hat Linux (2.4.20-24.9)
        root (hd0,0)
        kernel /vmlinuz-2.4.20-24.9 ro root=LABEL=/ hdc=ide-scsi
        initrd /initrd-2.4.20-24.9.img
title Red Hat Linux (2.4.20-8)
        root (hd0,0)
        kernel /vmlinuz-2.4.20-8 ro root=LABEL=/ hdc=ide-scsi
        initrd /initrd-2.4.20-8.img
```

Slide 189: Checking `/boot/grub/grub.conf`

```
...
default=0
...
```

Slide 190: Editing `/boot/grub/grub.conf`

Fixing the GRUB

To get the GRUB to boot the new kernel we need to change its configuration file. Changing the `default` line is all that is needed. Out of paranoia we do not remove the ability to boot off the old kernel. The new instruction block is the same as the old one was but with the version number updated.

Your system can now boot both the old and new kernels with the new kernel as the default.

- Boot off the original Red Hat 9 kernel.
- Now make the original 9 kernel the default again.

Slide 191: Exercises

7 Service configuration

- Continuous?
- On demand?
- Periodic?

Slide 192: Types of service

Types of service

The various services provided on a Unix system can be categorised in a number of ways: They can be network services or purely internal services. They can be continually running, run on demand or run at regular intervals. Finally, they can be subtle services whose absence you don't notice for weeks or unsubtle services whose absence has an immediate impact.

From the administrative point of view of this course, the way the processes run and the corresponding ways they are launched is the most important categorisation.

7.1 Continually running services

- `sshd`—Secure logins
- `gpm`—Mouse handler
- `syslogd`—Logging service
- `xinetd`—Launcher of on-demand network services
- `crond`—Launcher of periodic services
- `portmap`—Lookup service for RPC services

Slide 193: Examples of continually running services

Simple service daemons

The simplest daemons to consider are the simple daemons that sit around waiting to do something. These are typically launched from a startup script in the directory `/etc/init.d`. (This is actually a symbolic link to `/etc/rc.d/init.d`, but the former is more standard and Red Hat are moving towards using the standard directory. As the first stage in this migration they have put in this symbolic link.) There are two aspects of the configuration of these services: whether they should be run at all and how they should run if they are. The latter depends greatly on the nature of the service itself. The former has a uniform solution provided by the `chkconfig` program.

```
#!/bin/bash
#
# chkconfig: 2345 85 15
# description: GPM adds mouse support to text-based Linux ... \
#              the Midnight Commander. Is also allows ... \
#              cut-and-paste operations, and includes ... \
#              menus on the console.
# processname: gpm
# pidfile: /var/run/gpm.pid
# config: /etc/sysconfig/mouse
```

Slide 194: The comments at the top of `/etc/init.d/gpm`

The `chkconfig` program

Each service start up file has (or at least *should have*) some lines of structured comment at the top. See slide 194 for the example of the startup script for the `gpm` mouse handling daemon.

The directly pertinent line is

```
# chkconfig: 2345 85 15
```

which says that the defaults for the service are that it should be run in runlevels 2, 3, 4 and 5 and that its startup scripts for those runlevels should be prefixed by “`S85`”. For the remaining runlevels its shutdown script should be prefixed by “`K15`”.

The other lines provide hints for future, more comprehensive system service manipulation programs.

The `chkconfig` program provides the main tool for changing the set of runlevels a service runs in, and for listing what services run when. The output comes in two distinct halves. The first half describes the constantly running services which we are describing here. The second half describes the “on demand” services and whether they are available. These will be covered later.

```
# chkconfig --list
kudzu           0:off   1:off   2:off   3:on    4:on    5:on    6:off
syslog          0:off   1:off   2:on    3:on    4:on    5:on    6:off
...
httpd           0:off   1:off   2:off   3:off   4:off   5:off   6:off
xinetd based services:
        chargen-udp:    off
        rsync:  off
        ...
```

Slide 195: Listing all services

```
# chkconfig --list gpm
gpm 0:off 1:off 2:on 3:on 4:on 5:on 6:off

# ls -l /etc/rc?.d/*gpm
lrwxrwxrwx  ...  /etc/rc0.d/K15gpm -> ../init.d/gpm
lrwxrwxrwx  ...  /etc/rc1.d/K15gpm -> ../init.d/gpm
lrwxrwxrwx  ...  /etc/rc2.d/S85gpm -> ../init.d/gpm
lrwxrwxrwx  ...  /etc/rc3.d/S85gpm -> ../init.d/gpm
lrwxrwxrwx  ...  /etc/rc4.d/S85gpm -> ../init.d/gpm
lrwxrwxrwx  ...  /etc/rc5.d/S85gpm -> ../init.d/gpm
lrwxrwxrwx  ...  /etc/rc6.d/K15gpm -> ../init.d/gpm
```

Slide 196: Listing just one service

Listing all services

The `chkconfig` program has a `--list` option which will list all the services managed through the startup scripts. Next to each it will list whether or not the service is run in runlevels 0 to 6. See slide 195 for an example.

Alternatively it can be given the name of a service to list and will produce a single line of output for that service. (It cannot be passed multiple service names, unfortunately.)

```
# chkconfig --list gpm
gpm 0:off 1:off 2:on 3:on 4:on 5:on 6:off
# chkconfig --level 24 gpm off
# chkconfig --list gpm
gpm 0:off 1:off 2:off 3:on 4:off 5:on 6:off
# ls -l /etc/rc[24].d/*gpm
lrwxrwxrwx  ...  /etc/rc2.d/K15gpm -> ../init.d/gpm
lrwxrwxrwx  ...  /etc/rc4.d/K15gpm -> ../init.d/gpm
```

Slide 197: Changing when the `gpm` service is launched

Changing when a service runs

We can also use `chkconfig` to change the set of startup/shutdown symbolic links in a controlled manner. If we wanted to stop `gpm` being launched in runlevels 2 and 4 we can call `chkconfig` with the `--level` option as shown in slide 197.

Note that this changes the scripts run at the next shutdown and startup. If you are already running the runlevel you are changing you need to manually startup or shutdown the service to bring the running system into line with the new set of startup scripts.

```
# ps -ef | grep gpm | grep -v grep
root      1105     1  0 Nov04 ?        00:00:00 gpm -t imps/2 -m /dev/mouse
# runlevel
3 5
# chkconfig --level 3 gpm off
# ps -ef | grep gpm | grep -v grep
root      1105     1  0 Nov04 ?        00:00:00 gpm -t imps/2 -m /dev/mouse
# /etc/init.d/gpm stop
Shutting down gpm mouse services: gpm
# ps -ef | grep gpm | grep -v grep
#
```

Slide 198: Turning off a service in the current runlevel

```
service httpd start    =  /etc/init.d/httpd start
service httpd restart  =  /etc/init.d/httpd restart
service httpd status   =  /etc/init.d/httpd status
service httpd stop     =  /etc/init.d/httpd stop
```

Slide 199: The `service` program: Syntactic sugar for start-up scripts

Services that launch services

Two of these permanently running services deserve special attention. The "`crond`" is designed to start services periodically. These are used to perform routine system maintenance, etc. The "`xinetd`" listens to the network on behalf of a range of services. When a query comes in for a particular service then `xinetd` starts up that service for this particular query, connects the network connection to this newly started service and then waits for the next incoming connection.

7.2 Network services run from `xinetd`

- `in.ftpd`—Insecure file transfer
- `imapd`—Mail manipulation
- `rsync`—File distribution

Slide 200: Examples of on-demand services

The `xinetd` service

The `xinetd` program listens on a set of ports, each corresponding to a different service. When a connection is made to the port corresponding to a particular service (e.g. rsync), `xinetd` starts up the daemon for that service (e.g. `rsync --daemon`) and passes the connection through to this child process.

The `xinetd` program therefore needs to be told what ports to listen to and what programs it should launch to handle the requests. It is given this information in the configuration file `/etc/xinetd.conf`. This file simply sets some defaults and includes the contents of the `/etc/xinetd.d` directory. This directory then has one file per service that `xinetd` will be supporting.

```
#
# 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
```

Slide 201: The file `/etc/xinetd.conf`

```
# default: off
# description: The rsync server is a good addition to am ftp server, as it \
#       allows crc checksumming etc.
service rsync
{
        disable         = yes
        socket_type     = stream
        wait            = no
        user            = root
        server          = /usr/bin/rsync
        server_args     = --daemon
        log_on_failure  += USERID
}
```

Slide 202: The file `/etc/xinetd.d/rsync`

`instances` Number of daemons

`log_type`, `log_on_success`, `log_on_failure` Logging

`cps` Connections per second

`disable` Is this service turned off?

`socket_type`, `flags` Network options

`wait` Should `xinetd` wait for this service to end before starting another?

`user`, `group` Who the service should run as

`server` The program that runs the service

`server_args` Options passed to the program

Slide 203: `xinetd` keywords

The `/etc/xinetd.d/` files

The `/etc/xinetd.d/` files have a fairly simple format: blank lines are ignored and the "#" character is a comment character. A service is identified by name followed by its changes or additions to the default settings in curly brackets.

We'll consider the `/etc/xinetd.d/telnet` file as a simple example to get us going. First we consider the defaults defined in `/etc/xinetd.conf`.

- `instances = 60`

  At most 60 instances of the telnet daemon should be run at once.

- `log_type = SYSLOG authpriv`

  Logging should be via the system logger with the "privileged authentication information" facility.

- `log_on_success = HOST PID`

  If the connection attempt is successful, log the originating hostname and the process id of the launched daemon.

- `log_on_failure = HOST`

  If the connection attempt is unsuccessful, log the originating host.

This is then modified by the `/etc/xinetd.d/rsync` file's contents:

- `socket_type = stream`

  This says that the connection should be a TCP connection because it will be streaming data across the network. Setting `socket_type` to `dgram` will get a UDP (unreliable datagram) service.

- `wait = no`

  Some child processes can act as master servers in their own rights. In this case, `xinetd` can be used to launch that first one but should then not launch any more until the one it launched has died off. This would be called waiting for the child. The telnet daemon is not one of these services so we do not wait for it.

- `user = root`

  The service should be run as user `root`. There is a corresponding `group` option to set the child process' group.

- `server = /usr/bin/rsync`

  This is where the program is that actually runs the service.

- `server_args = --daemon`

  These are the options passed to the program, so the commabnd that will be run is `rsync --daemon`.

- `log_on_failure += USERID`

  In addition to the default `log_on_failure` options, also log the userid quoted for any failed connection. Note the use of += to add to the connections. Simply using = would have overridden the previous ones. You can use -= to remove individual options too.

- `disable = yes`

  Whether the service should be disabled, i.e. not actually offered.

Making changes to /etc/xinetd.conf

The `xinetd` reads its configuration file, `/etc/xinetd.conf`, and the configuration directory, `/etc/xinetd.d/`, when it is launched and when it receives the HUP signal. Signals are sent using the `killall` command which sends a signal to all running processes of a particular name as shown in slide 204. The slide also shows the "manual" version for comparison and the use of the "`reload`" option of the `inet` startup script. Many daemons' startup scripts have extra options to reload configuration files or to restart fresh versions of their daemons.

```
# killall -HUP xinetd

# ps -ef | grep xinetd | grep -v grep
root       438     1  0 Nov10 ?        00:00:00 xinetd
# kill -HUP 438

# /etc/init.d/xinetd reload
```

Slide 204: Three ways to send a HUP signal to the running `xinetd`

```
# grep disable /etc/xinetd.d/rsync
        disable = yes
# chkconfig rsync on
# grep disable /etc/xinetd.d/rsync
        disable = no
# chkconfig rsync off
# grep disable /etc/xinetd.d/rsync
        disable = yes
```

Slide 205: Disabling an `xinetd` service

Disabling an xinetd service

Typically the change you want to make is to switch the disablement from `yes` to `no` or vice versa. This can be done with `chkconfig` or an editor. There is no mention of runlevels in this call of the program. If `xinetd` is running at a particular runlevel then so are all the services in `/etc/xinetd.d` with `disable = no`.

7.3 Services run from crond

- `logrotate`—Manage log files
- `slocate`—Set up databases for `slocate`
- `tmpwatch`—Clean out old junk from `/tmp`

Slide 206: Examples of periodically run services

```
$ ls -ld /etc/cron.*
drwxr-xr-x 2 root root 1024 Oct  7 18:56 /etc/cron.d
drwxr-xr-x 2 root root 1024 Oct  7 19:37 /etc/cron.daily
drwxr-xr-x 2 root root 1024 Aug 27 15:04 /etc/cron.hourly
drwxr-xr-x 2 root root 1024 Aug 27 15:04 /etc/cron.monthly
drwxr-xr-x 2 root root 1024 Oct  7 19:18 /etc/cron.weekly
```

Slide 207: The `crond` control directories

`crond`

The cron daemon has a variety of system jobs that it runs with various periods. These are coordinated from a series of directories listed in slide 207. We shall skip `/etc/cron.d` for the moment and look at the others. Each of these directories can contain programs (typically shell scripts) which are run daily, hourly, monthly or weekly according to the directory they are put in.

```
SHELL=/bin/bash
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root
HOME=/
# run-parts
01 * * * * root run-parts /etc/cron.hourly
02 4 * * * root run-parts /etc/cron.daily
22 4 * * 0 root run-parts /etc/cron.weekly
42 4 1 * * root run-parts /etc/cron.monthly
```

Slide 208: `/etc/crontab`

`/etc/crontab`

This is the `crond` master control file. The first half sets the environment for the daemon. The second half controls what jobs the daemon should launch and when. Each of these lines consists of a set of parameters dictating when the comand should be run followed by the command itself.

- Minute: 0–59
- Hour: 0–23
- Day of month: 1–31
- Month: 1–12
- Day of week: 0–7
- User
- The command (rest of line)

Slide 209: Format of `/etc/crontab` job lines

- Ranges: 1–5—every number from 1 to 5 inclusive
- All: *—every valid number
- Steps: */3—every third number in the range (starting with the first)
- Lists: 0,6,9,12,15,18
- Combinations: 0,6-18/3

Slide 210: Numeric constructions in crontab files

`run-parts`

It is easier for packages to make changes by adding and removing files than by editing the content of existing files. So if a package needs to add a cron job then *it does not edit this file*. Instead, this file runs a command called `run-parts` which takes a directory and runs all the shell scripts in it (subject to a few tests). The packages can then just drop files into these directories.

There are four of these directories:

`cron.hourly`: Jobs run at one minute past the hour.

`cron.daily`: Jobs run at 04:02 in the morning.

`cron.weekly`: Jobs run at 04:22 every Sunday morning.

`cron.monthly`: Jobs run at 04:42 every first of the month.

The `/etc/cron.daily` directory

The `/etc/cron.daily` directory contains a number of shell scripts for `crond` to run through at 04:02 each morning. Each shell script runs a related set of instructions and is typically dropped in by a package.

The shell scripts are typically very simple, often just launching a single process `/etc/cron.daily/logrotate` is a good example of this.

```
#!/bin/sh

/usr/sbin/logrotate /etc/logrotate.conf
```

Slide 211: The `/etc/cron.daily/logrotate` file

`/etc/cron.d`

Sometimes you want finer tuning than this. You might want to run programs every ten minutes or so. The files that can be run from `/etc/cron.d` are not simple shell scripts. They are files that have the same syntax as the master control file, `/etc/crontab`. The `crond` checks this directory for updates once a minute.

- Simple

  Every hour

  `date >> /var/tmp/hourly.output`

- Harder

  Every other minute

  from 01 past the hour

  to 29 past the hour

  `date >> /var/tmp/other.output`

Slide 212: Exercise: Writing cron jobs

7.4 Log files

- Run from `/etc/cron.daily/logrotate`
- Configuration: `/etc/logrotate.conf`, `/etc/logrotate.d/`

Slide 213: Log rotation

Log rotation

One of the more important system services that conducts its business behind the scenes is the log rotation facility run from `crond`. This has a primary configuration file `/etc/logrotate.conf` which in turn includes all the files in the directory `/etc/logrotate.d/`. It is in this directory that packages and other software should place their log rotation instructions.

The default behaviour in the primary configuration file is to rotate logs weekly (though the service is started daily in case there are any sets of logs that override this. Four copies of old log files are kept. A file called `logfile` will have four old versions maintained: `logfile.0`, `logfile.1`, `logfile.2` and `logfile.3`. When it's time to rotate the log file the file `logfile.2` will be renamed to `logfile.3`, destroying the oldest version in the process, `logfile.1` will be renamed to `logfile.2`, `logfile.0` to `logfile.1` and `logfile` to `logfile.0`. Then an empty file `logfile` will be created for the logging to continue to. It's possible, and advisable, to set the owner and the mode of this file in the configuration file. An example of such a set of overrides can be seen in the rotation instructions for the login record `/var/log/wtmp` in slide 214. This specifies monthly rotation with a single old copy kept. The new file should be created with mode 0664, owned by user `root` and group `utmp`.

```
/var/log/wtmp {
    monthly
    create 0664 root utmp
    rotate 1
}
```

Slide 214: A rotation instruction in `/etc/logrotate.conf`

```
/var/log/cron {
    sharedscripts
    postrotate
        /bin/kill -HUP `cat /var/run/syslogd.pid 2>/dev/null` 2>/dev/null || true
    endscript
}
```

Slide 215: The rotation of `/var/log/cron`

The file `/etc/logrotate.d/syslog`

The `ksyslogd` package provides a log rotation file to manage the log files it creates. For example, the `crond` logs via the system logger so the system logging package has to take care of the rotation because only it knows where it will be filing `crond` log messages. The section for `/var/log/cron` doesn't override any of the defaults set up in `/etc/logrotate.conf` but does cause a command to be run after the rotation has completed. This prods the `syslogd` to reopen its output files.

7.5 A review of common services

1. Find *every* service on your system.
2. Do you want to be running it?
3. Where is it configured?
4. Where does it log to and are its logs rotated?
5. What are the implications of running it?
   Security, auditing, DPA, ...

Slide 216: Exercise: Analysing your services

Commonly used services

The following section lists the services that might be running on a Red Hat Linux system. In each case the package is given. Listing the package contents should identify the programs, manual pages, documentation, configuration files and log rotation files. The file listing will probably not give the log files explicitly but examining the log rotation files or configuration files should identify these.

anacron: Run jobs regularly, allowing for downtime

Package: `anacron`

Launch: `/etc/init.d/anacron`

Description: Another daemon that runs programs regularly, like `cron`. Unlike `cron`, it does not assume that the machine is running continuously and on boot, it determines what should have been run during the downtime.

apmd: Support for power-saving in laptops

Package: `apmd`

Launch: `/etc/init.d/apmd`

Description: The daemon deals with power-saving mode on laptops and is also capable of shutting down the PCMCIA sockets before a suspend and can watch the laptop's battery to provide warnings of imminent power troubles. Run this only if you are using a laptop.

arpwatch: Tracks changes in ethernet/IP matches

Package: `arpwatch`

Launch: `/etc/init.d/arpwatch`

Description: A simplistic test to spot crude attempts at spoofing systems. Tends to generate large amounts of data about quite innocent systems when first launched.

at: Service to launch a program (once) at a certain time

Package: `at`

Launch: `/etc/init.d/atd`

Description: Very similar to `cron` but jobs get scheduled to be run once only.

autofs: Package: `autofs`

Launch: `/etc/init.d/autofs`

Description: Runs the NFS automounter which can mount remote file systems on demand.

comsat: An ancient mail notification service

Package: `comsat`

Launch: `/etc/xinetd.d/comsat`

Description: This service is obsolete. Don't run it.

cron: Run jobs regularly

Package: `vixie-cron`

Launch: `/etc/init.d/crond`

Description: The standard cron program that runs programs regularly. Your system needs to be running this service.

finger: The classic finger service

Package: `finger-server`

Launch: `/etc/xinetd.d/finger`

Description: This service provides information about users. Cambridge blocks this service at the JANET router for DPA reasons.

FTP: file transfer service

Package: `wu-ftpd`

Launch: `/etc/xinetd.d/wu-ftpd`

Description: The classic FTP service. Note that *anonymous* FTP is provided only if you have a `/home/ftp` directory. Support for anonymous FTP is provided by the additional `anonftp` package. The file transfer protocol passes passwords in clear across the network; use `sshd` instead if possible.

gpm: mouse server for the console

Package: `gpm`

Launch: `/etc/init.d/gpm`

Description: Provides cut and paste for the text mode console and an optional common mouse interface for X.

httpd: Web server

Package: `apache`

Launch: `/etc/init.d/httpd`

Description: The classic web service. If you are going to run this service you need to be aware of the DPA implications of the logs you keep. Access records to a user's web pages count as personal data for that user but may need certain third party data removed before being passed on.

IMAP, IMAPS: **Package:** `imap`

Launch: `/etc/xinetd.d/imap`, `/etc/xinetd.d/imaps`

Description: Mail reading protocols, only suitable for mail servers. If you can, restrict to using the IMAPS (S for "secure") protocol. This doesn't send userid and password in plain text over the network.

identd: The identification server

Package: `pidentd`

Launch: `/etc/init.d/identd`

Description: This provides information to remote sites about who is connecting to them from your system. You may run this system under the DPA but you must inform your users that, as part of having an account, they agree to have their outgoing connections not be anonymous.

irda: **Package:** `irda-utils`

Launch: `/etc/rc.d/init.d/irda`

Description: IrDA(TM) (Infrared Data Association) is the infrared communication protocol.

ISDN: **Package:** `isdn4k-utils`

Launch: `/etc/rc.d/init.d/isdn`

Description: Access via high speed phone lines.

LDAP: **Package:** `openldap-servers`

Launch: `/etc/init.d/ldap`

Description: LDAP is a network directory service.

locate: index and search for files on your system

Package: `slocate`

Launch: `/etc/cron.daily/slocate.cron`

Description: The service maintains a database of all the files on the system in `/var/lib/slocate/slocate.db` together with their access permissions. A user can then search for a file by name with the `slocate` command.

log rotation: Rotate system logs

Package: `logrotate`

Launch: `/etc/cron.daily/logrotate`

Description: This is the log rotation service. The configuration files governing how often various logs are rotated is typically provided by the packages responsible for the log files.

lpd: Printing

Package: `LPRng`

Launch: `/etc/init.d/lpd`

Description: The print daemon required for lpr to work properly

makewhatis: build the man pages indexes

Package: `man`

Launch: `/etc/cron.daily/makewhatis.cron`

Description: The file `/usr/man/whatis` is used for “`man -k`” queries. It is built automatically from the manual pages each night.

NFS, NFS locking: **Package:** `nfs-utils`

Launch: `/etc/init.d/nfs`, `/etc/init.d/nfslock`

Description: Network File Service.

NTP: Network Time Protocol

Package: `ntp`

Launch: `/etc/init.d/ntpd`

Description: The Network Time Protocol (NTP) is used to synchronize a computer’s time with another reference time source. You want to run this service to keep your system’s clock right.

PCMCIA: **Package:** `kernel-pcmcia-cs`

Launch: `/etc/init.d/pcmcia`

Description: Support hot-swap PCMCIA cards.

POP2, POP4, POP3S: **Package:** `imap`

Launch: `/etc/xinetd.d/ipop2`, `/etc/xinetd.d/ipop3`, `/etc/xinetd.d/pop3s`

Description: Mail reading protocols, only suitable for mail servers. IMAP is better than POP.

PXE: **Package:** `pxe`

Launch: `/etc/init.d/pxe`

Description: PXE is the “Preboot eXecution Environment”. This services supports systems that boot off their network cards.

remote shell, login, execution, copy: run commands remotely, login or copy files between machines

Package: `rsh-server`

Launch: `/etc/xinetd.d/rexec`, `/etc/xinetd.d/rlogin`, `/etc/xinetd.d/rsh`

Description: Simple remote access for logins or running single commands etc. Passwords do not always need to be passed as remote accounts can be labelled as trusted. However, when passwords are passed they are passed in clear and the host trusting mechanism is easily spoofed. Use `sshd` instead if possible.

routed: Routing daemon

Package: `routed`

Launch: `/etc/init.d/routed`

Description: The route daemon maintains routing information dynamically according to RIP (Routing Information Protocol) messages from routers. Most of Cambridge works on *static* routing and the `routed` package should be removed in those circumstances.

RPC portmapper: RPC look up service

Package: `portmap`

Launch: `/etc/init.d/portmap`

Description: The RPC portmapper informs remote clients about which local port a particular RPC service is listening on.

rstat, rusersd: Remote access to performance metrics and lists of logged in users

Package: `rusers-server`

Launch: `/etc/init.d/rstatd`, `/etc/init.d/rusersd`

Description: These protocols allow remote users to retrieve performance metrics and lists of logged in users for the machine. Typically you don't want the information provided by the `rusers` package made available and you should remove the package.

rwall: Forward messages to every user

Package: `rwall-server`

Launch: `/etc/init.d/rwalld`

Description: The daemon receives messages from the outside world and sends the message on to every logged in user. Do not run this service.

rwho: Cooperating systems inform each other about who is logged in

Package: `rwho`

Launch: `/etc/init.d/rwhod`

Description: The `rwho` program doesn't just look at the system it is running on but polls the network too. Any local systems running the matching daemon respond and the result is a list of who is logged on where.

SMTP: E-mail

Package: `sendmail`

Launch: `/etc/init.d/sendmail`

Description: Used to send email out of the system and within the system. It is also used to receive email sent to this machine. Do not confuse it with mail reading protocols (e.g. IMAP and POP) that are used to read email delivered to another machine. You almost certainly want to replace the `sendmail` package with the `exim` package and to stop it listening on the network as email will not be normally directed to this system. Set "`DAEMON=no`" in `/etc/sysconfig/sendmail` or `/etc/sysconfig/exim` to achieve this.

SNMP: **Package:** `net-snmp`

Launch: `/etc/init.d/snmpd`, `/etc/init.d/snmptrapd`

Description: SNMP (Simple Network Management Protocol) is a poor-quality protocol used for network management.

SSH: Secure Shell access

Package: `openssh-server`

Launch: `/etc/init.d/sshd`

Description: Provides cryptographically secure network access to the system for login, commands and file transfer. You *do* want to be running this service.

syslog: System logging

Package: `sysklogd`

Launch: `/etc/init.d/syslog`

Description: This provides a common interface for services to log to. It is very commonly used and your system is not running properly without it.

talk: A variety of system-to-system chat programs

Package: `talk-server`

Launch: `/etc/xinetd.d/ntalk`, `/etc/xinetd.d/talk`

Description: Old fashioned chat programs.

TELNET: remote login

Package: `telnet-server`

Launch: `/etc/xinetd.d/telnet`

Description: The classic remote login program. The TELNET protocol passes passwords in clear across the network; use `sshd` instead if possible.

Trivial file transfer: A simple file transfer protocol

Package: `tftp-server`

Launch: `/etc/xinetd.d/tftp`

Description: This protocol provides for a very light-weight protocol to transfer small files. It is typically used to support network booting of simple network devices (printers, X-terminals etc.).

tmpwatch: Stop `/tmp` and `/var/tmp` from overfilling

Package: `tmpwatch`

Launch: `/etc/cron.daily/tmpwatch`

Description: By default, any file over 10 days old is deleted from `/tmp` and `/var/tmp`.

xfs: X Font Server

Package: `XFree86-xfs`

Launch: `/etc/init.d/xfs`

Description: This service provides X font definitions over the network. So, if a remote X server needs a font for a client running on the local system the local system can provide the font via this service.

8 Managing users and groups

Introduction

This section describes a number of methods for managing users and groups and introduces the concept of the *private group scheme*. This section will also illustrate how this grouping mechanism is closely intertwined with system security and show you how to configure the user environment through the use of global system files. A demonstration of the *User Manager* utility will also be given.

- Some background information.
- A `passwd` and `shadow` file tour.
- An introduction to groups.
- Utilities for managing accounts.
- The private group scheme.
- How to globally manage the user's enviroment.
- A `User Manager` utility tour.
- Exercises

Slide 217: Agenda

- So what is a user?
  - an individual who utilizes the resources of the machine, who can log in, edit files, read mail.
  - has a login name and a password.
  - may have a home directory and a bunch of files.
- User details are stored on the system in the shape of accounts.
- Accounts provide a way to distinguish between different users on the system.
  - *But* an account can also be a system account.
  - not an individual.
  - owned and used by daemons to perform system tasks.
- Remember that an account is more than just a login name:
  - it's an environment!

Slide 218: Background—So what are users?

What is an account?

Accounts provide a number of services on a Unix system. They provide predominantly a way for the system to distinguish between different people who use the system by allocating each user a unique username and password so that they can authenticate themselves to the system.

In most cases an account is a particular individual or a user of the system who can log in, edit files and run programs. However some accounts are not used by individuals at all. These accounts are system accounts and are generally used by system daemons to obtain access to files in order for them to carry out their tasks.

It must be remembered that an account is more than just a name; it relates to all the files, resources and information belonging to that user on the system.

As a system administrator it is your job to create and manage accounts.

- Accounts are stored in `/etc/passwd`:
  - one line for each account.
  - colons are used to delimit the 7 fields.
- `dnb1002:x:500:500:Darran Bryant:/home/dnb1002:/bin/bash`
  - the user's login name or the name of the account.
  - the password place holder.
  - the user indentification number, called the *UID*.
  - the group identification number, called the *GID*.
  - a description of the account.
  - the home directory.
  - the login shell.

Slide 219: The Password file

- `/etc/shadow`
  - one line for each account.
  - 9 colon-delimited fields.
- `dnb1002:$1$TksLmO0im7eoAUxP1:10933:30:1000:7:10:10957:134550524`
  - readable only by root for security reasons.
  - stores encrypted password for each account.
  - optional password aging mechanism.

Slide 220: The Shadow file

The password file

Every account on the system is held in text file called `/etc/passwd`. Here is an example `/etc/passwd` file:

```
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:
daemon:x:2:2:daemon:/sbin:
adm:x:3:4:adm:/var/adm:
lp:x:4:7:lp:/var/spool/lpd:
sync:x:5:0:sync:/sbin:/bin/sync
dnb1002:x:500:500:Darran Bryant:/home/dnb1002:/bin/bash
```

The file has one line per account and is divided into seven colon-delimited fields. Lets take the following entry :

```
dnb1002:x:500:500:Darran Bryant:/home/dnb1002:/bin/bash
```

The fields have the following meaning:

`dnb1002`	The user's login name or the name of the account.
x	The password place holder.
500	This is a unique number, called the *UID*, it is used by the system to uniquely identify each account.
500	The group identification number, the *GID*, is an integer that specifies which group the user belongs to.
Darran Bryant	A description of the account. Usual practice is to put the user's name in this field .
`/home/dnb1002`	The home directory.
`/bin/bash`	The login shell.

- `dnb1002:$1$TksLmO0im7eoAUxP1:10933:30:1000:7:10:10957:134550524`
  - user login name.
  - encrypted password.
  - last day the password was changed since 1st Jan, 1970.
  - days until change allowed.
  - days after which the password must be changed.
  - the number of days a warning is issued before the password expires.
  - days after the password expires that account is disabled.
  - days since 1st Jan, 1970 that account will be disabled.
  - reserved for future use.

Slide 221: The password aging mechanism

- The default password aging values are set in two different files:
- `/etc/login.defs`

  `PASS_MAX_DAYS`—maximum days a password may be used.

  `PASS_MIN_DAYS`—minimum number of days allowed between password changes

  `PASS_WARN_AGE`—number of days warning given before a password expires.
- `/etc/default/useradd`

  `INACTIVE`—days after the password expires that the account is permanently disabled.

  Default is `INACTIVE=-1`.

  `EXPIRE`—date when the user account will be disabled (YYYY-MM-DD).

Slide 222: Setting the password aging values

The `/etc/shadow` file

In the past the encrypted password was stored in the `/etc/passwd` file. However in almost all recent Unix implementations including Red Hat Linux this entry has moved to the shadow password file, `/etc/shadow`. This file is only readable by root and stops potential hackers from trying to crack the encrypted password entry. The `/etc/shadow` file also has additional flags to provide an optional password aging mechanism.

The file has one line per account and is divided into 9 colon-delimited fields. Here are some lines from the `/etc/shadow` file:

```
root:$1$.iOZeMG$K3klp1zA3vy4fwp23.FbO/:10920:0:99999:7:-1:-1:134549444
bin:*:10918:0:99999:7:::
daemon:*:10918:0:99999:7:::
adm:*:10918:0:99999:7:::
lp:*:10918:0:99999:7:::
sync:*:10918:0:99999:7:::
shutdown:*:10918:0:99999:7:::
nobody:*:10918:0:99999:7:::
xfs:!!:10918:0:99999:7:::
dnb1002:$1$b7dPLL9M$1Ac6HFNURAabQ6vK50B.U.:10920:0:99999:7::22281:134550524
```

The default password aging settings are defined in both the `/etc/login.defs` and the `/etc/default/useradd` files.

- when files are created they are automatically owned by both a user and a group.
- `rw-rw-r-- dnb1002 staff /home/dnb1002/voicebox`
- `/etc/group` is the file which defines groups.
  - `dnb1002:x:500:800:Darran Bryant:/home/dnb1002:/bin/bash`
  - `staff:x:800`
  - `dnb1002's` *primary* group is `staff`
- the group that owns the file is the primary group of the user that created it.
- Why groups?
  - allows files be organised in a more flexible manner.
  - allows an arbitrary number of users to share files transparently.

Slide 223: Groups, file ownerships and the primary group

Groups and `/etc/group`

Under Unix files are owned by both a user and a group. The motivation behind this is to allow files to be organized in a more flexible manner. For example the group mechanism can allow a selection of files to be accessible by a completely arbitrary collection of users. The classic example is a group of users working on the same project. By creating a group for the project users can then share project files transparently.

When a user creates a file, the file is owned by both the user and the primary group of the user that created it. However if the file is created in a directory that has its group bit set then the primary group of the user who created the file is ignored. Instead, any files or subdirectories created in the shared directory will be owned by the user who created it *but* will be assigned the group ownership of the directory.

The `/etc/group` file is an ASCII file which defines the groups to which users belong. There is one entry per line, and each line has the following format:

group-name:password:GID:user-list

group-name:	the name of the group
password:	the password field
GID:	the numerical group ID
user-list:	the users that belong to the group, separated by commas

Each user on the system is assigned to at least one group. See the User Private Group scheme later in this section.

- Manually managing accounts is cumbersome.
- For example to manually add a user:
  - edit `/etc/password`, `/etc/shadow` and `/etc/group`.
  - create the user's home directory.
  - copy the global administration files from `/etc/skel`.
  - change the ownership on the home directory and the files within it to be owned by the new user.
  - give user new password.
- So, Red Hat Linux provides some tools:
  - `useradd`, `userdel`, `usermod`.
  - `redhat-config-users`.

Slide 224: Managing users

Adding users

There are a number of ways to create user accounts. To create an account manually you would have to perform a number of tasks (see the above slide).

Fortunately there are some scripts that can help you with this.

`useradd dnb1002`

  - Creates an account using the values from the system files.
  - `/etc/default/useradd`, `/etc/login.defs`

`useradd dnb1002 -s /bin/csh`

  - Use command line arguments to override default values.

- With no options `useradd -D` displays the current defaults values.

Slide 225: `useradd`

Adding users with `useradd`

The `useradd` command when invoked without the `-D` option will create a new user account using the values specified on the command line and merge them with the default values from the system. Default values are stored in the `/etc/default/useradd` file. When a user is added, the user's account details are entered into the `/etc/passwd` file, a home directory is created and the default files are copied over from the `/etc/skel` directory. A private group is also created for the user.

To add a user with the username `dnb1002` using the system defaults simply run the command:

```
# useradd dnb1002
# passwd dnb1002
```

If you wish to create an account that has different values from the defaults then you can override the system defaults by specifying values on the command. For example:

```
# useradd dnb2005 -d /export/home -e 2005-01-01 -G staff,emacs
```

This command would add `dnb2005`'s details to the `/etc/passwd` file and also add him to the staff and emacs groups. His account will also expire on the 1st of January 2005. All the other account details will be set to the values defined in the `/etc/defaults/useradd` file. The `useradd` command can also be used to change the default values stored in `/etc/defaults/useradd` by using the - D option. For example:

```
# adduser -D -b /export/home -s /etc/csh
```

This would change the default home directory and default shell for all users by modifying the values in the `/etc/default/useradd` file.

If no options are specified, `useradd -D` displays the current default values.

- `# userdel dnb1002`
  - Removes an account from the system.
- `# userdel -r dnb1002`
  - The `-r` option removes the user's home directory as well.
- Can not remove an account if the user is logged in or if the user has a process running.

Slide 226: `userdel`

- `usermod` modifies the system account files to reflect the changes that you specify on the command line.
- The command has many options:

  -c: the new value of the user's password file comment field.
  -d: the user's new home directory.
  -e: the date on which the user account will be disabled.
  -f: the number of days after expirer that the account is permanently disabled.
  -g: the group name or number of the user's new initial login group.
  -G: the list of supplementary groups which the user is also a member of.
  -l: the user's login name
  -s: the name of the user's new login shell.
  -u: the numerical value of the user's ID.
  -L: locks the user's account.
  -U: unlocks the user's account.

Slide 227: `usermod`

Removing users with `userdel`

To manually delete user `dnb2005` you would need to, remove the user from the `/etc/passwd`,`/etc/shadow` and the `/etc/group` files, then delete the user's home directory. The `userdel` command will perform this task for you. However by default the user's home directory is not deleted. To delete the user's home directory use the `-r` option.

`userdel -r dnb2005`

The `userdel` command will not allow you to remove an account if the user is currently logged in.

Modifying users with `usermod`.

The `usermod` command modifies the system account files to reflect the changes that you specify on the command line. The command allows you to change almost any information concerning a user's account details.

Some useful options are:

`usermod -L dnb2005`

This locks an account and stops the user from logging in.

`usermod -U dnb2005`

This unlocks an account.

- The User Private Group (UPG) scheme:
  - nothing new and it is used on many flavours of Unix.
  - *It is the default behaviour!*
- What happens is this:
  - each new user added to the system triggers the creation of a new group.
  - the new group is given the same name as the user's username.
  - the new group is the primary group for the new user.
  - only the new user is a member of the group created.

Slide 228: UPG - User private group scheme

- `useradd david`
  - `david:x:500:505:A user for the course:/home/david606:/bin/bash`
  - `david:x:505`

Slide 229: User private group scheme

The Private Group scheme

On a Red Hat Linux system the creation of a new user triggers the creation of a new group. This is called the User Private Group (UPG) scheme. The scheme is implemented by allocating each user their own private group, of which they are the only member, and by setting the default `umask` for all users to be 002.

The `umask` is used to set the initial file permissions on newly-created files. Specifically, permissions in the `umask` are turned off when the file is created. For example the common `umask` default value of 022 results in new files being created with the permissions of `rw-r--r--`. Using a `umask` of 002 the permissions are `rw-rw-r--`.

- So why the UPG?
  - management of groups and shared directories is easier and nearly transparent.
- How to give multiple users write access to files in a shared directory:
  - create an auxiliary group in `/etc/group`.
  - add the trusted users to the auxiliary group.
  - create the shared directory.
  - set the group ownership on the shared directory to be that of the auxiliary group.
  - allow read, write and execute access to the directory for the auxiliary group.
  - stop write access to the shared directory for everyone else.
  - add the set-group-ID bit to the shared directory.

Slide 230: UPG - Sharing directories

The Private Group Rationale

Many people question the private group scheme. To illustrate the scheme let us consider creating a shared directory to allow a privileged group of people access to the files within it while restricting access to others who are not in the privileged group.

For this example we will create a shared directory called `/shared/projects` and have a privileged group called `hackers`.

The overall plan is to allow members of the `hackers` group write access to the files in the shared directory but to deny write access to users who are not in the `hackers` group.

First we must create the shared directory and the privileged group:

```
# mkdir -p /shared/projects
# groupadd hackers
```

We would then need to change the group ownership on the directory to be owned by the `hackers` group:

```
# chgrp hackers /shared/projects
```

This directory also needs the permissions changed so the members of the `hackers` group can write into it:

```
# chmod g+w /shared/projects
```

We now need to force all files that are created in this directory to have the group ownership set to the privileged group. We can do this by setting the set-group-ID bit on the shared directory:

```
# chmod g+s /shared/projects
```

From now on everything created in this directory will have the group ownership of `hackers`. This will allow members of the `hackers` group to edit the files with the `/shared/projects` directory.

Add the trusted users to the `hackers` group in the `/etc/group` file.

- *Why the UPG?*
- Why create a unique group for each user?
- The default `umask` is 002.
  - `rw-rw-r--`
  - this allows members of a shared group to edit files in a shared directory
- *BUT* the `umask` affects all files - including those in user's home directories.
- *So*, User Private Group Scheme (UPG):
  - stop group writeable file in user's home directories being an issue.
- Because the user and group ownership on a file are the same.

Slide 231: Sharing directories - Why it works

The above example illustrates two points: Since we are using an `umask` of 002 each user in the group will be able to edit the files created by other users. If we were using a `umask` of 022 then this wouldn't be the case! However, the `umask` of 002 affects all files created by the user, including those in their home directories. The private group scheme protects these files being writeable by anyone else because the only member of the group is the user who owns the files.

- `/etc/skel`
  - provides global management of the user's home directory.
  - any directory or file can be added to `/etc/skel`.
- `/etc/bashrc`
  - the global setup file for the `bash` shell.
  - normally contains functions and aliases.
- `/etc/profile`
  - contains system-wide environmental variable settings.
  - executed by all `bash` and `ksh` users.
- `/etc/profile.d/`
  - scripts called from `/etc/profile`.
  - used because it is more reliable for a RPM to add a script to a directory then to edit the lines in `/etc/profile`.

Slide 232: System wide configuration file

The initial user environment

The global management of the user's initial environment is accomplished through the /etc/skel directory. Each file or directory in /etc/skel is copied into a user's home directory when an account is created with either useradd or the User Manager utility.

However it is usually a good idea to keep the /etc/skel directory small and whenever possible to add global configurations to the /etc/profile file or the /etc/profile.d/ directory.

The following system wide configurations files are quite commonly found within /etc/skel:

```
.Xdefaults
.bash_logout
.bash_profile
.bashrc
```

Any directory or file that you wish to have included in the standard users initial setup can be added to /etc/skel.

The User Manager utility

The User Manager utility (see figure 32) allows you to view, modify, add, and delete local users and groups. The utility can be started by running the command redhat-config-users at a shell prompt or by starting it from the desktop (Main Menu Button → System Settings → Users & Groups).

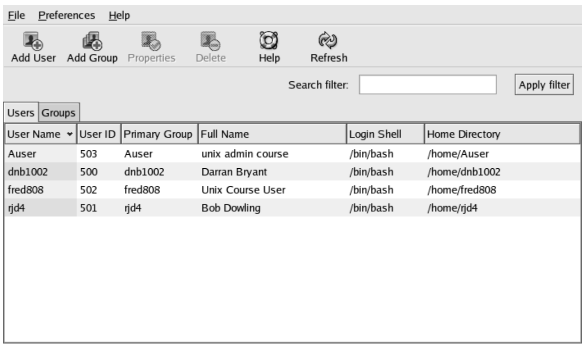


Figure 32: The User Manager utility

- Create 3 new users called `staff1`,`staff2` and `student` with the `useradd` command.
- Create passwords for each of the accounts you created above.
- Create an extra group called `hackers`
- Add `staff1` and `staff2` to the `hackers` group in `/etc/group`.
- Create a shared directory called `/shared/directory/` so that `staff1` and `staff2` can create and edit files in this shared directory but the `student` account cannot.
  - remember to set the group ownership on `/shared/directory/` to be owned by the `hackers` group.
  - remember to add the set-group-ID bit to the shared directory. Use the command `chmod g+s /shared/directory/`.
  - remember to change the premissions on the directory so that the `hackers` group can write into it.
- cd `/shared/directory/`
- Look at the permissions on the directory, notice the set-group-ID bit and the group ownership.
- Log in as `staff1` and create a file called `example1` in `/shared/directory/`.
- Examine the file permissions on this file. Notice the group ownership.
- Log in as `staff2`
- Edit `/shared/directory/example1` and save the file.
- Examine who owns the file now.
- Log in as `student`, you should not be able to change the `/shared/directory/example1` file.

Slide 233: Exercise

- Change the permissions on the shared directory so that only the owner of the file within the shared directory can delete it.
- Stop the user `student` from listing the files within the shared directory.

Slide 234: Additional Exercise

9 Practical filesystems

9.1 Introduction

This section will deal with the topic of filesystems, not from the viewpoint of the layout of the files, but from a somewhat lower level. The aim of this section is not to give you all the answers on filesystems, but to help you find the answers to the specific questions that you might have in the future.

Unfortunately this topic consists of lots of interlocking aspects, which means that you need to know everything before you can learn anything else. For this reason information will be given in a potentially bizarre sequence.

9.2 Hardware

Before talking about filesystems it is useful to know about the devices on which these filesystems exist. Hard drives typically contain a central spindle. Mounted on this spindle will be a number of platters (glass disks coated in a magnetically responsive material, similar to that found on video tapes). Each surface of the platter has a head associated with it. The head is used for reading and writing, and is positioned with extreme accuracy by a magnetic coil. The data is arranged into tracks, with each track forming a ring on the surface centred at the spindle. All the tracks at the same radius from the spindle together are called a cylinder. Tracks are also subdivided into arcs, each of these arcs is called a sector.

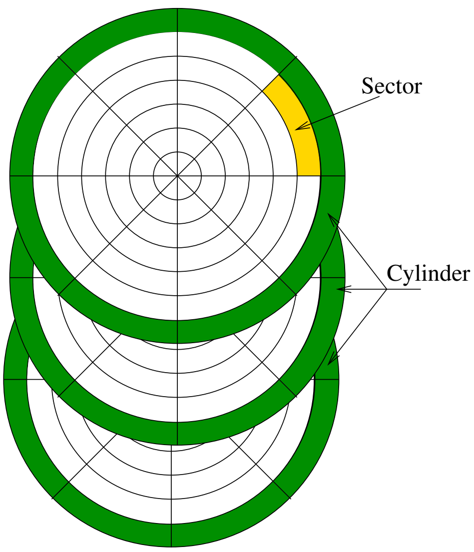


Figure 33: The layout of a hard drive

In slide 241 there is an example of how many heads cylinders and sectors a disk claims to have. These are completely fictitious, and the electronics on the hard drive make the figures up, and translate these numbers into reality.

9.3 Filetypes

Unix has many mottoes to which it adheres to some extent or other. One of these is "Everything is a file, except things that are not." Many of the traditional exceptions have been removed with the introduction of `/proc`. In order to make everything a file there needed to be several different filetypes (listed in slide 235). The first three you should have all come across already. The next two, block and character devices, are special files traditionally found in `/dev/` which are hooks into the kernel, and which represent devices such as hard disc drives (block) or terminals (character). Few things use named pipes by default but one important exception is `init` which can be controlled via `/dev/initctl`. They can be used by other things, and behave as a buffered pipe. The last device is a socket, which is even stranger than a named pipe. It is the filesystem representation of a "Unix-domain socket". The programmers amongst you can look at `socket`(2) and `unix`(7). However when the program using the socket exits, the socket becomes an orphan file, with no significance as far as the kernel is concerned. If another process wants to use a socket file of the same name the old one must be removed and a fresh one created.

File type	Example
regular file	`/dev/MAKEDEV`
directory	`/dev/`
symbolic link	`/dev/mouse`
block (buffered) special	`/dev/hda`
character (unbuffered) special	`/dev/ttyS0`
named pipe (FIFO)	`/dev/initctl`
socket	`/dev/log`

Slide 235: Filetypes

9.4 What is in a name?

As you can see if you do an `ls -ld` on the files in slide 235 they have different first letters in the “permissions” field, each letter representing a different file type. If you look at the block and character devices you will see that rather than a single number in the “size” field there are two numbers. These are called major and minor numbers, and the type (block/character) and major and minor numbers identify a system device uniquely. This is significant, whereas the name is not. It would be possible to make a block device somewhere else with major number 3 and minor number 0, and that too would refer to the primary master IDE device. See `man mknod`(1) and `info mknod` for more details.

9.5 Partitioning disks

Before you can create files you need somewhere to store them. The first stage is to create a partition on a disk onto which to put the filesystem. You will have already used Disk Druid, Red Hat’s own variant on `fdisk`, to partition and organise a disk. Even if you were to want to use the somewhat feature-lacking Disk Druid this is not an option since, as has been said, it is not available outside the installation procedure. So a different tool is needed to achieve a similar function. Slide 236 details the tools that I have found.

`fdisk`	The original
`cfdisk`	A curses/graphical version *[missing]*
`sfdisk`	Even more obscure than `fdisk`. Can work non-interactively
`parted`	Yet another partition editor

Slide 236: Disk partitioning tools

9.6 `fdisk`

The tools listed in slide 236 are all loosely based on the `fdisk` that comes with DOS, however they can deal with far more situations. You will find differing advice on which program you should use. In the man page for `fdisk`(8) it recommends that you use `cfdisk` even though this program is no longer distributed by Red Hat. For the purposes of the course the recommended program is `fdisk`. However, during the practical feel free to experiment with other tools. At other times use whichever tool you are most comfortable with provided it does what you need it to do. If you wish to do things in a different way, or do something more complex, then it is useful to be aware of the other tools. Also note that it is possible to use `fdisk` during the installation. If you dislike Disk Druid then there is no real issue with using `fdisk` instead.

9.7 Things you need to know to use `fdisk`

To use `fdisk` you have to know which is the right block device to use. There are rules which Linux uses to label devices, see slide 237, slide 238 and slide 239, which you can use to work out which one applies to the disk that you need to modify.

Information mostly repeated from the section, "Installing Red Hat Linux".

`/dev/hda`	Represents the primary IDE master
`/dev/hdb`	Represents the primary IDE slave
`/dev/hdc`	Represents the secondary IDE master
`/dev/hdd`	Represents the secondary IDE slave

Slide 237: IDE hard drive devices

`/dev/hd[a-t]`	Represents the IDE devices on other controllers if they exist (Note that the default only goes to `h`)
`/dev/sd[a-z]`	Represent the SCSI disks.
`/dev/sd[a-i][a-z]`	Represent further SCSI disks up to `sddx` (256 disks).

Slide 238: Generic Hard drive devices

Note: Since SCSI disks are numbered `a`, `b`, `c` etc. in controller and SCSI ID order, adding a new disk can shift their device numbers around, so you need to be careful when adding SCSI disks.

`/dev/hda1`	The first partition on the primary master IDE device.
`/dev/hdb5`	The first logical partition in an extended partition on the primary slave IDE device.
`/dev/[hs]d[a-z]`*n*	Represents partitions on the hard drives.

Slide 239: Partition devices

Alternatively if some other part of the device is already in use then you can use that information to work out the parent device. The example in slide 240 shows that `/` is on a partition which is on the disk represented by the device `/dev/hda`.

`/` is mounted from `/dev/hda6` (see the output of `df /`).
The parent device of `/dev/hda6` is (somewhat predictably) `/dev/hda`.
Therefore `/dev/hda` is the device which refers to the hard drive which `/` is on.

Slide 240: Working out which device to use.

9.8 Using `fdisk`

On IDE-based systems there will almost certainly be a primary master, so the example in slide 241 should work on most normal PCs. The slide gives an example output from "`fdisk -l /dev/hda`" on a system.

- The Device column lists the devices each partition links up with.
- The Boot column flags which partition a standard bootloader stored in the master boot record (MBR) would boot from. If GRUB is stored in the MBR it will ignore this. The boot loader which comes with DOS or Windows checks for this flag however, which means it is (potentially) useful in multi-boot systems
- The Start and End columns show the start and end cylinders for each partition. The size of each cylinder is defined by the number of blocks (which is the number of heads times the number of sectors) and the size of each block. In the example in slide 241 there are 255 heads, 63 sectors and 2434 cylinders. This means that there are 2434 cylinders of 16065 blocks each (63×255). Each block is 512 bytes in size, so there are 2434 cylinders of 8032 kb (16065×512 bytes) which is 19549888 kb (8032×2434).
- The Blocks column shows how many 512-byte blocks are used by a partition. The trailing + signs indicate that rounding has taken place, and that the actual value is slightly more. It is possible to get a - sign as well which means that the value is slightly less.
- The Id column displays the numerical ID for the filesystem type in that partition. (See the "*l*" key in slide 242 and slide 243)
- The System column gives the name for the ID in the previous column.

```
# fdisk -l /dev/hda

Disk /dev/hda: 20.0 GB, 20020396032 bytes
255 heads, 63 sectors/track, 2434 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

   Device Boot    Start       End    Blocks   Id  System
/dev/hda1   *         1        13    104391   83  Linux
/dev/hda2            14       523   4096575   83  Linux
/dev/hda3           524       778   2048287+  83  Linux
/dev/hda4           779      2434  13301820    5  Extended
/dev/hda5           779       905   1020096   83  Linux
/dev/hda6           906      1032   1020096   83  Linux
/dev/hda7          1033      1097    522081   82  Linux swap
```

Slide 241: Example output from `fdisk -l`

By far the most likely options for you to deal with are shown in slide 243.

```
# fdisk /dev/hda
Command (m for help): m
Command action
   a   toggle a bootable flag
   b   edit bsd disklabel
   c   toggle the dos compatibility flag
   d   delete a partition
   l   list known partition types
   m   print this menu
   n   add a new partition
   o   create a new empty DOS partition table
   p   print the partition table
   q   quit without saving changes
   t   change a partition's system id
   u   change display/entry units
   v   verify the partition table
   w   write table to disk and exit
   x   extra functionality (experts only)

Command (m for help):
```

Slide 242: Options on `fdisk`

m	print the menu of options	Effectively the *help* key.
p	print the current partition table	So you know where you are at the moment.
d	delete a partition	This only takes effect when you *w*rite out the details.
n	create a new partition	Also only takes effect when you *w*rite out the details.
q	quit without saving changes	For when you make a mistake, also [Control]+[C]
w	write table to disk and exit	For when you are sure that you have it right.
a	make a partition the bootable	Useful for dual booting.
l	list known partition types	List possible partition types for different filesystems
t	change a partition's system id	Select something other than the default ext3 (Linux) partition type

Slide 243: `fdisk` options explained

9.9 Creating a partition

Slide 244 shows a new partition being created on a small blank disk. It sets the first cylinder of the new partition to be the first free one, which in this case is also the first one. Then it sets the size to be a given number of megabytes (alternatively the size could have been specified by giving the last cylinder for the partition). The only step that is not shown is the final one, "**w**" to write the new partition table out.

```
Command (m for help): n
Command action
   e   extended
   p   primary partition (1-4)
p
Partition number (1-4): 1
First cylinder (1-648, default 1): 1
Last cylinder or +size or +sizeM (1-648, default 648): +48M
```

And then to check that the partition was created as expected

```
Command (m for help): p
Disk /dev/hdb: 64 heads, 63 sectors, 648 cylinders
Units = cylinders of 4032 * 512 bytes
   Device Boot    Start       End    Blocks   Id  System
/dev/hdb1             1        25     50368+  83  Linux
```

Slide 244: Creating a new partition example

Rather than add a new disk to the system we will add a new partition to the disk which is already installed. If you have obeyed the installation instructions carefully then `fdisk` on your computer will show that there is some spare space on your system which can be used to create a new partition. So use `fdisk` on the right device, and have a look at the options using the "**m**" key as in slide 242. A later exercise in this section will get you to move `/var` into there, so the partition should be bigger than the current `/var`. There should currently be plenty of space spare on the disk.

- Experiment with `fdisk`

  do not write anything out unless you are sure that it will not delete anything important.

  [Control]+[C] is your friend.

- When you are happy with the options available under `fdisk` create a spare partition which is larger than the current `/var`.
- Note: Because this exercise uses a partition on an active disk a reboot is needed to make this take effect.

Slide 245: Exercise: Create a new partition

9.10 Filesystems

Now you have a blank partition on which you need a filesystem. There are large numbers of filesystems which Linux recognises and can manipulate. The (current) standard for Red Hat Linux is the "ext3" filesystems, and that is what this section concentrates on. "ext3" is the "third extended filesystem". This has one major enhancement over "ext2", namely it is a logging filesystem. This means that it will write changes to a log before writing to the filesystem itself. The significance is that it is easy to maintain a consistant filesystem, of which more in section 9.17. "ext2" itself which has several enhancements over the original "extended filesystem" which in turn had large improvements over the Minix filesystem that Linus Torvalds used in the original release. Section 9.16 looks in more detail at filesystem concepts, and section 9.18 looks at even more advanced filesystems. "ext3" is such a small internal change over "ext2" that many of the tools you will use still have 'e2' included within the name. This is merely an accident of history.

9.11 Creating a filesystem

Filesystems are created with the `mkfs` command, although this is really just a front end for programs like `mkfs.ext3`, `mkfs.minix` and `mkfs.msdos`. The manual pages list the options as `-V`, `-t` *fstype* and *fs-options*. The `-V` option will show what commands are being executed, and if given twice will not execute the commands, but only print them. The `-t` option takes a parameter which indicates which filesystem type is to be used, this results in the right version of `mkfs` being called. The *fs-options* are passed onto whatever program is executed. The manual page for `mkfs.ext3`(8) lists all the option that it takes, most of which you are unlikely to ever need to use. If you want to know more details then read the manual page, however some features that you should be aware of options to are:

- Check the partition for bad blocks
- Set the volume label for the partition.
- Change the total number of inodes or the percentage of inodes.
- Change the percentage of reserved blocks.

Normally `mkfs /dev/`*`partition`* will suffice to create a sensible filesystem, and unless you are sure that you wish to stray from the norm you are advised not to change the default options.

- Use `fdisk -l` and `df` to check which partition to create the filesystem on.
- Use `mkfs` to create a filesystem on the partition that you created

  MS-DOS calls this `formatting` a partition.

  If you choose a partition that is currently mounted and in use then you will be warned.

  Warning: It is possible to destroy data if the partition table has overlaps.

Slide 246: Exercise: Create a filesystem

9.12 Mounting filesystems

Now you have a filesystem which is on a partition, but you cannot access it. The filesystem needs to be mounted somewhere so that it can be accessed. The standard form of the mount command is

\# **`mount -t`** ***`vfstype device dir`***

The *vfstype* is one of the supported filesystems. These include, but are not limited, to *ext2 ext3 iso9660 msdos vfat*. More details can be found in `mount`(8) and in `/usr/src/redhat/BUILD/linux/fs/filesystems.c` if you have the sources installed.

The *device* is the entry in `/dev` refering to the relevant partition. The *dir* is the directory which will have the filesystem mounted onto it. It should be noted that under Linux any existing contents of the underlying directory tree are hidden from view when something is mounted on a directory.

The manual page for `mount(8)` gives a long, if not comprehensive, list of options to mount. Some of the more interesting options are detailed in slide 247

Option	Effect and use of option
`async`	All I/O is done asynchronously. This is fast, but slightly increases the chances of a filesystem corruption in the case of a system crash.
`auto`	Make `mount -a` mount this filesystem. Only really useful in `/etc/fstab` (see section 9.13)
`defaults`	Sets all the defaults (`rw,suid,dev,exec,auto,nouser,async`)
`noexec`	Do not allow execution of any binaries on the mounted filesystem.
`nosuid`	Do not allow set-user-ID or set-group-ID bits to take effect.
`remount`	Re-mount a filesystem which is already mounted. Allows changes in mount flags without unmounting a filesystem, which may not be possible at this time.
`ro`	Make the filesystem read only.
`user`	Allow a normal user to mount the specified filesystem, only really useful in `/etc/fstab` (see section 9.13)
`owner`	Allow a normal user who *owns* a device to mount that filesystem. Particularly useful with CD-ROMs and floppies. Only really relevant in `/etc/fstab` (see section 9.13)
`noload`	ext3 specific option to prevent the journal being played when mounting an ext3 filesystem.

Slide 247: Interesting mount options

- Mount the newly created filesystem on `/mnt` (a traditional location).

Slide 248: Exercise: Mount the filesystem

It also would be helpful at this point to know how to unmount a filesystem. The command is `umount`. By and large using `umount` with the mount point or the device name as an argument will do exactly what you would want it to do. The manual page for `umount(8)` covers what few options it has. When unmounting a filesystem you can get the message that the "`device is busy`". This means that something is using a resource on the filesystem. Often this will be an open file, although it can be a working directory, or a number of other possible things. Normally `fuser` and `lsof` can be used to track down the offending processes. One hint I will include is that if you have a filesystem mounted under the filesystem that you are trying to unmount then only `/etc/mtab` and the output from `mount` will give you a clue as to why you cannot unmount the filesystem. When the mount table has become corrupted then it is almost impossible to track down the cause of the busy message.

9.13 `/etc/fstab`

Being able to mount filesystems is all well and good, although naturally it would be useful to be able to have some filesystems mounted at boot time, and to be able to mount things without having to remember the specific options that are needed for that particular device.

Example entries in `/etc/fstab`

```
LABEL=/boot   /boot   ext3   defaults   1  2
/dev/hda7     swap    swap   defaults   0  0
```

Field	*Description*	*Example*
1	The block device (or volume label) to be mounted	`/dev/hda7`, a block device.
2	The directory to mount on	`/boot`, a directory
3	The filesystem type	`ext3`, `swap`, `ignore`, `iso9660`, `nfs`
4	Comma separated list of mount options	`ro,noauto,user,owner` (see `mount`(8))
5	Number used by `dump`	defaults to 0, meaning do not dump
6	Number in the boot time `fsck` order	defaults to 0, meaning do not fsck on boot

Slide 249: `/etc/fstab` explained

If you look at `/etc/fstab` you will see how partitions are arranged on your current system. Included in slide 249 are details extracted from `fstab`(5). The file `/etc/fstab` is used by the mount command to automate aspects of its operation, especially at boot time. The file is also used to declare which partitions are to be used as swap, see `swapon`(8). Additionally `/etc/fstab` is used to control the sequence for `fsck` at boot time, and potentially `/sbin/dump` is partially controlled by the sixth field. If you need to add a partition to be available for use by the system then this is where you need to add it. See `mount`(8) on how to override any options in `/etc/fstab` on the command line, or how to mount devices which have no entry in `/etc/fstab` by specifying all the options. Note that you need to be careful with the options, and it is strongly recommended that you check them carefully.

Volume labels have existed for some time, but Red Hat have only started using them in recent releases. Unfortunately they are unhelpful in most respects. Knowing the volume label gives you no clues as to what the device is. In order to be able to manipulate a partition you need to know what the device is.

9.14 `tunefs`

In slide 249 you can see how a volume label can (and is) used to specify which partition the rest of the config line applies to. If you failed to set the volume label when using `mke2fs` to create the filesystem (which you will have done, since the instructions didn't tell you to) then you can add (or change) a label with `tune2fs`. Many other different parameters can be altered with this program, but it is rare that you will need to change any of these away from the defaults.

- Set the label for the newly created filesystem to be `FUBAR`.
- Confirm that you have set this name.
- Now change the name to be `/var` and confirm the change.

Slide 250: Exercise: Set and change the volume label

There are other commands which manipulate the same metadata on the disk. One such command is `e2label` which appears to have a subset of functionality of the `tune2fs` command.

9.15 Backups

We all know that disks fail, or mistakes happen, or a hacker comes along and deletes some files. For these, and many other reasons backups are a good idea. There is no time or space to cover backups in depth: it could take up another section in its own right. I include the URL `http://www.taobackup.com/` which I urge you to read. It is (supposedly) humorous, and vaguely attempts to sell a product, but it has some important points in it.

In the experience of this system administrator, almost all backup solutions in the Unix world end up being 'home brew'. This is largely because few things are flexible enough to cope with the generic cases, and the ones that are flexible enough are horribly complicated to use. `Amanda` is one program that makes a reasonable attempt at being flexible enough, but can be difficult to set up. Once it has been set up its features are almost certainly better than something you might create yourselves. Details can be found at `http://www.amanda.org/`.

If you do end up creating your own setup then there are several possible programs to use as building blocks. You could use `tar`, because it is simple, and you already know how to use it. You could use `cpio`, because it is relatively simple, but unlike `tar` works (mostly). Or you could use the right program, which is `dump` along with its counterpart `restore`. This has a number of features that make it useful. It can do backups of filesystems, no matter how deep the filesystem tree is, it can access the area to be backed up as a user other than root, it can back up onto multiple tapes and over the network, and it can do incremental backups. Oh, and it was *designed* to do backups of filesystems.

When dumping, a level between 0 and 9 can be specified. A level 0 dump will guarantee that the entire filesystem is backed up. Higher dump levels are incremental, and back up all the files which are new or have been modified since the last dump of the same or lower level. The manual pages for `dump`(8) gives much more information on its usage.

If a dump is taken whilst the data in that filesystem is changing then some corruption could occur. This is why backups should, if at all possible, be taken when the number of running programs which could make a modification to the system is as small as possible. This normally means single user mode. On many systems this is not an option, so a small risk is taken and dumps are taken during normal system operation, albeit in the middle of the night when hopefully not too much is happening.

In this exercise you will move the partition `/var` is on.

- Read the man page for `dump`(8). Work out how to dump `/var` into a file.
- Read the man page for `restore`(8). Work out how to restore from a file into a directory.
- Drop to single user mode with `/sbin/telinit 1`.
- Use `umount` to unmount `/var`.
- Do a level-0 `dump` into `/home/var-dump`.
- If necessary `umount /mnt`, and then `mount` the blank filesystem onto `/var`.
- Use `restore` to recover from the file in `/home/var-dump` into the new `/var`.
- Edit `/etc/fstab` to reflect the new `/var`.
- Make sure all the volume labels are sensible.
- Come back into your previous runlevel (3 or 5) with another `/sbin/telinit`.

You should now have a larger `/var` than you started with. Note that at no time was the machine rebooted.

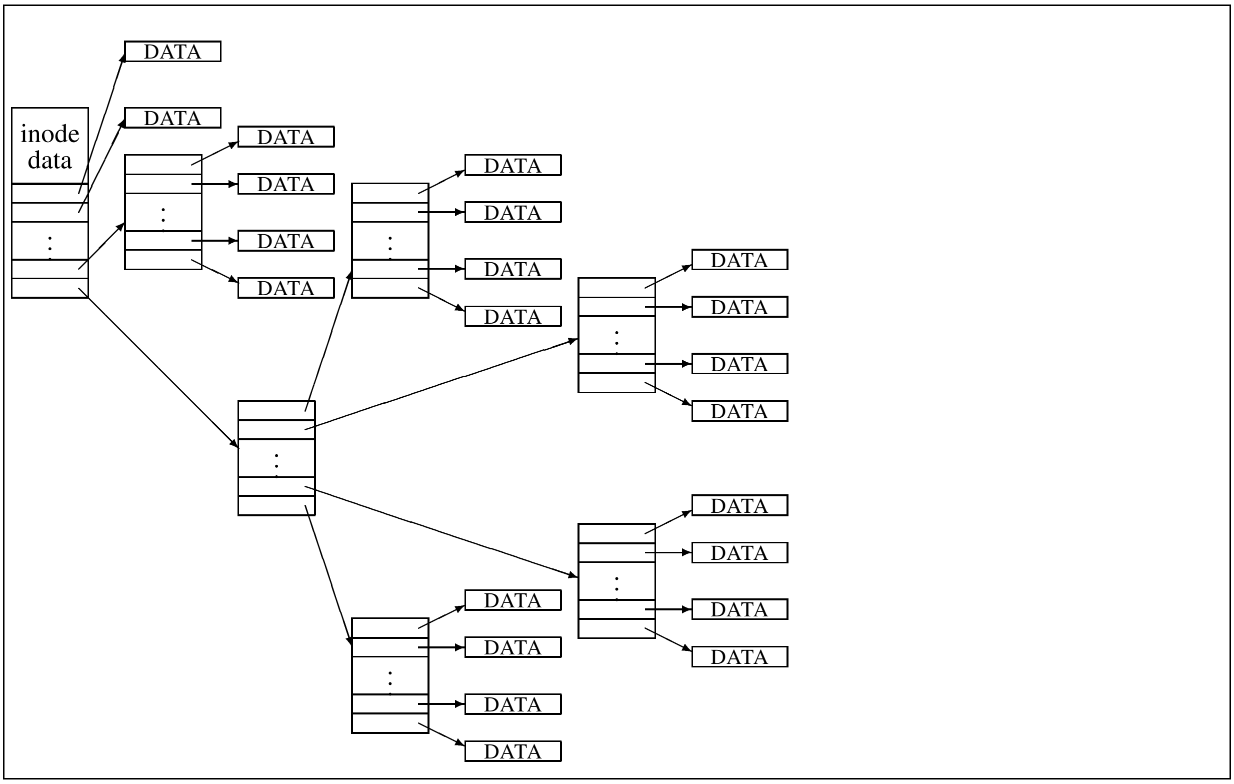
Slide 251: Exercise: Dumping and moving `/var`

9.16 Filesystems concepts

Although the ext3 filesystem does have a few extra features, it shares most of its concepts with most other Unix filesystems. Files are represented by *i-nodes*, which are data structures containing information about the file to which they refer. This information includes:

The Mode of the file (see section 9.16.1).
The Owner and Group of the file
The size of the file
The timestamps—N.B. there are multiple timestamps. (see section 9.16.2)
Link count (How many directory entries have references to this inode)
Block count (The number of blocks the file the inode refers to uses)
12 Pointers to datablocks (see figure below)
Indirect block pointer, pointing to a block full of pointers
Double indirect block pointer, pointing to a block full of pointers to blocks full of pointers
Triple indirect block pointer, pointing to a block full of pointers to blocks full of pointers to blocks full of pointers.

Most of the information in this section and a lot more can be found at `http://en.tldp.org/LDP/khg/HyperNews/get/fs/ext2intro.html`. This does not cover the ext3 extensions.



Slide 252: Inode pointer and data layout

9.16.1 Mode

The mode comprises the following bits of information encoded into a bit-stream: the type of file, the set-user-ID, set-group-ID and sticky bits, and the read, write and execute bits for the user, group and other.

You should all know the basics of Unix file permissions, much of which was given earlier in the course. However what may not be obvious is that there is a direct relationship between the inode and the permissions and ownership of a file. Since the permissions are stored in the inode, if there are multiple entries for that inode in directories then the "new" file will not only have the the same contents, it will also have the same ownership and the same permissions. It is the *same* file, with two different locations. This is a hard link, often used by binaries so that they have multiple names. For example `/sbin/mke2fs` is the *same file* as `/sbin/mkfs.ext3`.

9.16.2 Timestamps

There are three standard timestamps stored on the inode.

- The time of the last file access ("atime"), set whenever the file is opened, even in read only mode.
- The time of the last inode modification ("ctime"), set whenever some detail on the inode is changed, such as the permissions or the name.
- The time that the data in the file itself was last changed ("mtime").

All of these are of course unchanged if the *filesystem* is read only.

Additionally there is a fourth timestamp, the deletion time ("dtime") for the inode. This is used as part of a further hint when running `fsck` as to what the status of that inode/file is.

9.17 `fsck`

Much like `mkfs`, `fsck` will take the filesystem option and run the appropriate "real" command. The program that is normally run will be `fsck.ext3` (which is the same as `e2fsck`). Normally Linux will check on boot if a filesystem is dirty, and if it is then it will run `fsck -p` on that filesystem.

Given the difficulty of corrupting your filesystems in a predictable fashion in order to demonstrate how `fsck` works, it has been decided to cheat. You will now be exposed to some magic. Please ignore the magic and remember the lesson.

```
# mkdir /mnt-fsck
# mount nfs-uxsup.csx.cam.ac.uk:/local /mnt-fsck
# /mnt-fsck/Linux-course/wand start
```

Slide 253: Some magic for `fsck`

You will now have a new filesystem on a partition called `/dev/md0`. Any actions with `/sbin/fsck` can now be done using `/dev/md0` as the device to be checked. Do the exercise in slide 254. When you need to restart to try again then you can follow the instructions in slide 255

For the exercise try to fix the filesystem as best you can. Pay attention to the questions asked. Remember that deleting things will be destructive. Remember that you can also have multiple passes in fixing a file system if you need to. Good commands to try are:

```
# /sbin/fsck -p /dev/md0
# /sbin/fsck -b 8193 /dev/md0
# /sbin/fsck -y /dev/md0
```

If you want to know what the options mean, and what other options are available to you then have a look at `fsck`(8) and `e2fsck`(8).

In order to see how successful you were you will want to mount the filesystem with
`# /bin/mount /dev/md0 /mnt`.
If you want to try again then follow the instructions in slide 255

Slide 254: Exercise: Using `fsck` to fix a filesystem

```
# /mnt/Linux-course/wand restart
```

Slide 255: Some more magic for `fsck`

9.18 Advanced Filesystems

Since the traditional Unix filesystem was developed nothing much has changed in decades. However relatively recently a few features were added, and these are now making their way into mainstream. Red Hat are using the third extended filesystem (ext3) whilst other people are using the Reiser filesystem (reiserfs). Broadly these new filesystems have the same improvement, and use a log. The way this works is that instead of modifying the filesystem you keep a log of modifications that are made. Then, every so often you synchronise the filesystem with the log. The advantages that this offers are that, in theory, the filesystem should always be recoverable to a 'good' state, which reduces the time taken by fsck to recover. Also, it can offer speed improvements, especially where you have short lived temporary files which don't need to be properly created if they are destroyed again before the filesystem is synchronised against the log file.

Red Hat offered ext3 support as of the 7.2 release. The main advantage of this over the competition is that it is backwards compatible with ext2, and easy to convert an ext2 filesystem into ext3.

10 Security

- Security is a big topic
- I'll cover what I can
- Look on the Web for more

Slide 256: So you want to run a secure Linux system, do you?

Security is a horribly big topic, and one it can be disastrous not to know about. It's also difficult to teach the necessary mind-set for running systems securely. What I'll present here is a mixture of several years' personal experience of dealing with real security incidents and a brief coverage of the tools you'll need to keep any system vaguely secure. Unix Support also have a standard document, entitled "So you want to run a secure Unix system, do you?"[16], which is well worth reading (and much shorter than my outpourings here).

- Confidentiality
- Availablity
- Integrity

Slide 257: Components of security

What do we mean?

Before going any further, we should probably have some idea what we mean by security. It's all too easy to describe things as "secure" or "insecure" without having a clear idea what's actually meant by those terms.

Computer security is traditionally divided into three areas: confidentiality, integrity and availability. Confidentiality means ensuring that the wrong people don't get your information, availability means ensuring that you can get at your information, and integrity means ensuring that your information isn't tampered with. Here, we're mostly concerned with integrity, in its wider meaning of preventing your resources being used without your consent.

- Vulnerability
- Exploit
- Script Kiddie

Slide 258: Components of insecurity

How it goes wrong

Far more interesting to us than security, though, is insecurity, and knowing how it's exploited. Most security problems start with a vulnerability, some bug in a program that means that, if fed a certain set of inputs, it will do something it was never intended to do.

[16] `http://www-uxsup.csx.cam.ac.uk/security/unix-box.html`

Given a vulnerability, someone will write an exploit. This is a program which automatically sends the right inputs to a program to cause it to make its privileges available to an attacker.

Finally, to cause us any problems, the exploit needs to get used. Unhappily, the Internet is full of people who, while lacking the technical knowledge to exploit a vulnerability themselves, are quite capable of using an exploit script written by someone else. We call these people "Script Kiddies", and they are our primary enemies.

10.1 Defences

- `ucam.comp.security.announce` – for all of you
- `comp.security.announce` – vendor announcements
- Your vendor's security list
  - <`redhat-watch-list@redhat.com`>
  - <`debian-security-announce@lists.debian.org`>
- BugTraq (`chiark.mail.bugtraq`) – behind the scenes

Slide 259: Keeping informed

Keeping informed

Perhaps the most important aspect of keeping your system secure is to stay informed about the currently popular vulnerabilities, and current advice on fixing them. At a minimum, you need to read the `ucam.comp.security.announce` newsgroup. If you can't drive a newsreader, you can get at it through a Web interface[17], though the interface is rather crude. This newsgroup carries local announcements about security incidents and vendor alerts that the Computing Service thinks are particularly relevant to Cambridge. It sees about ten messages a month.

The best way to find out about security updates to the particular version of Linux that you're running is to subscribe to the mailing-list that announces such updates. Red Hat[18] and Debian[19] both have such lists.

Most sysadmins should also read `comp.security.announce`, which is a global newsgroup for security announcements. It consists mostly of vendor announcements of holes in their own products, and thus tends to hear about things late.

The enthusiastic sysadmin will want to subscribe to BugTraq, a moderated mailing list run from `securityfocus.com`. It's probably best read in the form of the newsgroup `chiark.mail.bugtraq`. It's good for getting news of vulnerabilities early, but the noise level is rather high.

- Turn off services you don't need
- That's it really

Slide 260: Limiting services

[17] `http://www.cam.ac.uk/cgi-bin-ucam/WWWnews-ucam?grp=ucam.comp.security.announce`
[18] `https://listman.redhat.com/mailman/listinfo/redhat-watch-list`
[19] `http://www.debian.org/MailingLists/subscribe#debian-security-announce`

Limiting services

As explained in the section on service configuration earlier, there are many services that can be run on a Linux system, only a few of which are actually necessary in any given case. Any service you're not running is one fewer place for security holes to occur, and one fewer package to upgrade every time a hole is discovered. It may seem obvious, but a good half of the break-ins we see come through services that weren't being used.

- Need to know what's insecure
- ... what you're running
- ... how to ensure they don't overlap

Slide 261: Keeping up-to-date

Keeping up-to-date

Probably the next most important aspect of security is actually paying attention to all those warnings. Usually they boil down to "If you're running the FooSoft XYZP server, you should upgrade to version 6.42 or later." Thus, you need to know what you're running, and you need to upgrade it if you're told to do so. The second of these is fairly easy, since Linux distributors (Red Hat included) usually release updated packages fairly quickly when there's a problem. The first was described to some extent in the section on service configuration, but there's another way.

- Check what you're running
- Results available to institution COs
- Details on the Web

Slide 262: Friendly probing

Friendly probing

One of our activities when we're not giving courses is running a set of machines whose purpose is to find out what services every networked machine in the University is running. The output of this process is currently available to anyone who's vaguely responsible for computing within a subdomain of `cam.ac.uk`. We have some Web pages which describe this in more detail (and provide the results)[20].

We're still working on making the output of this more generally available.

1. Your user name is `course` and your password is `fainbore`.
2. Find the probing results for your machine.

   `http://www-uxsup.csx.cam.ac.uk/security/probing/domains/titan1.pwf/`

3. See if you can account for everything they contain.

Slide 263: Exercise: Looking at probing output

[20] `http://www-uxsup.csx.cam.ac.uk/security/probing/`

10.2 Access controls

- Not all services need be globally accessible
- Access controls are not infallible
- Real security still necessary

Slide 264: Access controls

Most detected break-ins in the University come through software bugs, usually where a server program takes insufficient care over auditing its input. If you need a service to be visible to the world, there's not much you can do about these aside from keeping up with the fixes. If a service doesn't need to be accessible to the world, there are a variety of ways of ensuring that it isn't. Note that none of the techniques described here should be used as a primary form of security. IP source addresses can be faked, so you need some real security as well.

- `libwrap` – linked with `xinetd` and with non-`xinetd` services
- `tcpd` – no longer used
- Common behaviour
- Logging
- Access control

Slide 265: `tcp_wrappers`

`tcp_wrappers`

Most of the network servers shipped with Red Hat Linux (notably including `xinetd`) are linked against a library called `libwrap`, which is part of the `tcp_wrappers` package.

The first service provided by `libwrap` is to optionally log every connection it handles. Not all of `libwrap`'s clients use this service, and `xinetd` notably doesn't. The more useful service `libwrap` provide is a crude form of access control. This is controlled through two files, `/etc/hosts.allow` and `/etc/hosts.deny`, whose format is described in `hosts_access`(5) and `hosts_options`(5). Note that in most builds of `tcp_wrappers`, the latter takes precedence.

- *daemons* : *clients* [: *options* ...]
- `ALL : .impstud.cam.ac.uk :  rfc931 :  allow`
- `sshd :  ALL : rfc931 :  allow`
- `ALL : ALL : deny`

Slide 266: `hosts.allow` and `hosts.deny`

hosts.allow and hosts.deny

In general, each line in these files contains three colon-separated fields. The first is the name of the service that the line refers to, specifically the value of the zeroth argument to the server process (usually its filename). "ALL" can be specified to match all services and several services (separated by spaces) can be supplied.

The second field is a specification of a set of potential clients. Again, these are separated by spaces and can include host names, IP addresses, partial names or addresses, netgroups, IP addresses with netmasks and other more esoteric forms. The simplest form is a single hostname (like pcttr101.titan1.pwf.cam.ac.uk) or IP address (128.232.253.1). It's also possible to specify ranges of machines by giving domain names starting with a dot (.titan1.pwf.cam.ac.uk) or IP addresses ending with one (128.232.253.). IP address ranges can also be specified as an address mask pair (128.232.253.0/255.255.255.0).

The final field, which is optional, contains an action to apply if this line matches. The default is "allow" for lines in hosts.allow and "deny" for lines in hosts.deny. This, and the fact that hosts.allow is read first, is the only difference between the files.

The three lines in slide 266 are a possible set-up. The first line states that any host with an address corresponding to a name in impstud.cam.ac.uk can do anything at all to the machine, but that they will have RFC-931 identd lookups applied to them. The second line, allows anyone in the world to connect to the SSH server on the machine, again with an ident lookup. The final line denies any other access from anywhere, which is probably reasonable on a small machine.

1. Experiment with hosts.allow and hosts.deny
2. When you've finished, remove both files.

Slide 267: Exercise: hosts.allow and hosts.deny

- NFS has host lists in /etc/exports
- Web servers do their own thing
- lpd uses /etc/hosts.lpd (also /etc/hosts.equiv)
- xdm uses /etc/X11/xdm/Xaccess
- ypserv uses /etc/securenets (or libwrap)
- Samba uses /etc/smb.conf
- xinetd can do access control itself too
- ...and more

Slide 268: Service-specific controls

Service-specific controls

Many network services have their own particular way of configuring in access controls. Usually this is because they predate tcp_wrappers, or just because their authors thought they could do better. In many cases, this is the only kind of security they offer. Yes, this is a bad idea.

The NFS server stores its configuration in /etc/exports. This lists which directories are exported and to whom, and will be discussed in detail later. It should be noted that NFS also depends on the portmapper, which can have an entry in `hosts.allow`.

Web servers have their own access controls, and can usually be configured to enforce different controls for different sections of the document space.

The line printer daemon, `lpd`, keeps an access list in `/etc/hosts.lpd`. It also pays attention to `/etc/hosts.equiv`. See `lpd`(8).

The X11 display manager, `xdm` uses an access file (usually `/etc/X11/xdm/Xaccess`) to control access. The precise syntax of this file, which allows for fixed or wildcarded host names, but not IP addresses, is described in `xdm`(1).

The NIS server, `ypserv`, traditionally keeps an access list in `/var/yp/securenets`, though it can be compiled to use `tcp_wrappers` instead. See `ypserv`(8).

The SMB server, Samba, uses `libwrap` but keeps the access list in its own configuration file.

`xinetd`, which we've seen using `libwrap` already, can also do some crude IP-address-based access control internally using the `only_from` and `no_access` attributes.

There are more. Many more. Look at the documentation for the service you want to restrict access to and see what it can do.

As of 2002-11-18, the CUDN blocks:

- incoming:
  - SMTP, IMAP, IMAPS, POP and POPS (mail) to non-mail servers
  - DNS to non-DNS servers
  - FTP non non-FTP servers
  - TFTP, SNMP (network management), `syslog`, `lpr` and `rexec` to the entire CUDN
  - Sun RPC portmapper, NetBIOS and Finger to almost all machines
- outgoing:
  - SMTP except from mail servers

Slide 269: CUDN access controls

CUDN access controls

In addition to access controls you implement yourself, it's also possible for the CUDN to block certain types of network traffic. This doesn't happen very much, since there's very little that can be safely blocked, but SMTP, POP, IMAP, SNMP, finger and a few more obscure protocols are blocked at the JANET router except for a few destinations. This shouldn't cause problems in most cases, and where it does, holes can be arranged. It should be noted that these restrictions aren't guaranteed to be perfect – as ever, they're no substitute for real security. The current list of blocked ports is on the Web[21].

[21] `http://www.cam.ac.uk/localusersonly/cs/portblock.html`

10.3 Password security

- /etc/passwd contains encypted passwords
- encryption uses a one-way fuction ("hash")

  DES

  MD5
- reversing the hash is theoretically difficult (months or years)

Slide 270: Password encryption

Why password files can be readable

Something that occasionally confuses people about traditional Unix systems is that it's possible for any user to read `/etc/passwd`, the system password file. This is possible because the passwords stored in that file aren't held in plain text. Instead, what the password file contains is a cryptographic hash of the password. It's very easy to generate this hash from the password, but difficult to generate a password from the hash. This is a generally good design, though it does make many network authentication protocols hard to implement on Unix (because they tend to rely on the server knowing the user's plaintext password).

Unix encrypted passwords (as the hashes are known) usually use a variant of DES, the United States government's Data Encryption Standard. While this cipher is showing its age, it would still take a typical modern PC many months to "crack" (find a plaintext for) a single encrypted password. The design of the hash (with a two-byte "salt" added to the password before encrypting) also makes cracking many passwords in parallel a rather painful process. Modern dedicated hardware (like the Electronic Frontier Foundation's DES Cracker[22]) can break DES keys in a matter of hours, but this technology is still out of reach of your average script kiddie.

- Users choose bad passwords
- Dictionary attacks
- Obvious modifications
- Can usually get some passwords in minutes

Slide 271: User folly

Why password files shouldn't be readable

I've explained why it's theoretically fairly safe for encrypted passwords to be world-readable. Unfortunately, in the real world, it isn't. As is well known, people are a problem. They tend to choose bad passwords, and thus a good dictionary-based password guesser on a decent machine can expect to guess one password from a list of a few hundred in hours. This employs the standard script kiddie approach of attacking a large number of entities (passwords or computers) in parallel and using whichever one breaks first. For instance, I fed the encrypted password we got from the BIND attack above to Crack (the canonical Unix password cracker), and it found the clear-text password (`lamer1`) inside ten minutes.

Of course, if your users have really easy passwords, an attacker could just throw likely words at your password prompt until they got lucky. This is orders of magnitude slower than off-line cracking, though, and rather visible

[22] `http://www.eff.org/pub/Privacy/Crypto_misc/DESCracker/`

to the alert administrator.

Thus, it's a good idea to keep your encrypted passwords hidden. This is what most modern systems do, by having a shadow password file. The name and format of this file vary between Unices, but under Linux, it's called `/etc/shadow`, and has the format described in `shadow`(5). Of course, if an attacker ever does get to see your encrypted passwords, then you should assume they've all been cracked (some of them will be soon enough) and change every one.

Incidentally, the Computing Service have a leaflet[23] on how to pick a good password. It's worth distributing to your users.

10.4 Network sniffing

- Sniffing: listing for other people's traffic
- Broadcast networks (e.g. Ethernet)
- Old-Internet assumption this doesn't happen

Slide 272: Network sniffing

Network sniffing

Sniffing is the process of using a computer attached to a network to observe (and record interesting bits of) traffic on the network. On a broadcast network like an unswitched Ethernet, a machine can see all the network traffic between machine on that network. Sniffers can be very useful for network debugging, but many older network protocols base their security on the assumption that no-one would do such a thing without good cause. Thus, TELNET, FTP, HTTP, POP, IMAP, SNMP and others all default to sending passwords in clear-text over the network, and any network sniffer in the vicinity can record the passwords for future use. Various other protocols, such as NFS and NIS, make similar assumptions and have similar holes.

Several options:

- Secure the network (doomed)
- Challenge-response authentication
- Encrypted channels

Slide 273: Preventing sniffing

Ways to prevent sniffing

There are many ways to deal with this problem. Conceptually, the simplest is to ensure that your network is secure. This is impossible, but it's possible to come close with switched networks or secure repeaters. Unfortunately, this only works if you control all of the networks your connections travel over, which is rarely, if ever, the case. Even then, it's often easy to trick systems into sending traffic to places it's not meant to go.

A far better approach is to use an authentication mechanism that doesn't rely on sending passwords across the network, instead using cryptographic means for the client to prove to the server that it knows the password. Most

[23] `http://www.cam.ac.uk/cs/docs/infosheets/is6/`

secure authentication mechanisms in the non-Unix world use this kind of trick, not least because the form of cryptography it uses has never been prohibited from export from the United States. As I explained in section 10.3, these protocols are fundamentally incompatible with the Unix tradition of storing one-way hashes of passwords on the server. This is usually circumvented by having a separate password file (containing unencrypted passwords) for protocols which need them. A subclass of this is the S/Key one-time password system (also called "OPIE" under Linux), which allows such a system to be implemented without the need for specialised clients.

A final possibility, and probably the best in the real world, is to use proper encryption. This generally means either SSH (Secure Shell: an encrypted replacement for `rsh` and TELNET) or SSL (Secure Sockets Layer: an encryption layer that's independent of the overlying protocol). Both of these encrypt all traffic on a connection, and hence allow normal password authentication to be used, as the password can be sent encrypted and decrypted by the server.

10.5 SSH – the Secure Shell

- Encrypted remote login
- Other encrypted channels
- Real security with surprisingly little pain
- OpenSSH shipped with Red Hat Linux
- Available on CD from the Computing Service
- Two main parts, `ssh` and `sshd`
- Two versions: SSH1 and SSH2

Slide 274: SSH – the Secure Shell

SSH is an encrypted remote login protocol, and provides a means of making a really secure connection between two machines. This shouldn't be difficult, but the U. S. government's attitude towards the export of cryptography has held up this field for a while. Red Hat Linux comes with SSH client and server packages, called `openssh-clients` and `openssh-server`, along with an `openssh` package that both of these require. Many other operating systems don't come with SSH facilities, so the Computing Service has produced a CD-ROM[24] which can be obtained free of charge from Sales containing a selection of useful SSH related things. We like SSH.

The Unix SSH software has two main parts: the client (`ssh`) and the server (`sshd`). The server usually runs persistently, because that allows it to have some of the cryptographic leg-work done before anyone connects, which is worthwhile on a slow system. In the simple case, there isn't much configuration to be done of the daemon. When started for the first time, it automatically creates a set of host keys (which allows the server to authenticate itself to clients).

One thing that makes SSH more complex than it needs to be is that the protocol comes in two versions, SSH1 and SSH2. As the names suggest, SSH1 is the older protocol and is slowly dying out. OpenSSH supports both versions, but not in a terribly transparent fashion. Notably, because the two versions use different public-key authentication

[24] `http://www-uxsup.csx.cam.ac.uk/CD/`

algorithms (and SSH2 supports two different ones as well), you end up with a profusion of different keys, not all of which will work with all systems. This should improve as SSH1 dies out.

`ssh [-afgknqstvxACNPTX1246] [-c` *cipher_spec*`] [-e` *escape_char*`] [-i` *identity_file*`] [-l` *login_name*`] [-m` *mac_spec*`] [-o` *option*`] [-p` *port*`] [-L` *port*`:`*host*`:`*hostport*`] [-R` *port*`:`*host*`:`*hostport*`] [`*login-name*`@]`*host* [*command*]

- `ssh zombie` – use local username
- `ssh root@zombie`
- `ssh -v root@zombie` – verbose
- Warns if it doesn't recognize the host key
- Warns very noisily if the host key changes

Slide 275: SSH client usage

Client usage

The `ssh` client program is also fairly simple to use. In the simple case, you simply run it with a host name as an argument, or with a user ID and host name. For instance, to log onto `zombie` as `root`, I could do:

```
$ ssh root@zombie
wraith:~$ ssh root@zombie
Host key not found from the list of known hosts.
Are you sure you want to continue connecting (yes/no)?
```

The warning is normal for your first connection to a new machine – it indicates that, since your client has no prior knowledge of the server, it can't be sure it's the one you think it is. It's usually safe to say "`yes`" here. This will cause `ssh` to put a copy of the host's public key in `.ssh/known_hosts` in your home directory, so it'll recognise this host next time you use it.

Another warning can occur if the host you're connecting to appears to have changed its host key. This looks like:

```
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@       WARNING: HOST IDENTIFICATION HAS CHANGED!         @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
IT IS POSSIBLE THAT SOMEONE IS DOING SOMETHING NASTY!
Someone could be eavesdropping on you right now (man-in-the-middle attack)!
It is also possible that the host key has just been changed.
Please contact your system administrator.
```

As the message says, this can mean something bad's happening – SSH's check on whether the remote host is what you think it is has failed. If you know that the host key should have changed (for instance, if the machine's operating system's been re-installed), you should find the line corresponding to this host in your `known_hosts` file and delete it. Note that this file has rather long lines – make sure your text editor doesn't wrap them. If you aren't certain that the host key has changed, you should contact the people responsible for the machine and ask.

If you're playing with `ssh`, or it's misbehaving and you want to know why, it's well worth running it with the `-v` flag. This causes it to output a verbose, and fairly intelligible, set of messages explaining what's going on during a connection.

- `ssh root@zombie ps -aux` – run a process elsewhere
- X11 forwarding (`-X`)
- Port forwarding (`-L` and `-R`) – read documentation

Slide 276: Advanced(ish) SSH usage

Advanced client usage

In addition to carrying simple text logins, SSH can be used for more exotic purposes. The simplest is as a replacement for `rsh`, for executing commands on a remote system. Thus (for instance):

```
$ ssh root@zombie ps -aux
root@zombie's password:
USER       PID %CPU %MEM   VSZ  RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.7  1104  460 ?        S    Dec08   0:03 init [3]
root         2  0.0  0.0     0    0 ?        SW   Dec08   0:00 [kflushd]
root         3  0.0  0.0     0    0 ?        SW   Dec08   0:04 [kupdate]
root         4  0.0  0.0     0    0 ?        SW   Dec08   0:00 [kpiod]
root         5  0.0  0.0     0    0 ?        SW   Dec08   0:24 [kswapd]


...
```

There are several programs which traditionally use `rsh` internally, notably `rdist`, `rsync` and `cvs`. Such programs can generally be coerced into using SSH instead, and this automatically increases their security.

In addition to the usual terminal stream, SSH can forward X11 connections back to the client machine if you specify the `-X` flag. This means that you can use SSH to transparently run an X11 program on a remote machine. SSH actually makes this process a lot easier than it used to be, as it automatically handles passing X authentication tokens around.

SSH can also be persuaded to forward arbitrary network connections. This is a rather complex area, and I'd recommend reading the documentation before trying it.

- Configuration is in `/etc/ssh/sshd_config`
- Keys are in `/etc/ssh/ssh_host*_key` and `/etc/ssh/ssh_host*_key.pub`
- Private key must be kept secret
- No good way to proclaim a key dead

Slide 277: SSH server operation

Server operation

The SSH server, `sshd` can mostly just be left to get on with its job. Its configuration file is `/etc/ssh/sshd_config`.

Its contents are described in `sshd`(8) and I shan't go into them further here.

The main thing that does need a little care in managing the SSH server is the host keys. An OpenSSH server will have three host keys, one RSA key for SSH1, one RSA key for SSH2 and one DSA key for SSH2. Each host key is actually a pair of keys, one public and one private. The public keys are stored in `/etc/ssh/ssh_host_key.pub`, `/etc/ssh/ssh_host_rsa_key.pub`, and `/etc/ssh/ssh_host_dsa_key.pub`, while the private keys are stored in `/etc/ssh/ssh_host_key`, `/etc/ssh/ssh_host_rsa_key`, and `/etc/ssh/ssh_host_dsa_key`. The way these works is that given the public key, a client can verify whether the server possesses the private key, and hence can verify that it's the right server. Thus, SSH's anti-spoofing protection relies on the private key's never being revealed to a third party. If the host key is ever accidentally revealed (for instance, if the server's broken into) it's necessary to generate a new host key for the server and to tell everyone to stop using the old one. Sadly, SSH doesn't provide any facilities for doing this automatically.

Authenticating yourself with SSH

As I explained above, SSH uses public-key cryptography to allow the server to prove its identity to the client. In the usual case, the client doesn't prove its identity to the server – instead the user proves their identity by providing a password. It is possible, though, to configure SSH to authenticate users without using passwords. This involves running `ssh-keygen` on the client side, copying the resulting identity file (`.ssh/identity.pub`, `.ssh/id_rsa.pub`, or `.ssh/id_dsa.pub`, depending on the type of key) to the server and adding it to the end of `.ssh/authorized_keys` or `.ssh/authorized_keys2` there. There's also a halfway-house mechanism which works like `rsh`'s `.rhosts` mechanism which I shan't explain here.

10.6 X11 security

- Network transparency: both a blessing and a curse
- `Xsecurity`(1)
- Three levels of security

Slide 278: X11 security

This is a brief aside into the security facilities provided by X11. X's network transparency is one of its nicest features, but it also means that your windowing system can be a security hole. This is because X11 provides very little security between the clients of a display. Any program that can connect to your display can observe any keystrokes or mouse movements on it, can generate fake keystrokes and mouse movements, and can observe and modify any part of the screen display. Allowing untrusted programs access to your X display is thus clearly a rather bad idea. Allowing trusted ones to do so, on the other hand, is necessary for X to be of any use at all.

There are effectively three ways of allowing yourself to run X applications from multiple machine on one display, and I'll go through them in order of increasing security. The `Xsecurity`(1) manual page has information on the first two.

- Don't

Slide 279: Host-based security

Host-based security with `xhost`

It's possible for an X server to have a list of hosts it trusts absolutely to display windows on it. As I described above, relying entirely on host-based security is a very bad idea, and this is no exception. All you need to know is that the `xhost` command, especially with any plus signs in its argument, is a very bad thing. On a correctly set up X server, running `xhost` with no arguments (which displays the current state) should produce one line of output:

```
access control enabled, only authorized clients can connect
```

Anything else is cause for concern.

- MIT-MAGIC-COOKIE-1
- Basically password authentication for programs
- Cookies managed by `xauth`
- Can be propagated between machines
- Usual sniffing problem

Slide 280: Authentication tokens

Authentication tokens and `xauth`

There is a rather better way to manage X security. X has a built-in authentication mechanism with a variety of authentication schemes, which are controlled by the `xauth` command. Unfortunately[25], U. S. export regulations mean that most of the properly secure ones never became popular, so the world's stuck with what's known as MIT-MAGIC-COOKIE-1. This is effectively a password authentication system where the password is stored in the `.Xauthority` file in your home directory. Any program which can get at that password can get at your X display. Obviously, this is as prone to sniffing as other forms of unencrypted password authentication.

The cookies in the `.Xauthority` file can be manipulated with the `xauth` program. With a bit of cunning, it's possible to copy them between systems, so that accounts on different machines which don't share your home directory can still have access to your display.

- Usually automatic (may need `-X`)
- Secure in most respects
- Permission to use the display revoked on disconnection
- Usually the Right Thing

Slide 281: X forwarding over SSH

X forwarding with SSH

By far the best way to handle using X11 displays is to use SSH. This provides more-or-less automatic forwarding of X connections between machines, and handles security in a sensible way. When you set up a connection with X forwarding, the SSH server creates a fake X server on the remote machine and gives it its own magic cookie, then forwards any connections to this server back to the original, making the necessary changes to the cookie as it goes past. This means that the X connection is encrypted (and hence secure against eavesdropping and tampering) and

[25] I'm using that word rather a lot, aren't I?

the cookie twiddling ensures that when the SSH connection is closed, the remote machine loses the right to use your display.

10.7 Threat models – know your enemy

It's generally good practice in security, and in computer security, to have some idea of what kind of attacker you're trying to defend against. Without such a model, it's very easy to get confused about whether your defences actually do anything useful. Here, I'll present a brief description of the behaviour what seems to be the most common kind of attacker these days, the *script kiddie*.

Three phases:

- *ab initio* attacks
- Gaining accounts by password sniffing
- Upgrading non-`root` accounts to `root`

Slide 282: Script kiddies

The life cycle of the script kiddie

> **script kiddies** pl.n.
>
> The lowest form of cracker; script kiddies do mischief with scripts and programs written by others, often without understanding the exploit.
>
> —Jargon File 4.1.4

The script kiddie, as their name implies, is someone armed with an automated, or semi-automated tools for attacking networked machines. Their automation means that while the lowest form of life to many, they're also the most likely people to break into your systems, and thus the ones you most need to stop.

There seem, in general, to be three phases to script kiddie activity (though I suspect these aren't actually sequential). The first is the *ab initio* break-in, where the kiddie picks a netblock (say `131.111.0.0/16`) and tries an exploit on every machine in the block, collecting a list of the successes. Having gained `root`-level access to a machine (or forty), the kiddie will often run network sniffers on them, collecting plain-text passwords for other machines on the same Ethernet, thus gaining access (albeit not at root level) to them. The final phase (in this list: the cycle continues) is to take one of those user-level accounts and exploit one of the many user-to-`root` attacks against it to gain another `root`-level account.

23rd November 1999

- 08:30 – download DNS zone
- 08:40 – probe all machines
- 09:10 – break into vulnerable machines
- 10:00 – go home

Slide 283: Chronology of an attack

Anatomy of an attack

Here, I'll describe a particular attack on the University's systems. This is a true story, and one of about ten I could have produced from my first six months in Unix Support. It just happened to be the most recent when I wrote this section.

The attack started at about 08:30 on 23rd November 1999. Our traffic logs show that `146.83.22.169` downloaded the entire `cam.ac.uk` DNS zone from the University's primary name servers, and also the zones of all `cam.ac.uk`'s delegated subdomains. It then started at the top of the list, and sent a 55-byte DNS packet to each host. Most of them weren't running a DNS server and promptly sent back an ICMP Port Unreachable message. Many others weren't turned on, and thus the packets got lost (and re-sent by the attacker). A few were listening, replied and had their IP addresses noted down by the attacker.

Starting at about 09:10, the attacker, using a different machine (`tassksv.icrr.u-tokyo.ac.jp`, `133.11.133.138`) sent a selection of interesting DNS packets, designed to tickle a recently-published bug in BIND, to each of the machines he'd collected. As far as we know, the exploit caused BIND (which traditionally runs as `root`) to add an extra root user to `/etc/passwd`, with a password known to the attacker:

```
foom:VZOc9N00KKQtg:0:0::/:/bin/bash
```

The attacker then logged in to each machine by telnet, downloaded an assortment of software by FTP and installed it. The software included an SSH server (so the attacker could log in securely), replacement versions of `ps`, `ls` and a few other tools, to hide his activities, and a network packet sniffer, to collect new passwords. By 10:00, the attacker had finished their work, having root access to ten machines and a network sniffer running on most of them.

10.8 Dealing with attacks

- The attacker will try to stop you
- We will often notice before you
- Mainly a matter of intuition
- Automated tools – `tripwire` etc
- If in doubt, ask!

Slide 284: Detecting a break-in

Detecting a break-in

Noticing that one of your machines has been compromised is difficult. Attackers will generally do their best to ensure that the machine looks normal. Luckily for us, their best is often not good enough. In our experience, the most common way for an administrator to find out their machine's been compromised is for us to tell them. This is simply because after a break-in, we usually search for other compromised machines and notify their owners.

Someone, though, has to notice the break-in in the first place. This is usually because something about the system has changed, so it's necessary to have a good feel for how the system usually behaves. You should get in the habit of looking through the system log files, so that you know what they usually look like. The usual sign of a break-in in the logs is a gap where the attacker's removed a chunk of them, but they frequently leave something suspicious.

There are also tools that can be used to detect break-ins. `tripwire` is the most common of these. Of course, the tools themselves can be tampered with, which means that they need to be obscure enough that the attacker hasn't heard of them. Security through obscurity is a bad idea, but in some cases it's all you've got left.

If you're not certain whether a set of symptoms indicate a real break-in, it's far better to get in touch with us than not to. Unless you're crying "wolf" every month, we won't mind much.

- Don't panic
- Don't halt or reboot
- Disconnect from the network
- e-mail <`cert@cam.ac.uk`> and <`unix-support@ucs.cam.ac.uk`>
- expect to have to re-install the system
- expect to have to change everyone's passwords

Slide 285: So, you've been hacked?

First things first

Discovering that a machine's been broken into is never a relaxing experience, but it's one you're likely to have eventually. If you discover (or even suspect) that a machine you're responsible for has been broken into, the most important thing to do is to prevent it causing more problems. Usually, the best way to do this is simply to disconnect it from the local network. This prevents its being used to attack other machines, while disturbing any evidence on the machine as little as possible.

The next thing to do is to tell Cambridge CERT <`cert@cam.ac.uk`>. Assuming it's a Unix machine that's involved, it's also a good idea to send a copy of your message to us <`unix-support@ucs.cam.ac.uk`>, since we like to keep an eye on what's going on, and can help if you're not sure what to do.

What to do beyond that depends very much on the nature of the attack and the importance of the victim machine. In most cases, we can advise, but for those occasions when this isn't possible, and for those who'd like to know how much grief a break-in will cause, here's an outline.

- Very little can be trusted
- `tripwire` and `rpm --verify` may help
- If in doubt, assume the worst

Slide 286: Determining the extent of the damage

Determine the extent of the damage

The important things to do are to determine the extent of the damage caused, and to repair it. Tracking the offender down and killing their family are secondary, and in any case generally a pain. If you can, it's a good idea to take a look around the system to determine what the attacker has done, and more importantly, what they're still doing. This can be an awkward process, as useful system binaries (and in extreme cases, the operating system kernel) are likely to have been changed to hide any interesting behaviour. You can't rely on `rpm --verify` to tell you about all changed binaries, either, since either `rpm` or its database may have been tampered with. Copying in programs from a known-safe machine works much of the time, as does using programs the attacker didn't think to change (looking through `/proc` by hand is good for this).

- Best to re-build from scratch or backups
- Your backups had better be recent.

Slide 287: Re-installation

Re-install

Having found out all you can (or all you can be bothered to), you should disconnect the machine from the network. It's probably a good idea to do this before starting to investigate, but it may cause you to lose information as network connections (revealing the address of other affected machines) die. At this point, taking a complete back-up of the machine is a good idea. For one thing, this is often useful for later analysis. More importantly, unless you're absolutely certain that you know the full extent of the damage to the machine, you're going to have to re-install the operating system (or at least restore the whole thing from the last known-good backup). This is the point where having a recent backup is really important.

- Assume attacker knows everything
- Yes, it is a pain
- Change host keys as well

Slide 288: Cleaning up

Change all passwords

Finally, it's usually a good idea to force anyone with accounts on the broken machine to change their passwords. This includes anyone with an account in a NIS domain accessible to the victim. Any account that was logged into from the victim should also have its password changed, as should any that was logged into over the network the victim is on. Changing passwords often turns out to be more effort than all of the rest of dealing with an incident, as anyone who attended the last CUS password-change will know. It'll also be necessary to change the machine's SSH host key and any other details that are meant only to be known by it.

11 The Network File System

11.1 Introduction

- Concept: Transparent access to files on remote system.
  - Server exports directory hierarchy.
  - Client mounts directory as if local disc partition.
- Properties:
  - Client inherits file ownership and permissions from server
  - Only approximates local disc: weak cache consistency.
  - Architecture neutral
- NB: NFS support in Linux still under development

Slide 289: Introduction to NFS

Introduction to NFS

The Network File System (NFS) is a network protocol which allows one machine to access files stored on a second system. An NFS server system exports certain directory hierarchies from its local disc partitions. An NFS client system can then mount directories from that hierarchy as if they were filesystems stored on local disc. This requires a certain amount of cooperation between the two systems which are involved. An NFS client system inherits file ownerships and permissions from its server. If a particular user needs to access files on an NFS server, the UID and GID information for that user needs to be consistent on the two machines.

NFS is intended to be more or less transparent (other than the fact that access to a disc on a remote system will clearly be slower than access to an equivalent disc which is attached directly to a machine). Unfortunately, NFS provides only a rough approximation to a standard Unix filesystem, especially where concurrent access to files and directories by different NFS client systems is concerned. NFS provides only weak cache consistency, which means that changes on one client system may not be reflected on other client systems for some time (typically a number of seconds). This can clearly cause a great deal of confusion and even corrupt data if applications have not been explicitly designed to cope with and work around artifacts from NFS caching.

NFS is designed to work across many different types of platform. It is possible to export a directory from one type of Unix system and mount it on a second machine which is running an entirely different kind of Unix. It is even possible to NFS mount filesystems which live on VMS systems or IBM mainframes although subtle translation problems can occur, especially where unusual filenames and symbolic links are involved.

- NFS clients and servers all need to run:
  - `portmap` (the RPC portmapper)
  - `lockd` and `statd`
- NFS servers (only) need to run:
  - `nfsd`
  - `mountd`
  - `rquotad`
- Not just yet! Need to set up one file first.

Slide 290: NFS daemons

NFS daemons

NFS client and server systems need to be running various system daemons before NFS will work reliably. NFS is an RPC based service, consequently the RPC portmapper must be running. They should also run the `lockd` and `statd` daemons for file locking to work reliably. File locking is used to guarantee exclusive access to a file.

`lockd` running on a client system records lock requests and then forwards them onto the `lockd` daemon on the appropriate server system. A program which is attempting to lock a file will wait until the server replies to indicate that the lock has been taken.

When an NFS client or server system reboots, `lockd` daemons running on different systems can end up in inconsistent states. The `statd` program is used to record the state of various systems. This daemon causes `lockd` on a client system to reassert its locks when an NFS server reboots. Conversely it causes `lockd` on an NFS server system to cancel locks when an NFS client reboots. `statd` stores state information in the directories `/var/lib/nfs/sm` and `/var/lib/nfs/sm.bak`. You may need to clear out old entries in these directories if an NFS client system is shut down permanently or renamed.

File locking over NFS has a very poor reputation, mostly because of buggy or careless implementations. It was even possible to hang clusters of systems in such a way that the entire cluster would have to be rebooted. Modern implementations tend to be far more reliable in this regard.

NFS servers also need to run the `nfsd`, `rquotad` and `mountd` daemons. `nfsd` is the main NFS daemon: it is the daemon which actually provides file data to an NFS client system. `mountd` processes requests from client machines to mount and unmount directories. `rquotad` is used to report disc quota use if disc quotas are enabled on the server.

11.2 NFS experiments on a single system

Warning: `/etc/hosts.deny` likely to get in the way!

1. Create some test directories as root:

   # **mkdir /home/test /mnt/test**

2. Add following line to the file `/etc/exports`: (NB: underscores)

   ```
   /home/test    localhost(rw,no_root_squash)
   ```

3. Start up NFS daemons:

   # **/etc/init.d/nfslock start**

   # **/etc/init.d/nfs start**

4. Set up NFS mount:

   # **mount localhost:/home/test /mnt/test**

Slide 291: Exercise: Simple NFS experiment: Part I

1. `/mnt/test` is now an alias for `/home/test`
2. Try creating files under `/home/test`

   Will appear under `/mnt/test`
3. Try creating files under `/mnt/test`

   Will appear under `/home/test`
4. When you have finished, remove the NFS mount:

   # **cd /**

   # **umount /mnt/test**

Slide 292: Exercise: Simple NFS experiment: Part II

Experimenting with NFS on a single system

Let's try some simple experiments with NFS. Create test directory named `/home/test` and `/mnt/test` on your system:

mkdir /home/test /mnt/test

Then add the following line to the file `/etc/exports`:

```
/home/test  localhost(rw,no_root_squash)
```

This allows the machine `localhost` (our own system!) to NFS mount the directory `/home/test` read-write. The `no_root_squash` part means that the root account on the NFS client machine has unrestricted root access to the

NFS server. This avoids of lot of nasty gotchas with access permissions while we are getting to grips with NFS on a single system. However you clearly don't want to do this when multiple systems are involved unless the server trusts the root account on the client machines.

Once the `/etc/exports` file has been set up, we can start up the various NFS daemons. NFS clients and servers need to run `/etc/init.d/nfslock` in order to start up `lockd` and `statd`. NFS servers also need to run `/etc/init.d/nfs` in order to start up `nfsd`, `mountd` and `quotad`. If this works correctly, you should get something like the following:

```
# /etc/init.d/nfslock start
Starting NFS statd:                          [  OK  ]
# /etc/init.d/nfs start
Starting NFS services:                       [  OK  ]
Starting NFS quotas:                         [  OK  ]
Starting NFS daemon:                         [  OK  ]
Starting NFS mountd:                         [  OK  ]
```

If we now run the following command:

```
# mount localhost:/home/test /mnt/test
```

`/mnt/test` becomes an alias for `/home/test`. If we create a file under `/home/test`, this file will appear under `/mnt` and vice versa. The critical concept is that the two directories in question can reside on two entirely independent systems. When you have finished with the `/mnt` mount you should remove the mount using the `umount` command:

```
# cd
# umount /mnt/test
```

11.3 NFS server configuration

- Some examples (underscores and spacing significant!):

```
/home/test    pcttr101.titan1.pwf.cam.ac.uk(rw,root_squash)
/home/test    *.titan1.pwf.cam.ac.uk(rw,no_root_squash)

/home/test2   128.232.253.1(rw)
/home/test2   128.232.253.0/255.255.255.0(ro)

/pub          magenta.csi.cam.ac.uk(rw,squash_uids=0-499)
/pub          (ro,all_squash,anonuid=500,anongid=500)
```

- Restrictions: NFS mounts. (Normally) subdirectory exports.

Slide 293: Access control for NFS servers: `/etc/exports`

Access control for NFS servers: `/etc/exports`

An NFS server clearly needs to be able to restrict just which remote systems are allowed to access local files via NFS. This is controlled by a single file: `/etc/exports` whose format is defined in the exports(5) manual page. Each line in this file consists of two columns. The first column is the name of a local directory which is to be exported. The second column is a machine or set of machines that this directory should be exported to, followed by a list of options in parentheses. A single directory can appear in the export list several times: this allows you to specify different export options depending on the NFS client system.

The list of hosts can take any of the following forms:

- Hostname (e.g: `pcttr101.titan1.pwf.cam.ac.uk`).
- Host wildcard (e.g: `*.titan1.pwf.cam.ac.uk`).
- IP address (e.g: `128.232.253.1`)
- IP network/mask (e.g: `128.232.253.0/255.255.255.0`).

The most useful options include:

- `rw`: Permit read-write access from clients
- `ro`: Restrict clients to read-only access
- `no_root_squash`: `root` user on client systems has `root` access to server.
- `root_squash`: `root` user on clients maps to anonymous user (typically `nobody`).
- `all_squash`: maps all users to the anonymous user
- `anonuid`: defines the user ID (UID) for anonymous access. The default is the UID of user nobody if that user exists on the system, otherwise -2, which is user "`nobody`" on most Unix systems.
- `anongid`: defines the group ID (GID) for anonymous access. The default is the GID of group "`nobody`" if that group exists on the system, otherwise -2, which is group "`nobody`" on most Unix systems.

NB: Red Hat Linux uses some rather unusual names for export options. If you use other Unix platforms you will find that similar options exist but with different names. Its always wise to read the `exports`(5) manual page.

NFS servers can only export data on local disc: it is not possible for one system to NFS mount a directory from another system and then export the combined directory hierarchy to a third system. If you attempt to do this, the third system will see the directory structure on the middle system without the overlaid files from the original server.

Most NFS implementations do not allow you to export a directory to a particular client system and then export some subdirectory to the same client with different access permissions. Red Hat Linux does let you do this: the most specific match is used when the NFS client mounts a directory. This can occasionally be useful if you play strange games with the `anonuid` and `squash_uids` options to provide restricted access to private data within the directory structure. However, its probably best not to rely on this behaviour as it is non-standard and may disappear at some later point in time.

1. Add following to start of `/etc/exports` file:

```
/home/test *.titan1.pwf.cam.ac.uk(rw,no_root_squash)
```

2. Update active exports list using "`exportfs`":

```
# exportfs -rv
unexporting localhost.localdomain:/home/test
unexporting *.titan1.pwf.cam.ac.uk:/home/test
exporting localhost.localdomain:/home/test
exporting *.titan1.pwf.cam.ac.uk:/home/test
```

Slide 294: Exercise: Exporting a directory to other systems

11.4 NFS client configuration

- Can run `mount`(8) directly. Have seen this many times already e.g:

```
# mount nfs-uxsup.csx.cam.ac.uk:/linux/redhat /redhat
```

- Strictly speaking should be:

```
# mount -t nfs -o ro ...
```

- Can also add to `/etc/fstab` just like disc partition:

```
nfs-uxsup.csx.cam.ac.uk:/linux/redhat /redhat nfs defaults 0 0
```

Slide 295: NFS client side configuration: `mount`(8) and `/etc/fstab`

NFS client configuration: `mount` and `/etc/fstab`

A client machine can NFS mount directories from the command line using the `mount`(8) command:

```
mount -t nfs -o ro nfs-uxsup.csx.cam.ac.uk:/linux/redhat /redhat
```

The `-o` option provides a (comma separated) list of options for the mount. The `-t nfs` part is actually optional: the `mount` command defaults to NFS if `hostname:path` is given instead of the device name of a local disc partition.

The `/etc/fstab` file can list NFS mounts as well as local filesystems. This is a convenient shorthand if a particular mount point is always used for the same NFS mount. It also allows Linux to automatically mount directories when the system boots. Example:

```
nfs-uxsup.csx.cam.ac.uk:/linux/redhat  /redhat  nfs  defaults  0 0
```

- Common options to change:
  - `rsize=8192,wsize=8192`. (Default 1024)
  - `nosuid` and `nodev`
  - `noac`
- Some standard options (defaults normally appropriate):
  - `fg` and `bg`. (Default: `fg`)
  - `hard` and `soft`. (Default: `hard`)
  - `intr` and `nointr` (Default: `intr`)
- (Partial) example:

```
# mount -t nfs -o rsize=8192,wsize=8192 ...
```

Slide 296: Common mount options for NFS: see `nfs`(5)

Common mount options for NFS: see `nfs`(5)

Some common mount options for NFS, all described in the `nfs`(5) manual page:

`rsize` and `wsize` are the block sizes for read and write commands to the NFS server. The default is 1024 which is much too small. 8192 is the recommended block size.

`nosuid` and `nodev` disable set-user-id (suid) bits and device special files on NFS mounted partitions. Both these options are sensible security precautions, otherwise NFS client systems are trivial to compromise if their NFS servers are broken into.

The `noac` option disables the attribute cache on an NFS mount. This degrades performance considerably as every single NFS operation will force an attribute check against the server to check whether files have been changed since the last NFS operation. However this is the only guaranteed safe way to work if multiple client systems are accessing the same NFS server at the same time. The general recommendation is that if `/var/spool/mail` is shared by several systems, it should be mounted with the attribute cache disabled.

A directory can be NFS mounted in the foreground (`fg`) or in the background (`bg`). When a directory is mounted in the foreground, the NFS client will wait until that mount succeeds before it proceeds. If the NFS server in question is unavailable for any reason, the client can wait indefinitely. In contrast, background mounts are deferred if the NFS server is unavailable when the mount is first attempted. Critical system directories should be flagged as `fg` mounts in `/etc/fstab`, otherwise a reboot may have unpleasant side effects. Other directories would typically be flagged as `bg` directories so that an NFS client can reboot even if some of its home directory servers have been shut down or are still in the process of rebooting. If neither option is specified the default behaviour is to mount in the foreground.

A directory can be NFS mounted as a `hard` or `soft` mount. Operations on a `hard` mounted directory will retry indefinitely: the NFS client system will never give up retrying. In contrast operations to a `soft` mounted directory will eventually fail with unpredictable consequences. Hard mounts are strongly recommended for read-write filesystems and this is the default.

An NFS mount can be interruptible (`intr`) or non-interruptible (`nointr`). If NFS operations are timing out (which either means that the NFS server involved is very slow, or otherwise that it is dead), users can interrupt processes which are blocked trying to access files on an interruptible mount. They can do this either by using [Control]+[C] or by using the `kill` command to send signals to the process in question. `intr` is slightly more friendly from a user's perspective. However interrupting a program part way through a read or write operation may have unexpected side-effects. `nointr` is much safer for critical filesystems.

- To check whether a machine is an NFS server:

  # **rpcinfo -p pcttr101.titan1.pwf.cam.ac.uk**

- To find out which directories an NFS server is exporting:

  # **showmount -e pcttr101.titan1.pwf.cam.ac.uk**

- To list existing NFS mounts on a client system:

  $ **mount -t nfs**

  $ **df -t nfs**

- The file `/etc/mtab` also shows current active mounts.

Slide 297: Some useful diagnostic tools for NFS clients

Some useful diagnostic tools for NFS clients

`showmount` shows a list of directories that an NFS server exports. `mount` and `df` will show both NFS and local mounts unless you specifically ask for only NFS mounts to be listed. This information is also listed in the file `/etc/mtab`. Note that the `mount` command gives useful information about the options used on a mount.

```
# showmount -e localhost
Export list for localhost:
/home      localhost.localdomain

# mount -t nfs
localhost:/home/test on /mnt/test type nfs (rw,addr=127.0.0.1)

# df -t nfs
Filesystem           1k-blocks      Used Available Use% Mounted on
localhost:/home/test   3209944   1565352   1481536  51% /mnt/test
```

1. Check that server is running NFS services:

   $ **/usr/sbin/rpcinfo -p pcttr101.titan1.pwf.cam.ac.uk**

   No response? Check `/etc/hosts.deny` on NFS server!

2. List available mounts on server:

   $ **/usr/sbin/showmount -e pcttr101.titan1.pwf.cam.ac.uk**

3. Try mounting a remote filesystem. e.g:

   # **mount pcttr101.titan1.pwf.cam.ac.uk:/home/test /mnt/test**

4. Look at and then create files under `/mnt/test`

5. Don't forget to remove the mount when you are finished:

   # **cd /**

   # **umount /mnt/test**

Slide 298: Exercise: Mounting a directory from a remote system

11.5 The Automounter

- Basic concept:

  Directory mounts automatically when accessed.

- Two competing technologies:
  - AMD: User space automounter, relies on symlinks
    Established technology: reliable but slow.
  - Autofs: Kernel space automounter
    New: Some rough edges. Likely to dominate long term.

Slide 299: Automounters

Introducing the Automounter

An automounter is a piece of system software which mounts filesystems automatically when certain directories are accessed.

```
$ rpm -qi autofs
Name        : autofs              Relocations: (not relocateable)
Version     : 3.1.7               Vendor: Red Hat, Inc.
...
Summary     : A tool for automatically mounting and unmounting filesystems.
Description :
Autofs controls the operation of the automount daemons. The automount
daemons automatically mount filesystems when you use them and
unmount them after a period of inactivity. Filesystems can include
network filesystems, CD-ROMs, floppies, and other media.
```

Slide 300: Checking `autofs` is installed

- `/etc/auto.master` devolves control to subsidiary files:

```
#/misc  /etc/auto.misc  --timeout 60
```

- Subsidiary files define mount points and options. `/etc/auto.misc`:

```
cd          -fstype=iso9660,ro,nosuid,nodev   :/dev/cdrom
```

Slide 301: Automounter configuration

Automounter configuration

The central configuration file for autofs is `/etc/auto.master`. This file defines the directories which are to be placed under autofs control and starts with a single line . The default `/etc/auto.master` contains the following single entry:

```
/misc  /etc/auto.misc  --timeout 60
```

This specifies that the directory structure under `/misc` will be defined by the file `/etc/auto.misc` and that mounts under `/misc` will timeout and be removed after 60 seconds of inactivity. The `autofs` daemon must be restarted whenever the `/etc/auto.master` file is updated using the `restart` option to `/etc/init.d/autofs`. However you do not need to restart `autofs` if any of the subsidiary files change.

Each line of `/etc/auto.misc` consists of a directory name, followed by mount options for this directory and a device name or NFS directory specifier which will be used as the source for the mount:

```
kernel          -ro,soft,intr          ftp.kernel.org:/pub/linux
cd              -fstype=iso9660,ro     :/dev/cdrom
```

The first example mounts the directory `/pub/linux` directory from `ftp.kernel.org` onto `/misc/kernel` using NFS. The second example automatically mounts ISO 9660 format CDROMs on the local system to the directory `/misc/cd`. In both cases the middle column is a set of options that will be provided when the filesystem in question is mounted: see `mount`(8) for details.

1. Look at the file `/etc/auto.master`
2. Add following to end of `/etc/auto.misc`

   ```
   linux  -ro,soft,intr  nfs-uxsup.csx.cam.ac.uk:/linux
   ```

3. Automount `/misc/linux` should exist, but *only* when accessed:

   ```
   $ cd /misc
   $ ls
   $ ls linux
   debian kernel redhat slackware
   ```

Slide 302: Exercise: Using autofs