

debian

Debian Reference

Osamu Aoki

Copyright © 2013-2024 Osamu Aoki

This Debian Reference (version 2.152) (2026-10-10 14:56:25 UTC) is intended to provide a broad overview of the Debian system as a post-installation user's guide. It covers many aspects of system administration through shell-command examples for non-developers.

Contents

1 GNU/Linux tutorials **1**

1.1 Console basics . . . 1

1.1.1 The shell prompt . . . 1

1.1.2 The shell prompt under GUI . . . 2

1.1.3 The root account . . . 2

1.1.4 The root shell prompt . . . 3

1.1.5 GUI system administration tools . . . 3

1.1.6 Virtual consoles . . . 3

1.1.7 How to leave the command prompt . . . 3

1.1.8 How to shutdown the system . . . 4

1.1.9 Recovering a sane console . . . 4

1.1.10 Additional package suggestions for the newbie . . . 4

1.1.11 An extra user account . . . 5

1.1.12 sudo configuration . . . 5

1.1.13 Play time . . . 6

1.2 Unix-like filesystem . . . 6

1.2.1 Unix file basics . . . 6

1.2.2 Filesystem internals . . . 8

1.2.3 Filesystem permissions . . . 8

1.2.4 Control of permissions for newly created files: umask . . . 10

1.2.5 Permissions for groups of users (group) . . . 11

1.2.6 Timestamps . . . 12

1.2.7 Links . . . 13

1.2.8 Named pipes (FIFOs) . . . 14

1.2.9 Sockets . . . 15

1.2.10 Device files . . . 15

1.2.11 Special device files . . . 16

1.2.12 procfs and sysfs . . . 16

1.2.13 tmpfs . . . 17

1.3 Midnight Commander (MC) . . . 17

1.3.1 Customization of MC . . . 17
1.3.2 Starting MC . . . 18
1.3.3 File manager in MC . . . 18
1.3.4 Command-line tricks in MC . . . 18
1.3.5 The internal editor in MC . . . 19
1.3.6 The internal viewer in MC . . . 19
1.3.7 Auto-start features of MC . . . 19
1.3.8 Virtual filesystem of MC . . . 20
1.4 The basic Unix-like work environment . . . 20
1.4.1 The login shell . . . 20
1.4.2 Customizing bash . . . 21
1.4.3 Special key strokes . . . 21
1.4.4 Mouse operations . . . 21
1.4.5 The pager . . . 22
1.4.6 The text editor . . . 23
1.4.7 Setting a default text editor . . . 23
1.4.8 Using vim . . . 23
1.4.9 Recording the shell activities . . . 24
1.4.10 Basic Unix commands . . . 25
1.5 The simple shell command . . . 25
1.5.1 Command execution and environment variable . . . 27
1.5.2 The "$LANG" variable . . . 27
1.5.3 The "$PATH" variable . . . 28
1.5.4 The "$HOME" variable . . . 28
1.5.5 Command line options . . . 29
1.5.6 Shell glob . . . 29
1.5.7 Return value of the command . . . 30
1.5.8 Typical command sequences and shell redirection . . . 31
1.5.9 Command alias . . . 32
1.6 Unix-like text processing . . . 32
1.6.1 Unix text tools . . . 32
1.6.2 Regular expressions . . . 33
1.6.3 Replacement expressions . . . 34
1.6.4 Global substitution with regular expressions . . . 35
1.6.5 Extracting data from text file table . . . 36
1.6.6 Script snippets for piping commands . . . 37

2 Debian package management **39**

2.1 Debian package management prerequisites . . . 39
2.1.1 Debian package management system . . . 39
2.1.2 Package configuration . . . 39
2.1.3 Basic precautions . . . 40
2.1.4 Life with eternal upgrades . . . 41
2.1.5 Debian archive basics . . . 42
2.1.6 Debian is 100% free software . . . 45
2.1.7 Package dependencies . . . 47
2.1.8 The event flow of the package management . . . 48
2.1.9 First response to package management troubles . . . 49
2.1.10 How to pick Debian packages . . . 49
2.1.11 How to cope with conflicting requirements . . . 50
2.2 Basic package management operations . . . 50
2.2.1 `apt` vs. `apt-get` / `apt-cache` vs. `aptitude` . . . 50
2.2.2 Basic package management operations with the commandline . . . 51
2.2.3 Interactive use of aptitude . . . 53
2.2.4 Key bindings of aptitude . . . 53
2.2.5 Package views under aptitude . . . 54
2.2.6 Search method options with aptitude . . . 55
2.2.7 The aptitude regex formula . . . 55
2.2.8 Dependency resolution of aptitude . . . 57
2.2.9 Package activity logs . . . 57
2.3 Examples of aptitude operations . . . 57
2.3.1 Seeking interesting packages . . . 57
2.3.2 Listing packages with regex matching on package names . . . 58
2.3.3 Browsing with the regex matching . . . 58
2.3.4 Purging removed packages for good . . . 58
2.3.5 Tidying auto/manual install status . . . 58
2.3.6 System wide upgrade . . . 59
2.4 Advanced package management operations . . . 60
2.4.1 Advanced package management operations with commandline . . . 60
2.4.2 Verification of installed package files . . . 62
2.4.3 Safeguarding for package problems . . . 62
2.4.4 Searching on the package meta data . . . 62
2.5 Debian package management internals . . . 62
2.5.1 Archive meta data . . . 63
2.5.2 Top level ”Release” file and authenticity . . . 63
2.5.3 Archive level ”Release” files . . . 64

2.5.4 Fetching of the meta data for the package . . . 65
2.5.5 The package state for APT . . . 65
2.5.6 The package state for aptitude . . . 65
2.5.7 Local copies of the fetched packages . . . 66
2.5.8 Debian package file names . . . 66
2.5.9 The dpkg command . . . 67
2.5.10 The update-alternatives command . . . 67
2.5.11 The dpkg-statoverride command . . . 68
2.5.12 The dpkg-divert command . . . 68
2.6 Recovery from a broken system . . . 68
2.6.1 Incompatibility with old user configuration . . . 69
2.6.2 Caching errors of the package data . . . 69
2.6.3 Rescue with the dpkg command . . . 69
2.6.4 Failed installation due to missing dependencies . . . 69
2.6.5 Different packages with overlapped files . . . 70
2.6.6 Fixing broken package script . . . 70
2.6.7 Recovering package selection data . . . 70
2.7 Tips for the package management . . . 71
2.7.1 Who uploaded the package? . . . 71
2.7.2 Limiting download bandwidth for APT . . . 71
2.7.3 Automatic download and upgrade of packages . . . 71
2.7.4 Updates and Backports . . . 72
2.7.5 External package archives . . . 72
2.7.6 Packages from mixed source of archives without **apt-pinning** . . . 72
2.7.7 Tweaking candidate version with **apt-pinning** . . . 73
2.7.8 Blocking packages installed by ”Recommends” . . . 75
2.7.9 Tracking `testing` with some packages from `unstable` . . . 75
2.7.10 Tracking `unstable` with some packages from `experimental` . . . 77
2.7.11 Emergency downgrading . . . 77
2.7.12 The equivs package . . . 78
2.7.13 Porting a package to the stable system . . . 78
2.7.14 Proxy server for APT . . . 79
2.7.15 More readings for the package management . . . 79

3 The system initialization 80
3.1 An overview of the boot strap process . . . 80
3.1.1 Stage 1: the UEFI . . . 80
3.1.2 Stage 2: the boot loader . . . 81
3.1.3 Stage 3: the mini-Debian system . . . 82
3.1.4 Stage 4: the normal Debian system . . . 83
3.2 Rescue system . . . 83
3.2.1 GRUB UEFI rescue system on USB . . . 84
3.2.2 Linux live rescue system on USB . . . 85
3.2.3 Linux live rescue system from GRUB . . . 86
3.3 Systemd . . . 86
3.3.1 Systemd init . . . 86
3.3.2 Systemd login . . . 87
3.4 The kernel message . . . 87
3.5 The system message . . . 88
3.6 System management . . . 88
3.7 Other system monitors . . . 90
3.8 System configuration . . . 90
3.8.1 The hostname . . . 90
3.8.2 The filesystem . . . 90
3.8.3 Network interface initialization . . . 90
3.8.4 Cloud system initialization . . . 91
3.8.5 Customization example to tweak sshd service . . . 91
3.9 The udev system . . . 91
3.10 The kernel module initialization . . . 92

4 Authentication and access controls 93
4.1 Normal Unix authentication . . . 93
4.2 Managing account and password information . . . 95
4.3 Good password . . . 95
4.4 Creating encrypted password . . . 96
4.5 PAM and NSS . . . 96
4.5.1 Configuration files accessed by PAM and NSS . . . 97
4.5.2 The modern centralized system management . . . 97
4.5.3 ”Why GNU su does not support the wheel group” . . . 98
4.5.4 Stricter password rule . . . 98
4.6 Security of authentication . . . 99
4.6.1 Secure password on the Internet . . . 99
4.6.2 Secure Shell . . . 99

4.6.3 Extra security measures for the Internet . . . 99
4.6.4 Securing the root password . . . 100
4.7 Other access controls . . . 100
4.7.1 Access control lists (ACLs) . . . 101
4.7.2 sudo . . . 101
4.7.3 PolicyKit . . . 101
4.7.4 Restricting access to some server services . . . 102
4.7.5 Linux security features . . . 102

5 Network setup 104
5.1 The basic network infrastructure . . . 104
5.1.1 The hostname resolution . . . 104
5.1.2 The network interface name . . . 106
5.1.3 The network address range for the LAN . . . 107
5.1.4 The network device support . . . 107
5.2 The modern network configuration for desktop . . . 107
5.2.1 GUI network configuration tools . . . 108
5.3 The modern network configuration without GUI . . . 108
5.4 The modern network configuration for cloud . . . 109
5.4.1 The modern network configuration for cloud with DHCP . . . 109
5.4.2 The modern network configuration for cloud with static IP . . . 109
5.4.3 The modern network configuration for cloud with Network Manager . . . 109
5.5 The low level network configuration . . . 110
5.5.1 Iproute2 commands . . . 110
5.5.2 Safe low level network operations . . . 110
5.6 Network optimization . . . 111
5.6.1 Finding optimal MTU . . . 111
5.6.2 WAN TCP optimization . . . 112
5.7 Netfilter infrastructure . . . 112

6 Network applications 114
6.1 Web browsers . . . 114
6.1.1 Spoofing the User-Agent string . . . 115
6.1.2 Browser extension . . . 115
6.2 The mail system . . . 115
6.2.1 Email basics . . . 115
6.2.2 Modern mail service limitation . . . 116
6.2.3 Historic mail service expectation . . . 116
6.2.4 Mail transport agent (MTA) . . . 117

6.2.4.1 The configuration of exim4 . . . 117
6.2.4.2 The configuration of postfix with SASL . . . 119
6.2.4.3 The mail address configuration . . . 120
6.2.4.4 Basic MTA operations . . . 121
6.3 The remote access server and utilities (SSH) . . . 121
6.3.1 Basics of SSH . . . 122
6.3.2 User name on the remote host . . . 123
6.3.3 Connecting without remote passwords . . . 123
6.3.4 Dealing with alien SSH clients . . . 123
6.3.5 Setting up ssh-agent . . . 123
6.3.6 Sending a mail from a remote host . . . 124
6.3.7 Port forwarding for SMTP/POP3 tunneling . . . 124
6.3.8 How to shutdown the remote system on SSH . . . 124
6.3.9 Troubleshooting SSH . . . 124
6.4 The print server and utilities . . . 124
6.5 Other network application servers . . . 125
6.6 Other network application clients . . . 126
6.7 The diagnosis of the system daemons . . . 126

7 GUI System 128

7.1 GUI desktop environment . . . 128
7.2 GUI communication protocol . . . 129
7.3 GUI infrastructure . . . 130
7.4 GUI applications . . . 130
7.5 User directories . . . 132
7.6 Fonts . . . 132
7.6.1 Basic fonts . . . 132
7.6.2 Font rasterization . . . 134
7.7 Sandbox . . . 134
7.8 Remote desktop . . . 135
7.9 X server connection . . . 135
7.9.1 X server local connection . . . 135
7.9.2 X server remote connection . . . 135
7.9.3 X server chroot connection . . . 137
7.10 Clipboard . . . 137

8 I18N and L10N 138

8.1 The locale . . . 138
8.1.1 Rationale for UTF-8 locale . . . 138
8.1.2 The reconfiguration of the locale . . . 139
8.1.3 Filename encoding . . . 140
8.1.4 Localized messages and translated documentation . . . 140
8.1.5 Effects of the locale . . . 140
8.2 The keyboard input . . . 141
8.2.1 The keyboard input for Linux console and X Window . . . 141
8.2.2 The keyboard input for Wayland . . . 142
8.2.3 The input method support with IBus . . . 142
8.2.4 The input method support with Fcitx . . . 142
8.2.5 An example for Japanese . . . 142
8.3 The display output . . . 145
8.3.1 The terminal configuration . . . 145
8.3.2 East Asian Ambiguous Character Width Characters . . . 146

9 System tips 147

9.1 The console tips . . . 147
9.1.1 Recording the shell activities cleanly . . . 147
9.1.2 The screen program . . . 148
9.1.3 Navigating around directories . . . 149
9.1.4 Readline wrapper . . . 149
9.1.5 Scanning the source code tree . . . 149
9.2 Customizing vim . . . 150
9.2.1 Customizing vim with internal features . . . 150
9.2.2 Customizing vim with external packages . . . 152
9.3 Data recording and presentation . . . 153
9.3.1 The log daemon . . . 153
9.3.2 Log analyzer . . . 153
9.3.3 Customized display of text data . . . 154
9.3.4 Customized display of time and date . . . 154
9.3.5 Colorized shell echo . . . 154
9.3.6 Colorized commands . . . 155
9.3.7 Recording the editor activities for complex repeats . . . 155
9.3.8 Recording the graphics image of an X application . . . 156
9.3.9 Recording changes in configuration files . . . 156
9.4 Monitoring, controlling, and starting program activities . . . 157
9.4.1 Timing a process . . . 157

9.4.2 The scheduling priority . . . 157
9.4.3 The ps command . . . 158
9.4.4 The top command . . . 158
9.4.5 Listing files opened by a process . . . 158
9.4.6 Tracing program activities . . . 158
9.4.7 Identification of processes using files or sockets . . . 159
9.4.8 Repeating a command with a constant interval . . . 159
9.4.9 Repeating a command looping over files . . . 159
9.4.10 Starting a program from GUI . . . 160
9.4.11 Customizing program to be started . . . 161
9.4.12 Killing a process . . . 162
9.4.13 Scheduling tasks once . . . 162
9.4.14 Scheduling tasks regularly . . . 162
9.4.15 Scheduling tasks on event . . . 163
9.4.16 Alt-SysRq key . . . 163
9.5 System maintenance tips . . . 164
9.5.1 Who is on the system? . . . 164
9.5.2 Warning everyone . . . 164
9.5.3 Hardware identification . . . 164
9.5.4 Hardware configuration . . . 166
9.5.5 System and hardware time . . . 166
9.5.6 The sound infrastructure . . . 167
9.5.7 Disabling the screen saver . . . 167
9.5.8 Disabling beep sounds . . . 167
9.5.9 Memory usage . . . 168
9.5.10 System security and integrity check . . . 168
9.6 Data storage tips . . . 169
9.6.1 Disk space usage . . . 169
9.6.2 Disk partition configuration . . . 170
9.6.3 Accessing partition using UUID . . . 170
9.6.4 LVM2 . . . 171
9.6.5 Filesystem configuration . . . 171
9.6.6 Filesystem creation and integrity check . . . 172
9.6.7 Optimization of filesystem by mount options . . . 172
9.6.8 Optimization of filesystem via superblock . . . 173
9.6.9 Optimization of hard disk . . . 173
9.6.10 Optimization of solid state drive . . . 173
9.6.11 Using SMART to predict hard disk failure . . . 173
9.6.12 Specify temporary storage directory via $TMPDIR . . . 174

9.6.13 Expansion of usable storage space via LVM . . . 174
9.6.14 Expansion of usable storage space by mounting another partition . . . 174
9.6.15 Expansion of usable storage space by bind-mounting another directory . . . 174
9.6.16 Expansion of usable storage space by overlay-mounting another directory . . . 175
9.6.17 Expansion of usable storage space using symlink . . . 175
9.7 The disk image . . . 175
9.7.1 Making the disk image file . . . 175
9.7.2 Writing directly to the disk . . . 176
9.7.3 Mounting the disk image file . . . 176
9.7.4 Cleaning a disk image file . . . 178
9.7.5 Making the empty disk image file . . . 178
9.7.6 Making the ISO9660 image file . . . 179
9.7.7 Writing directly to the CD/DVD-R/RW . . . 179
9.7.8 Mounting the ISO9660 image file . . . 180
9.8 The binary data . . . 180
9.8.1 Viewing and editing binary data . . . 180
9.8.2 Manipulating files without mounting disk . . . 181
9.8.3 Data redundancy . . . 181
9.8.4 Data file recovery and forensic analysis . . . 181
9.8.5 Splitting a large file into small files . . . 182
9.8.6 Clearing file contents . . . 182
9.8.7 Dummy files . . . 182
9.8.8 Erasing an entire hard disk . . . 182
9.8.9 Erasing unused area of an hard disk . . . 183
9.8.10 Undeleting deleted but still open files . . . 183
9.8.11 Searching all hardlinks . . . 184
9.8.12 Invisible disk space consumption . . . 184
9.9 Data encryption tips . . . 184
9.9.1 Removable disk encryption with dm-crypt/LUKS . . . 185
9.9.2 Mounting encrypted disk with dm-crypt/LUKS . . . 185
9.10 The kernel . . . 186
9.10.1 Kernel parameters . . . 186
9.10.2 Kernel headers . . . 186
9.10.3 Compiling the kernel and related modules . . . 186
9.10.4 Compiling the kernel source: Debian Kernel Team recommendation . . . 187
9.10.5 Hardware drivers and firmware . . . 188
9.11 Virtualized system . . . 189
9.11.1 Virtualization and emulation tools . . . 189
9.11.2 Virtualization work flow . . . 189
9.11.3 Mounting the virtual disk image file . . . 191
9.11.4 Chroot system . . . 191
9.11.5 Multiple desktop systems . . . 192

10 Data management **194**

10.1 Sharing, copying, and archiving . . . 194
10.1.1 Archive and compression tools . . . 195
10.1.2 Copy and synchronization tools . . . 195
10.1.3 Idioms for the archive . . . 195
10.1.4 Idioms for the copy . . . 197
10.1.5 Idioms for the selection of files . . . 198
10.1.6 Archive media . . . 199
10.1.7 Removable storage device . . . 200
10.1.8 Filesystem choice for sharing data . . . 201
10.1.9 Sharing data via network . . . 202
10.2 Backup and recovery . . . 203
10.2.1 Backup and recovery policy . . . 203
10.2.2 Backup utility suites . . . 204
10.2.3 Backup tips . . . 204
10.2.3.1 GUI backup . . . 206
10.2.3.2 Mount event triggered backup . . . 206
10.2.3.3 Timer event triggered backup . . . 207
10.3 Data security infrastructure . . . 208
10.3.1 Key management for GnuPG . . . 208
10.3.2 Using GnuPG on files . . . 210
10.3.3 Using GnuPG with Mutt . . . 210
10.3.4 Using GnuPG with Vim . . . 210
10.3.5 The MD5 sum . . . 212
10.3.6 Password keyring . . . 212
10.4 Source code merge tools . . . 212
10.4.1 Extracting differences for source files . . . 212
10.4.2 Merging updates for source files . . . 214
10.4.3 Interactive merge . . . 214
10.5 Git . . . 214
10.5.1 Configuration of Git client . . . 215
10.5.2 Basic Git commands . . . 215
10.5.3 Git tips . . . 216
10.5.4 Git references . . . 216
10.5.5 Other version control systems . . . 218

11 Data conversion 219
11.1 Text data conversion tools . . . 219
11.1.1 Converting a text file with iconv . . . 219
11.1.2 Checking file to be UTF-8 with iconv . . . 221
11.1.3 Converting file names with iconv . . . 221
11.1.4 EOL conversion . . . 221
11.1.5 TAB conversion . . . 222
11.1.6 Editors with auto-conversion . . . 222
11.1.7 Plain text extraction . . . 223
11.1.8 Highlighting and formatting plain text data . . . 223
11.2 XML data . . . 223
11.2.1 Basic hints for XML . . . 224
11.2.2 XML processing . . . 225
11.2.3 The XML data extraction . . . 225
11.2.4 The XML data lint . . . 227
11.3 Type setting . . . 227
11.3.1 roff typesetting . . . 227
11.3.2 TeX/LaTeX . . . 228
11.3.3 Pretty print a manual page . . . 228
11.3.4 Creating a manual page . . . 228
11.4 Printable data . . . 229
11.4.1 Ghostscript . . . 229
11.4.2 Merge two PS or PDF files . . . 229
11.4.3 Printable data utilities . . . 230
11.4.4 Printing with CUPS . . . 230
11.5 The mail data conversion . . . 231
11.5.1 Mail data basics . . . 231
11.6 Graphic data tools . . . 232
11.6.1 Graphic data tools (metapackage) . . . 232
11.6.2 Graphic data tools (GUI) . . . 232
11.6.3 Graphic data tools (CLI) . . . 232
11.7 Miscellaneous data conversion . . . 235

12 Programming 236
12.1 The shell script . . . 236
12.1.1 POSIX shell compatibility . . . 237
12.1.2 Shell parameters . . . 237
12.1.3 Shell conditionals . . . 239
12.1.4 Shell loops . . . 239

12.1.5 Shell environment variables . . . 240
12.1.6 The shell command-line processing sequence . . . 240
12.1.7 Utility programs for shell script . . . 241
12.2 Scripting in interpreted languages . . . 243
12.2.1 Debugging interpreted language codes . . . 243
12.2.2 GUI program with the shell script . . . 243
12.2.3 Custom actions for GUI filer . . . 244
12.2.4 Perl short script madness . . . 244
12.3 Coding in compiled languages . . . 245
12.3.1 C . . . 245
12.3.2 Simple C program (gcc) . . . 246
12.3.3 Flex — a better Lex . . . 246
12.3.4 Bison — a better Yacc . . . 246
12.4 Static code analysis tools . . . 248
12.5 Debug . . . 248
12.5.1 Basic gdb execution . . . 248
12.5.2 Debugging the Debian package . . . 250
12.5.3 Obtaining backtrace . . . 251
12.5.4 Advanced gdb commands . . . 251
12.5.5 Check dependency on libraries . . . 252
12.5.6 Dynamic call tracing tools . . . 252
12.5.7 Debugging X Errors . . . 252
12.5.8 Memory leak detection tools . . . 252
12.5.9 Disassemble binary . . . 252
12.6 Build tools . . . 253
12.6.1 Make . . . 253
12.6.2 Autotools . . . 254
12.6.2.1 Compile and install a program . . . 254
12.6.2.2 Uninstall program . . . 254
12.6.3 Meson . . . 254
12.7 Web . . . 255
12.8 The source code translation . . . 255
12.9 Making Debian package . . . 256

A Appendix **257**

A.1 The Debian maze . . . 257
A.2 Copyright history . . . 257
A.3 Document format . . . 258

List of Tables

1.1 List of interesting text-mode program packages . . . 4
1.2 List of informative documentation packages . . . 5
1.3 List of usage of key directories . . . 7
1.4 List of the first character of "`ls -l`" output . . . 9
1.5 The numeric mode for file permissions in chmod(1) commands . . . 10
1.6 The **umask** value examples . . . 11
1.7 List of notable system-provided groups for file access . . . 12
1.8 List of notable system provided groups for particular command executions . . . 12
1.9 List of types of timestamps . . . 12
1.10 List of special device files . . . 16
1.11 The key bindings of MC . . . 18
1.12 The reaction to the enter key in MC . . . 19
1.13 List of shell programs . . . 20
1.14 List of key bindings for bash . . . 22
1.15 List of mouse operations and related key actions on Debian . . . 22
1.16 List of basic Vim key strokes . . . 24
1.17 List of basic Unix commands . . . 26
1.18 The 3 parts of locale value . . . 27
1.19 List of locale recommendations . . . 28
1.20 List of "`$HOME`" values . . . 29
1.21 Shell glob patterns . . . 29
1.22 Command exit codes . . . 30
1.23 Shell command idioms . . . 31
1.24 Predefined file descriptors . . . 32
1.25 Metacharacters for BRE and ERE . . . 34
1.26 The replacement expression . . . 34
1.27 List of script snippets for piping commands . . . 38

2.1 List of Debian package management tools . . . 40
2.2 List of Debian archive sites . . . 43

2.3 List of Debian archive area . . . 44
2.4 The relationship between suite and codename . . . 44
2.5 List of key web site to resolving problems with a specific package . . . 49
2.6 Basic package management operations with the commandline using apt(8), aptitude(8) and apt-get(8) /apt-cache(8) . . . 52
2.7 Notable command options for aptitude(8) . . . 52
2.8 List of key bindings for aptitude . . . 53
2.9 List of views for aptitude . . . 54
2.10 The categorization of standard package views . . . 55
2.11 List of the aptitude regex formula . . . 56
2.12 The log files for package activities . . . 57
2.13 List of advanced package management operations . . . 61
2.14 The content of the Debian archive meta data . . . 63
2.15 The name structure of Debian packages . . . 66
2.16 The usable characters for each component in the Debian package names . . . 66
2.17 The notable files created by `dpkg` . . . 67
2.18 List of notable Pin-Priority values for **apt-pinning** technique. . . . 74
2.19 List of the proxy tools specially for Debian archive . . . 79

3.1 List of boot loaders . . . 81
3.2 The meaning of the menu entry of the above part of `/boot/grub/grub.cfg` . . . 82
3.3 List of boot utilities for the Debian system . . . 84
3.4 List of kernel error levels . . . 88
3.5 List of typical `journalctl` command snippets . . . 88
3.6 List of typical `systemctl` command snippets . . . 89
3.7 List of other monitoring command snippets under `systemd` . . . 90

4.1 3 important configuration files for pam_unix(8) . . . 93
4.2 The second entry content of "`/etc/passwd`" . . . 94
4.3 List of commands to manage account information . . . 95
4.4 List of tools to generate password . . . 96
4.5 List of notable PAM and NSS systems . . . 96
4.6 List of configuration files accessed by PAM and NSS . . . 97
4.7 List of insecure and secure services and ports . . . 99
4.8 List of tools to provide extra security measures . . . 100

5.1 List of network configuration tools . . . 105
5.2 List of network address ranges . . . 107
5.3 Translation table from obsolete `net-tools` commands to new `iproute2` commands . . . 110
5.4 List of low level network commands . . . 110

5.5 List of network optimization tools . . . 111
5.6 Basic guide lines of the optimal MTU value . . . 112
5.7 List of firewall tools . . . 113

6.1 List of web browsers . . . 114
6.2 List of mail user agent (MUA) . . . 116
6.3 List of basic mail transport agent related packages . . . 117
6.4 List of important postfix manual pages . . . 119
6.5 List of mail address related configuration files . . . 120
6.6 List of basic MTA operation . . . 121
6.7 List of remote access server and utilities . . . 121
6.8 List of SSH configuration files . . . 122
6.9 List of SSH client startup examples . . . 122
6.10 List of free SSH clients for other platforms . . . 123
6.11 List of print servers and utilities . . . 125
6.12 List of other network application servers . . . 125
6.13 List of network application clients . . . 126
6.14 List of popular RFCs . . . 127

7.1 List of desktop environment . . . 128
7.2 List of notable GUI infrastructure packages . . . 130
7.3 List of notable GUI applications . . . 131
7.4 List of notable TrueType and OpenType fonts . . . 133
7.5 List of notable font environment and related packages . . . 133
7.6 List of notable sandbox environment and related packages . . . 135
7.7 List of notable remote access server . . . 136
7.8 List of connection methods to the X server . . . 136
7.9 List of programs related to manipulating character clipboard . . . 137

8.1 List of IBus and its plugin packages . . . 143
8.2 List of Fcitx5 and its plugin packages . . . 144

9.1 List of programs to support console activities . . . 147
9.2 List of key bindings for screen . . . 149
9.3 Information on the initialization of `vim` . . . 153
9.4 List of system log analyzers . . . 154
9.5 Display examples of time and date for the "`ls -l`" command with the **time style value** . . . 155
9.6 List of graphics image manipulation tools . . . 156
9.7 List of packages which can record configuration history . . . 156
9.8 List of tools for monitoring and controlling program activities . . . 157

9.9 List of nice values for the scheduling priority . . . 158
9.10 List of ps command styles . . . 158
9.11 List of frequently used signals for kill command . . . 162
9.12 List of notable SAK command keys . . . 163
9.13 List of hardware identification tools . . . 165
9.14 List of hardware configuration tools . . . 165
9.15 List of sound packages . . . 167
9.16 List of commands for disabling the screen saver . . . 168
9.17 List of memory sizes reported . . . 168
9.18 List of tools for system security and integrity check . . . 169
9.19 List of disk partition management packages . . . 170
9.20 List of filesystem management packages . . . 172
9.21 List of packages which view and edit binary data . . . 180
9.22 List of packages to manipulate files without mounting disk . . . 181
9.23 List of tools to add data redundancy to files . . . 181
9.24 List of packages for data file recovery and forensic analysis . . . 181
9.25 List of data encryption utilities . . . 185
9.26 List of key packages to be installed for the kernel recompilation on the Debian system . . . 187
9.27 List of virtualization tools . . . 190

10.1 List of archive and compression tools . . . 196
10.2 List of copy and synchronization tools . . . 197
10.3 List of filesystem choices for removable storage devices with typical usage scenarios . . . 201
10.4 List of the network service to choose with the typical usage scenario . . . 202
10.5 List of backup suite utilities . . . 205
10.6 List of data security infrastructure tools . . . 208
10.7 List of GNU Privacy Guard commands for the key management . . . 209
10.8 List of the meaning of the trust code . . . 209
10.9 List of GNU Privacy Guard commands on files . . . 211
10.10 List of source code merge tools . . . 213
10.11 List of git related packages and commands . . . 214
10.12 Main Git commands . . . 216
10.13 Git tips . . . 217
10.14 List of other version control system tools . . . 218

11.1 List of text data conversion tools . . . 219
11.2 List of encoding values and their usage . . . 220
11.3 List of EOL styles for different platforms . . . 221
11.4 List of TAB conversion commands from `bsdmainutils` and `coreutils` packages . . . 222

11.5 List of tools to extract plain text data . . . 223
11.6 List of tools to highlight plain text data . . . 224
11.7 List of predefined entities for XML . . . 225
11.8 List of XML tools . . . 226
11.9 List of DSSSL tools . . . 226
11.10 List of XML data extraction tools . . . 226
11.11 List of XML pretty print tools . . . 227
11.12 List of type setting tools . . . 227
11.13 List of packages to help creating the manpage . . . 229
11.14 List of Ghostscript PostScript interpreters . . . 229
11.15 List of printable data utilities . . . 230
11.16 List of packages to help mail data conversion . . . 231
11.17 List of graphics data tools (metapackage) . . . 232
11.18 List of graphics data tools (GUI) . . . 233
11.19 List of graphics data tools (CLI) . . . 234
11.20 List of miscellaneous data conversion tools . . . 235

12.1 List of typical bashisms . . . 237
12.2 List of shell parameters . . . 238
12.3 List of shell parameter expansions . . . 238
12.4 List of key shell parameter substitutions . . . 238
12.5 List of file comparison operators in the conditional expression . . . 239
12.6 List of string comparison operators in the conditional expression . . . 240
12.7 List of packages containing small utility programs for shell scripts . . . 242
12.8 List of interpreter related packages . . . 242
12.9 List of dialog programs . . . 243
12.10 List of compiler related packages . . . 245
12.11 List of Yacc-compatible LALR parser generators . . . 246
12.12 List of tools for static code analysis . . . 249
12.13 List of debug packages . . . 249
12.14 List of advanced gdb commands . . . 251
12.15 List of memory leak detection tools . . . 252
12.16 List of build tool packages . . . 253
12.17 List of make automatic variables . . . 253
12.18 List of make variable expansions . . . 253
12.19 List of source code translation tools . . . 255

Abstract

This book is free; you may redistribute it and/or modify it under the terms of the GNU General Public License of any version compliant to the Debian Free Software Guidelines (DFSG).

Preface

This Debian Reference (version 2.152) (2026-10-10 14:56:25 UTC) is intended to provide a broad overview of the Debian system administration as a post-installation user guide.

The target reader is someone who is willing to learn shell scripts but who is not ready to read all the C sources to figure out how the GNU/Linux system works.

For installation instructions, see:

- Debian GNU/Linux Installation Guide for current stable system
- Debian GNU/Linux Installation Guide for current testing system

Disclaimer

All warranties are disclaimed. All trademarks are property of their respective trademark owners.

The Debian system itself is a moving target. This makes its documentation difficult to be current and correct. Although the current `testing` version of the Debian system was used as the basis for writing this, some contents may be already outdated by the time you read this.

Please treat this document as the secondary reference. This document does not replace any authoritative guides. The author and contributors do not take responsibility for consequences of errors, omissions or ambiguity in this document.

What is Debian

The Debian Project is an association of individuals who have made common cause to create a free operating system. It's distribution is characterized by the following.

- Commitment to the software freedom: Debian Social Contract and Debian Free Software Guidelines (DFSG)
- Internet based distributed unpaid volunteer effort: https://www.debian.org
- Large number of pre-compiled high quality software packages
- Focus on stability and security with easy access to the security updates
- Focus on smooth upgrade to the latest software packages in the `testing` archives
- Large number of supported hardware architectures

Free Software pieces in Debian come from GNU, Linux, BSD, X, ISC, Apache, Ghostscript, Common Unix Printing System , Samba, GNOME, KDE, Mozilla, LibreOffice, Vim, TeX, LaTeX, DocBook, Perl, Python, Tcl, Java, Ruby, PHP, Berkeley DB, MariaDB, PostgreSQL, SQLite, Exim, Postfix, Mutt, FreeBSD, OpenBSD, Plan 9 and many more independent free software projects. Debian integrates this diversity of Free Software into one system.

About this document

Guiding rules

Following guiding rules are followed while compiling this document.

- Provide overview and skip corner cases. (**Big Picture**)
- Keep It Short and Simple. (**KISS**)
- Do not reinvent the wheel. (Use pointers to **the existing references**)
- Focus on non-GUI tools and consoles. (Use **shell examples**)
- Be objective. (Use popcon etc.)

Tip
I tried to elucidate hierarchical aspects and lower levels of the system.

Prerequisites



Warning
You are expected to make good efforts to seek answers by yourself beyond this documentation. This document only gives efficient starting points.

You must seek solution by yourself from primary sources.

- The Debian site at https://www.debian.org for the general information
- The documentation under the "`/usr/share/doc/`*`package_name`*" directory
- The Unix style **manpage**: "`dpkg -L` *`package_name`* `|grep '/man/man.*/'`"
- The GNU style **info page**: "`dpkg -L` *`package_name`* `|grep '/info/'`"
- The bug report: https://bugs.debian.org/*package_name*
- The Debian Wiki at https://wiki.debian.org/ for the moving and specific topics
- The Single UNIX Specification from the Open Group's The UNIX System Home Page
- The free encyclopedia from Wikipedia at https://www.wikipedia.org/
- The Debian Administrator's Handbook
- The HOWTOs from The Linux Documentation Project (TLDP)

Note
For detailed documentation, you may need to install the corresponding documentation package named with "`-doc`" as its suffix.

Conventions

This document provides information through the following simplified presentation style with bash(1) shell command examples.

```
# command-in-root-account
$ command-in-user-account
```

These shell prompts distinguish account used and correspond to set environment variables as: "`PS1='\$'`" and "`PS2=' '`". These values are chosen for the sake of readability of this document and are not typical on actual installed system.

All command examples are run under the English locale "`LANG=en_US.UTF8`". Please don't expect the placeholder strings such as *`command-in-root-account`* and *`command-in-user-account`* to be translated in command examples. This is an intentional choice to keep all translated examples to be up-to-date.

Note
See the meaning of the "`$PS1`" and "`$PS2`" environment variables in bash(1).

Action required by the system administrator is written in the imperative sentence, e.g. "Type Enter-key after typing each command string to the shell."

The **description** column and similar ones in the table may contain a **noun phrase** following the package short description convention which drops leading articles such as "a" and "the". They may alternatively contain an infinitive phrase as a **noun phrase** without leading "to" following the short command description convention in manpages. These may look funny to some people but are my intentional choices of style to keep this documentation as simple as possible. These **Noun phrases** do not capitalize their starting nor end with periods following these short description convention.

Note
Proper nouns including command names keeps their case irrespective of their location.

A **command snippet** quoted in a text paragraph is referred by the typewriter font between double quotation marks, such as "`aptitude safe-upgrade`".

A **text data** from a configuration file quoted in a text paragraph is referred by the typewriter font between double quotation marks, such as "`deb-src`".

A **command** is referred by its name in the typewriter font optionally followed by its manpage section number in parenthesis, such as bash(1). You are encouraged to obtain information by typing the following.

```
$ man 1 bash
```

A **manpage** is referred by its name in the typewriter font followed by its manpage section number in parenthesis, such as sources.list(5). You are encouraged to obtain information by typing the following.

```
$ man 5 sources.list
```

An **info page** is referred by its command snippet in the typewriter font between double quotation marks, such as "`info make`". You are encouraged to obtain information by typing the following.

```
$ info make
```

A **filename** is referred by the typewriter font between double quotation marks, such as "`/etc/passwd`". For configuration files, you are encouraged to obtain information by typing the following.

```
$ sensible-pager "/etc/passwd"
```

A **directory name** is referred by the typewriter font between double quotation marks, such as "`/etc/apt/`". You are encouraged to explore its contents by typing the following.

```
$ mc "/etc/apt/"
```

A **package name** is referred by its name in the typewriter font, such as `vim`. You are encouraged to obtain information by typing the following.

```
$ dpkg -L vim
$ apt-cache show vim
$ aptitude show vim
```

A **documentation** may indicate its location by the filename in the typewriter font between double quotation marks, such as "`/usr/share/doc/base-passwd/users-and-groups.txt.gz`" and "`/usr/share/doc/base-passwd/users-` or by its URL, such as https://www.debian.org. You are encouraged to read the documentation by typing the following.

```
$ zcat "/usr/share/doc/base-passwd/users-and-groups.txt.gz" | sensible-pager
$ sensible-browser "/usr/share/doc/base-passwd/users-and-groups.html"
$ sensible-browser "https://www.debian.org"
```

An **environment variable** is referred by its name with leading "$" in the typewriter font between double quotation marks, such as "`$TERM`". You are encouraged to obtain its current value by typing the following.

```
$ echo "$TERM"
```

The popcon

The popcon data is presented as the objective measure for the popularity of each package. It was downloaded on 2026-10-07 05:10:21 UTC and contains the total submission of 291550 reports over 224298 binary packages and 25 architectures.

Note
Please note that the `amd64` `unstable` archive contains only 78072 packages currently. The popcon data contains reports from many old system installations.

The popcon number preceded with "V:" for "votes" is calculated by "1000 * (the popcon submissions for the package executed recently on the PC)/(the total popcon submissions)".

The popcon number preceded with "I:" for "installs" is calculated by "1000 * (the popcon submissions for the package installed on the PC)/(the total popcon submissions)".

Note
The popcon figures should not be considered as absolute measures of the importance of packages. There are many factors which can skew statistics. For example, some system participating popcon may have mounted directories such as "`/usr/bin`" with "`noatime`" option for system performance improvement and effectively disabled "vote" from such system.

The package size

The package size data is also presented as the objective measure for each package. It is based on the "`Installed-Size:`" reported by "`apt-cache show`" or "`aptitude show`" command (currently on `amd64` architecture for the `unstable` release). The reported size is in KiB (Kibibyte = unit for 1024 bytes).

Note
A package with a small numerical package size may indicate that the package in the `unstable` release is a dummy package which installs other packages with significant contents by the dependency. The dummy package enables a smooth transition or split of the package.

Note
A package size followed by "(*)" indicates that the package in the `unstable` release is missing and the package size for the `experimental` release is used instead.

Bug reports on this document

Please file bug reports on the `debian-reference` package using reportbug(1) if you find any issues on this document. Please include correction suggestion by "`diff -u`" to the plain text version or to the source.

Reminders for new users

Here are some reminders for new users:

- Backup your data
  - See Section 10.2.
- Secure your password and security keys
- KISS (keep it simple stupid)
  - Don't over-engineer your system
- Read your log files
  - The **FIRST** error is the one that counts
- RTFM (read the fine manual)
- Search the Internet before asking questions
- Don't be root when you don't have to be
- Don't mess with the package management system
- Don't type anything you don't understand
- Don't change the file permissions (before the full security review)
- Don't leave your root shell until you **TEST** your changes
- Always have an alternative boot media (USB flash drive, CD-ROM, …)

Some quotes for new users

Here are some interesting quotes from the Debian mailing list which may help enlighten new users.

- "This is Unix. It gives you enough rope to hang yourself." --- Miquel van Smoorenburg <`miquels at cistron.nl`>
- "Unix IS user friendly… It's just selective about who its friends are." --- Tollef Fog Heen <`tollef at add.no`>

Wikipedia has article "Unix philosophy" which lists interesting quotes.

Chapter 1

GNU/Linux tutorials

I think learning a computer system is like learning a new foreign language. Although tutorial books and documentation are helpful, you have to practice it yourself. In order to help you get started smoothly, I elaborate a few basic points.

The powerful design of Debian GNU/Linux comes from the Unix operating system, i.e., a multiuser, multitasking operating system. You must learn to take advantage of the power of these features and similarities between Unix and GNU/Linux.

Don't shy away from Unix oriented texts and don't rely solely on GNU/Linux texts, as this robs you of much useful information.

Note
If you have been using any Unix-like system for a while with command line tools, you probably know everything I explain here. Please use this as a reality check and refresher.

1.1 Console basics

1.1.1 The shell prompt

Upon starting the system, you are presented with the character based login screen if you did not install any GUI environment such as GNOME or KDE desktop system. Suppose your hostname is `foo`, the login prompt looks as follows.

If you installed a GUI environment, then you can still get to the character based login prompt by Ctrl-Alt-F3, and you can return to the GUI environment via Ctrl-Alt-F2 (see Section 1.1.6 below for more).

```
foo login:
```

At the login prompt, you type your username, e.g. `penguin`, and press the Enter-key, then type your password and press the Enter-key again.

Note
Following the Unix tradition, the username and password of the Debian system are case sensitive. The username is usually chosen only from the lowercase. The first user account is usually created during the installation. Additional user accounts can be created with adduser(8) by root.

The system starts with the greeting message stored in "`/etc/motd`" (Message Of The Day) and presents a command prompt.

```
Debian GNU/Linux 12 foo tty3

foo login: penguin
Password:

Linux foo 6.5.0-0.deb12.4-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.5.10-1~bpo12+1 (2023-11-23) ↩
     x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.

Last login: Wed Dec 20 09:39:00 JST 2023 on tty3
foo:~$
```

Now you are in the shell. The shell interprets your commands.

1.1.2 The shell prompt under GUI

If you installed a GUI environment during the installation, you are presented with the graphical login screen upon starting your system. You type your username and your password to login to the non-privileged user account. Use tab to navigate between username and password, or use the primary click of the mouse.

You can gain the shell prompt under GUI environment by starting a `x-terminal-emulator` program such as gnome-terminal(1), rxvt(1) or xterm(1). Under the GNOME desktop environment, press SUPER-key (Windows-key) and typing in "terminal" to the search prompt does the trick.

Under some other Desktop systems (like `fluxbox`), there may be no obvious starting point for the menu. If this happens, just try (right) clicking the background of the desktop screen and hope for a menu to pop-up.

1.1.3 The root account

The root account is also called superuser or privileged user. From this account, you can perform the following system administration tasks.

- Read, write, and remove any files on the system irrespective of their file permissions
- Set file ownership and permissions of any files on the system
- Set the password of any non-privileged users on the system
- Login to any accounts without their passwords

This unlimited power of root account requires you to be considerate and responsible when using it.



Warning
Never share the root password with others.

Note
File permissions of a file (including hardware devices such as CD-ROM etc. which are just another file for the Debian system) may render it unusable or inaccessible by non-root users. Although the use of root account is a quick way to test this kind of situation, its resolution should be done through proper setting of file permissions and user's group membership (see Section 1.2.3).

1.1.4 The root shell prompt

Here are a few basic methods to gain the root shell prompt by using the root password.

- Type `root` at the character based login prompt.
- Type "`su -l`" from any user shell prompt.
  - This does not preserve the environment of the current user.
- Type "`su`" from any user shell prompt.
  - This preserves some of the environment of the current user.

1.1.5 GUI system administration tools

When your desktop menu does not start GUI system administration tools automatically with the appropriate privilege, you can start them from the root shell prompt of the terminal emulator, such as gnome-terminal(1), rxvt(1), or xterm(1). See Section 1.1.4 and Section 7.9.



Warning
Never start the GUI display/session manager under the root account by typing in `root` to the prompt of the display manager such as gdm3(1).
Never run untrusted remote GUI program under X Window when critical information is displayed since it may eavesdrop your X screen.

1.1.6 Virtual consoles

In the default Debian system, there are six switchable VT100-like character consoles available to start the command shell directly on the Linux host. Unless you are in a GUI environment, you can switch between the virtual consoles by pressing the `Left-Alt-key` and one of the `F1` — `F6` keys simultaneously. Each character console allows independent login to the account and offers the multiuser environment. This multiuser environment is a great Unix feature, and very addictive.

If you are in the GUI environment, you gain access to the character console 3 by pressing `Ctrl-Alt-F3` key, i.e., the `left-Ctrl-key`, the `left-Alt-key`, and the `F3-key` are pressed together. You can get back to the GUI environment, normally running on the virtual console 2, by pressing `Alt-F2`.

You can alternatively change to another virtual console, e.g. to the console 3, from the commandline.

```
# chvt 3
```

1.1.7 How to leave the command prompt

You type `Ctrl-D`, i.e., the `left-Ctrl-key` and the `d-key` pressed together, at the command prompt to close the shell activity. If you are at the character console, you return to the login prompt with this. Even though these control characters are referred as "control D" with the upper case, you do not need to press the Shift-key. The short hand expression, `^D`, is also used for `Ctrl-D`. Alternately, you can type "exit".

If you are at x-terminal-emulator(1), you can close `x-terminal-emulator` window with this.

1.1.8 How to shutdown the system

Just like any other modern OS where the file operation involves caching data in memory for improved performance, the Debian system needs the proper shutdown procedure before power can safely be turned off. This is to maintain the integrity of files, by forcing all changes in memory to be written to disk. If the software power control is available, the shutdown procedure automatically turns off power of the system. (Otherwise, you may have to press power button for few seconds after the shutdown procedure.)

You can shutdown the system under the normal multiuser mode from the commandline.

```
# shutdown -h now
```

You can shutdown the system under the single-user mode from the commandline.

```
# poweroff -i -f
```

See Section 6.3.8.

1.1.9 Recovering a sane console

When the screen goes berserk after doing some funny things such as "`cat` *`some-binary-file`*", type "`reset`" at the command prompt. You may not be able to see the command echoed as you type. You may also issue "`clear`" to clean up the screen.

1.1.10 Additional package suggestions for the newbie

Although even the minimal installation of the Debian system without any desktop environment tasks provides the basic Unix functionality, it is a good idea to install few additional commandline and curses based character terminal packages such as `mc` and `vim` with apt-get(8) for beginners to get started by the following.

```
# apt-get update
 ...
# apt-get install mc vim sudo aptitude
 ...
```

If you already had these packages installed, no new packages are installed.

package	popcon	size	description
mc	V:40, I:176	1590	A text-mode full-screen file manager
sudo	V:761, I:869	6814	A program to allow limited root privileges to users
vim	V:83, I:338	4164	Unix text editor Vi IMproved, a programmers text editor (standard version)
vim-tiny	V:59, I:980	1867	Unix text editor Vi IMproved, a programmers text editor (compact version)
emacs-nox	V:3, I:12	46564	GNU project Emacs, the Lisp based extensible text editor
w3m	V:10, I:129	2853	Text-mode WWW browsers
gpm	V:8.3, I:9.0	524	The Unix style cut-and-paste on the text console (daemon)

Table 1.1: List of interesting text-mode program packages

It may be a good idea to read some informative documentations.

You can install some of these packages by the following.

```
# apt-get install package_name
```

package	popcon	size	description
doc-debian	I:886	185	Debian Project documentation, (Debian FAQ) and other documents
debian-policy	I:6.8	5147	Debian Policy Manual and related documents
developers-reference	V:0.2, I:2.2	2657	Guidelines and information for Debian developers
debmake-doc	I:0.33	11803	Guide for Debian Maintainers
debian-history	I:0.50	6513	History of the Debian Project
debian-faq	I:885	791	Debian FAQ

Table 1.2: List of informative documentation packages

1.1.11 An extra user account

If you do not want to use your main user account for the following training activities, you can create a training user account, e.g. `fish` by the following.

```
# adduser fish
```

Answer all questions.

This creates a new account named as `fish`. After your practice, you can remove this user account and its home directory by the following.

```
# deluser --remove-home fish
```

On non-Debian and specialized Debian systems, above activities need to use lower level useradd(8) and userdel(8) utilities, instead.

1.1.12 sudo configuration

For the typical single user workstation such as the desktop Debian system on the laptop PC, it is common to deploy simple configuration of sudo(8) as follows to let the non-privileged user, e.g. `penguin`, to gain administrative privilege just with his user password but without the root password.

```
# echo "penguin  ALL=(ALL) ALL" >> /etc/sudoers
```

Alternatively, it is also common to do as follows to let the non-privileged user, e.g. `penguin`, to gain administrative privilege without any password.

```
# echo "penguin  ALL=(ALL) NOPASSWD:ALL" >> /etc/sudoers
```

This trick should only be used for the single user workstation which you administer and where you are the only user.



Warning
Do not set up accounts of regular users on multiuser workstation like this because it would be very bad for system security.



Caution
The password and the account of the `penguin` in the above example requires as much protection as the root password and the root account.
Administrative privilege in this context belongs to someone authorized to perform the system administration task on the workstation. Never give some manager in the Admin department of your company or your boss such privilege unless they are authorized and capable.

Note
For providing access privilege to limited devices and limited files, you should consider to use **group** to provide limited access instead of using the `root` privilege via sudo(8).
With more thoughtful and careful configuration, sudo(8) can grant limited administrative privileges to other users on a shared system without sharing the root password. This can help with accountability with hosts with multiple administrators so you can tell who did what. On the other hand, you might not want anyone else to have such privileges.

1.1.13 Play time

Now you are ready to play with the Debian system without risks as long as you use the non-privileged user account.

This is because the Debian system is, even after the default installation, configured with proper file permissions which prevent non-privileged users from damaging the system. Of course, there may still be some holes which can be exploited but those who worry about these issues should not be reading this section but should be reading Securing Debian Manual.

We learn the Debian system as a Unix-like system with the following.

- Section 1.2 (basic concept)
- Section 1.3 (survival method)
- Section 1.4 (basic method)
- Section 1.5 (shell mechanism)
- Section 1.6 (text processing method)

1.2 Unix-like filesystem

In GNU/Linux and other Unix-like operating systems, files are organized into directories. All files and directories are arranged in one big tree rooted at "`/`". It's called a tree because if you draw the filesystem, it looks like a tree but it is upside down.

These files and directories can be spread out over several devices. mount(8) serves to attach the filesystem found on some device to the big file tree. Conversely, umount(8) detaches it again. On recent Linux kernels, mount(8) with some options can bind part of a file tree somewhere else or can mount filesystem as shared, private, slave, or unbindable. Supported mount options for each filesystem are available in "`/usr/share/doc/linux-doc-*/Documentation/files`

Directories on Unix systems are called **folders** on some other systems. Please also note that there is no concept for **drive** such as "`A:`" on any Unix system. There is one filesystem, and everything is included. This is a huge advantage compared to Windows.

1.2.1 Unix file basics

Here are some Unix file basics.

- Filenames are **case sensitive**. That is, "`MYFILE`" and "`MyFile`" are different files.
- The **root directory** means root of the filesystem referred as simply "`/`". Don't confuse this with the home directory for the root user: "`/root`".
- Every directory has a name which can contain any letters or symbols **except "`/`"**. The root directory is an exception; its name is "`/`" (pronounced "slash" or "the root directory") and it cannot be renamed.

- Each file or directory is designated by a **fully-qualified filename**, **absolute filename**, or **path**, giving the sequence of directories which must be passed through to reach it. The three terms are synonymous.
- All **fully-qualified filenames** begin with the "/" directory, and there's a "/" between each directory or file in the filename. The first "/" is the top level directory, and the other "/"'s separate successive subdirectories, until we reach the last entry which is the name of the actual file. The words used here can be confusing. Take the following **fully-qualified filename** as an example: "`/usr/share/keytables/us.map.gz`". However, people also refers to its basename "`us.map.gz`" alone as a filename.
- The root directory has a number of branches, such as "`/etc/`" and "`/usr/`". These subdirectories in turn branch into still more subdirectories, such as "`/etc/systemd/`" and "`/usr/local/`". The whole thing viewed collectively is called the **directory tree**. You can think of an absolute filename as a route from the base of the tree ("/") to the end of some branch (a file). You also hear people talk about the directory tree as if it were a **family** tree encompassing all direct descendants of a single figure called the root directory ("/"): thus subdirectories have **parents**, and a path shows the complete ancestry of a file. There are also relative paths that begin somewhere other than the root directory. You should remember that the directory "`../`" refers to the parent directory. This terminology also applies to other directory like structures, such as hierarchical data structures.
- There's no special directory path name component that corresponds to a physical device, such as your hard disk. This differs from RT-11, CP/M, OpenVMS, MS-DOS, AmigaOS, and Microsoft Windows, where the path contains a device name such as "`C:\`". (However, directory entries do exist that refer to physical devices as a part of the normal filesystem. See Section 1.2.2.)

Note
While you **can** use almost any letters or symbols in a file name, in practice it is a bad idea to do so. It is better to avoid any characters that often have special meanings on the command line, including spaces, tabs, newlines, and other special characters: `{ } ( ) [ ] ' ` " \ / > < | ; ! # & ^ * % @ $`. If you want to separate words in a name, good choices are the period, hyphen, and underscore. You could also capitalize each word, "`LikeThis`". Experienced Linux users tend to avoid spaces in filenames.

Note
The word "root" can mean either "root user" or "root directory". The context of their usage should make it clear.

Note
The word **path** is used not only for **fully-qualified filename** as above but also for the **command search path**. The intended meaning is usually clear from the context.

The detailed best practices for the file hierarchy are described in the Filesystem Hierarchy Standard ("`/usr/share/doc/deb`
and hier(7)). You should remember the following facts as the starter.

directory	usage of the directory
`/`	the root directory
`/etc/`	system wide configuration files
`/var/log/`	system log files
`/home/`	all the home directories for all non-privileged users

Table 1.3: List of usage of key directories

1.2.2 Filesystem internals

Following the **Unix tradition**, the Debian GNU/Linux system provides the filesystem under which physical data on hard disks and other storage devices reside, and the interaction with the hardware devices such as console screens and remote serial consoles are represented in an unified manner under "`/dev/`".

Each file, directory, named pipe (a way two programs can share data), or physical device on a Debian GNU/Linux system has a data structure called an inode which describes its associated attributes such as the user who owns it (owner), the group that it belongs to, the time last accessed, etc. The idea of representing just about everything in the filesystem was a Unix innovation, and modern Linux kernels have developed this idea ever further. Now, even information about processes running in the computer can be found in the filesystem.

This abstract and unified representation of physical entities and internal processes is very powerful since this allows us to use the same command for the same kind of operation on many totally different devices. It is even possible to change the way the kernel works by writing data to special files that are linked to running processes.

Tip
If you need to identify the correspondence between the file tree and the physical entity, execute mount(8) with no arguments.

1.2.3 Filesystem permissions

Filesystem permissions of Unix-like system are defined for three categories of affected users.

- The **user** who owns the file (**u**)
- Other users in the **group** which the file belongs to (**g**)
- All **other** users (**o**) also referred to as "world" and "everyone"

For the file, each corresponding permission allows following actions.

- The **read** (**r**) permission allows owner to examine contents of the file.
- The **write** (**w**) permission allows owner to modify the file.
- The **execute** (**x**) permission allows owner to run the file as a command.

For the directory, each corresponding permission allows following actions.

- The **read** (**r**) permission allows owner to list contents of the directory.
- The **write** (**w**) permission allows owner to add or remove files in the directory.
- The **execute** (**x**) permission allows owner to access files in the directory.

Here, the **execute** permission on a directory means not only to allow reading of files in that directory but also to allow viewing their attributes, such as the size and the modification time.

ls(1) is used to display permission information (and more) for files and directories. When it is invoked with the "`-l`" option, it displays the following information in the order given.

- **Type of file** (first character)
- Access **permission** of the file (nine characters, consisting of three characters each for user, group, and other in this order)
- **Number of hard links** to the file

- Name of the **user** who owns the file
- Name of the **group** which the file belongs to
- **Size** of the file in characters (bytes)
- **Date and time** of the file (mtime)
- **Name** of the file

character	meaning
-	normal file
d	directory
l	symlink
c	character device node
b	block device node
p	named pipe
s	socket

Table 1.4: List of the first character of "`ls -l`" output

chown(1) is used from the root account to change the owner of the file. chgrp(1) is used from the file's owner or root account to change the group of the file. chmod(1) is used from the file's owner or root account to change file and directory access permissions. Basic syntax to manipulate a `foo` file is the following.

```
# chown newowner foo
# chgrp newgroup foo
# chmod  [ugoa][+-=][rwxXst][,...] foo
```

For example, you can make a directory tree to be owned by a user `foo` and shared by a group `bar` by the following.

```
# cd /some/location/
# chown -R foo:bar .
# chmod -R ug+rwX,o=rX .
```

There are three more special permission bits.

- The **set user ID** bit (**s** or **S** instead of user's **x**)
- The **set group ID** bit (**s** or **S** instead of group's **x**)
- The **sticky** bit (**t** or **T** instead of other's **x**)

Here the output of "`ls -l`" for these bits is **capitalized** if execution bits hidden by these outputs are **unset**.

Setting **set user ID** on an executable file allows a user to execute the executable file with the owner ID of the file (for example **root**). Similarly, setting **set group ID** on an executable file allows a user to execute the executable file with the group ID of the file (for example **root**). Because these settings can cause security risks, enabling them requires extra caution.

Setting **set group ID** on a directory enables the BSD-like file creation scheme where all files created in the directory belong to the **group** of the directory.

Setting the **sticky bit** on a directory prevents a file in the directory from being removed by a user who is not the owner of the file. In order to secure contents of a file in world-writable directories such as "`/tmp`" or in group-writable directories, one must not only reset the **write** permission for the file but also set the **sticky bit** on the directory. Otherwise, the file can be removed and a new file can be created with the same name by any user who has write access to the directory.

Here are a few interesting examples of file permissions.

```
$ ls -l /etc/passwd /etc/shadow /dev/ppp /usr/sbin/exim4
crw------T 1 root root   108, 0 Oct 16 20:57 /dev/ppp
-rw-r--r-- 1 root root     2761 Aug 30 10:38 /etc/passwd
-rw-r----- 1 root shadow   1695 Aug 30 10:38 /etc/shadow
-rwsr-xr-x 1 root root   973824 Sep 23 20:04 /usr/sbin/exim4
$ ls -ld /tmp /var/tmp /usr/local /var/mail /usr/src
drwxrwxrwt 14 root root  20480 Oct 16 21:25 /tmp
drwxrwsr-x 10 root staff  4096 Sep 29 22:50 /usr/local
drwxr-xr-x 10 root root   4096 Oct 11 00:28 /usr/src
drwxrwsr-x  2 root mail   4096 Oct 15 21:40 /var/mail
drwxrwxrwt  3 root root   4096 Oct 16 21:20 /var/tmp
```

There is an alternative numeric mode to describe file permissions with chmod(1). This numeric mode uses 3 to 4 digit wide octal (radix=8) numbers.

digit	meaning
1st optional digit	sum of **set user ID** (=4), **set group ID** (=2), and **sticky bit** (=1)
2nd digit	sum of **read** (=4), **write** (=2), and **execute** (=1) permissions for **user**
3rd digit	ditto for **group**
4th digit	ditto for **other**

Table 1.5: The numeric mode for file permissions in chmod(1) commands

This sounds complicated but it is actually quite simple. If you look at the first few (2-10) columns from "`ls -l`" command output and read it as a binary (radix=2) representation of file permissions ("-" being "0" and "rwx" being "1"), the last 3 digit of the numeric mode value should make sense as an octal (radix=8) representation of file permissions to you.

For example, try the following

```
$ touch foo bar
$ chmod u=rw,go=r foo
$ chmod 644 bar
$ ls -l foo bar
-rw-r--r-- 1 penguin penguin 0 Oct 16 21:39 bar
-rw-r--r-- 1 penguin penguin 0 Oct 16 21:35 foo
```

Tip

If you need to access information displayed by "`ls -l`" in shell script, you should use pertinent commands such as test(1), stat(1) and readlink(1). The shell builtin such as "`[`" or "`test`" may be used too.

1.2.4 Control of permissions for newly created files: umask

What permissions are applied to a newly created file or directory is restricted by the `umask` shell builtin command. See dash(1), bash(1), and builtins(7).

```
(file permissions) = (requested file permissions) & ~(umask value)
```

The Debian system uses a user private group (UPG) scheme as its default. A UPG is created whenever a new user is added to the system. A UPG has the same name as the user for which it was created and that user is the only member of the UPG. UPG scheme makes it safe to set umask to `0002` since every user has their own private group. (In some Unix variants, it is quite common to setup all normal users belonging to a single **`users`** group and is a good idea to set umask to `0022` for security in such cases.)

umask	file permissions created	directory permissions created	usage
0022	-rw-r--r--	-rwxr-xr-x	writable only by the user
0002	-rw-rw-r--	-rwxrwxr-x	writable by the group

Table 1.6: The **umask** value examples

Tip
Enable UPG by putting "`umask 002`" in the `~/.bashrc` file.

1.2.5 Permissions for groups of users (group)



Warning
Please make sure to save unsaved changes before doing reboot or similar actions.

You can add a user `penguin` to a group `bird` in two steps:

- Change group configuration using one of following:
  - Execute "`sudo usermod -aG bird penguin`".
  - Execute "`sudo adduser penguin bird`". (only on typical Debian systems)
  - Execute "`sudo vigr`" for `/etc/group` and "`sudo vigr -s`" for `/etc/gshadow` to append `penguin` in the line for `bird`.
- Apply configuration using one of following:
  - Cold reboot and login. (Best option)
  - Logout via GUI menu and login. (This may not work under the modern Desktop environment.)

You can remove a user `penguin` from a group `bird` in two steps:

- Change group configuration using one of following:
  - Execute "`sudo usermod -rG bird penguin`".
  - Execute "`sudo deluser penguin bird`". (only on typical Debian systems)
  - Execute "`sudo vigr`" for `/etc/group` and "`sudo vigr -s`" for `/etc/gshadow` to remove `penguin` in the line for `bird`.
- Apply configuration using one of following:
  - Cold reboot and login. (Best option)
  - Execute "`kill -TERM -1`" and do some fix-up actions such as "`systemctl restart NetworkManager.service`"
  - Logout via GUI menu is not an option for Gnome Desktop.

Any warm reboot attempts are fragile replacements of the real cold reboot under the modern desktop system.

Note
Alternatively, you may dynamically add users to groups during the authentication process by adding "`auth optional pam_group.so`" line to "`/etc/pam.d/common-auth`" and setting "`/etc/security/group.conf`". (See Chapter 4.)

The hardware devices are just another kind of file on the Debian system. If you have problems accessing devices such as USB flash drive and CD-ROM from a user account, you should make that user a member of the relevant group.

Some notable system-provided groups allow their members to access particular files and devices without `root` privilege.

group	description for accessible files and devices
`dialout`	full and direct access to serial ports ("`/dev/ttyS[0-3]`")
`dip`	limited access to serial ports for **Dialup IP** connection to trusted peers
`cdrom`	CD-ROM, DVD+/-RW drives
`audio`	audio device
`video`	video device
`scanner`	scanner(s)
`adm`	system monitoring logs
`staff`	some directories for junior administrative work: "`/usr/local`", "`/home`"

Table 1.7: List of notable system-provided groups for file access

Tip
You need to belong to the `dialout` group to reconfigure modem, dial anywhere, etc. But if `root` creates predefined configuration files for trusted peers in "`/etc/ppp/peers/`", you only need to belong to the `dip` group to create **Dialup IP** connection to those trusted peers using pppd(8), pon(1), and poff(1) commands.

Some notable system-provided groups allow their members to execute particular commands without `root` privilege.

group	accessible commands
`sudo`	execute any command with superuser privileges
`lpadmin`	execute commands to add, modify, and remove printers from printer databases

Table 1.8: List of notable system provided groups for particular command executions

For the full listing of the system provided users and groups, see the recent version of the "Users and Groups" document in "`/usr/share/doc/base-passwd/users-and-groups.html`" provided by the `base-passwd` package.

See passwd(5), group(5), shadow(5), newgrp(1), vipw(8), vigr(8), and pam_group(8) for management commands of the user and group system.

1.2.6 Timestamps

There are three types of timestamps for a GNU/Linux file.

type	meaning (historic Unix definition)
mtime	the file modification time (`ls -l`)
ctime	the file status change time (`ls -lc`)
atime	the last file access time (`ls -lu`)

Table 1.9: List of types of timestamps

Note
ctime is not file creation time.

Note
The actual value of **atime** on GNU/Linux system may be different from that of the historic Unix definition.

- Overwriting a file changes all of the **mtime**, **ctime**, and **atime** attributes of the file.
- Changing ownership or permission of a file changes the **ctime** and **atime** attributes of the file.
- Reading a file changes the **atime** attribute of the file on the historic Unix system.
- Reading a file changes the **atime** attribute of the file on the GNU/Linux system if its filesystem is mounted with "`strictatime`".
- Reading a file for the first time or after one day changes the **atime** attribute of the file on the GNU/Linux system if its filesystem is mounted with "`relatime`". (default behavior since Linux 2.6.30)
- Reading a file doesn't change the **atime** attribute of the file on the GNU/Linux system if its filesystem is mounted with "`noatime`".

Note
The "`noatime`" and "`relatime`" mount options are introduced to improve the filesystem read performance under the normal use cases. Simple file read operation under the "`strictatime`" option accompanies the time-consuming write operation to update the **atime** attribute. But the **atime** attribute is rarely used except for the mbox(5) file. See mount(8).

Use touch(1) command to change timestamps of existing files.

For timestamps, the `ls` command outputs localized strings under non-English locale ("`fr_FR.UTF-8`").

```
$ LANG=C  ls -l foo
-rw-rw-r-- 1 penguin penguin 0 Oct 16 21:35 foo
$ LANG=en_US.UTF-8  ls -l foo
-rw-rw-r-- 1 penguin penguin 0 Oct 16 21:35 foo
$ LANG=fr_FR.UTF-8  ls -l foo
-rw-rw-r-- 1 penguin penguin 0 oct. 16 21:35 foo
```

Tip
See Section 9.3.4 to customize "`ls -l`" output.

1.2.7 Links

There are two methods of associating a file "`foo`" with a different filename "`bar`".

- Hard link
  - Duplicate name for an existing file
  - "`ln foo bar`"
- Symbolic link or symlink
  - Special file that points to another file by name
  - "`ln -s foo bar`"

See the following example for changes in link counts and the subtle differences in the result of the `rm` command.

```
$ umask 002
$ echo "Original Content" > foo
$ ls -li foo
1449840 -rw-rw-r-- 1 penguin penguin 17 Oct 16 21:42 foo
$ ln foo bar     # hard link
$ ln -s foo baz  # symlink
$ ls -li foo bar baz
1449840 -rw-rw-r-- 2 penguin penguin 17 Oct 16 21:42 bar
1450180 lrwxrwxrwx 1 penguin penguin  3 Oct 16 21:47 baz -> foo
1449840 -rw-rw-r-- 2 penguin penguin 17 Oct 16 21:42 foo
$ rm foo
$ echo "New Content" > foo
$ ls -li foo bar baz
1449840 -rw-rw-r-- 1 penguin penguin 17 Oct 16 21:42 bar
1450180 lrwxrwxrwx 1 penguin penguin  3 Oct 16 21:47 baz -> foo
1450183 -rw-rw-r-- 1 penguin penguin 12 Oct 16 21:48 foo
$ cat bar
Original Content
$ cat baz
New Content
```

The hardlink can be made within the same filesystem and shares the same inode number which the "`-i`" option with ls(1) reveals.

The symlink always has nominal file access permissions of "`rwxrwxrwx`", as shown in the above example, with the effective access permissions dictated by permissions of the file that it points to.



Caution
It is generally a good idea not to create complicated symbolic links or hardlinks at all unless you have a very good reason. It may cause nightmares where the logical combination of the symbolic links results in loops in the filesystem.

Note
It is generally preferable to use symbolic links rather than hardlinks unless you have a good reason for using a hardlink.

The "`.`" directory links to the directory that it appears in, thus the link count of any new directory starts at 2. The "`..`" directory links to the parent directory, thus the link count of the directory increases with the addition of new subdirectories.

If you are just moving to Linux from Windows, it soon becomes clear how well-designed the filename linking of Unix is, compared with the nearest Windows equivalent of "shortcuts". Because it is implemented in the filesystem, applications can't see any difference between a linked file and the original. In the case of hardlinks, there really is no difference.

1.2.8 Named pipes (FIFOs)

A named pipe is a file that acts like a pipe. You put something into the file, and it comes out the other end. Thus it's called a FIFO, or First-In-First-Out: the first thing you put in the pipe is the first thing to come out the other end.

If you write to a named pipe, the process which is writing to the pipe doesn't terminate until the information being written is read from the pipe. If you read from a named pipe, the reading process waits until there is nothing to read before terminating. The size of the pipe is always zero --- it does not store data, it just links two processes like the functionality offered by the shell "`|`" syntax. However, since this pipe has a name, the two processes don't have to be on the same command line or even be run by the same user. Pipes were a very influential innovation of Unix.

For example, try the following

```
$ cd; mkfifo mypipe
$ echo "hello" >mypipe & # put into background
[1] 8022
$ ls -l mypipe
prw-rw-r-- 1 penguin penguin 0 Oct 16 21:49 mypipe
$ cat mypipe
hello
[1]+  Done                    echo "hello" >mypipe
$ ls mypipe
mypipe
$ rm mypipe
```

1.2.9 Sockets

Sockets are used extensively by all the Internet communication, databases, and the operating system itself. It is similar to the named pipe (FIFO) and allows processes to exchange information even between different computers. For the socket, those processes do not need to be running at the same time nor to be running as the children of the same ancestor process. This is the endpoint for the inter process communication (IPC). The exchange of information may occur over the network between different hosts. The two most common ones are the Internet socket and the Unix domain socket.

Tip
"`ss -t -u -a`" (from the `iproute2` package) provides a very useful overview of sockets that are open on a given system.

1.2.10 Device files

Device files refer to physical or virtual devices on your system, such as your hard disk, video card, screen, or keyboard. An example of a virtual device is the console, represented by "`/dev/console`".

There are 2 types of device files.

- **Character device**
  - Accessed one character at a time
  - 1 character = 1 byte
  - E.g. keyboard device, serial port, …
- **Block device**
  - accessed in larger units called blocks
  - 1 block > 1 byte
  - E.g. hard disk, …

You can read and write device files, though the file may well contain binary data which may be an incomprehensible-to-humans gibberish. Writing data directly to these files is sometimes useful for the troubleshooting of hardware connections. For example, you can dump a text file to the printer device "`/dev/lp0`" or send modem commands to the appropriate serial port "`/dev/ttyS0`". But, unless this is done carefully, it may cause a major disaster. So be cautious.

Note
For the normal access to a printer, use lp(1).

The device node number are displayed by executing ls(1) as the following.

```
$ ls -l /dev/sda /dev/sr0 /dev/ttyS0 /dev/zero
brw-rw---T  1 root disk     8,  0 Oct 16 20:57 /dev/sda
brw-rw---T+ 1 root cdrom   11,  0 Oct 16 21:53 /dev/sr0
crw-rw---T  1 root dialout  4, 64 Oct 16 20:57 /dev/ttyS0
crw-rw-rw-  1 root root     1,  5 Oct 16 20:57 /dev/zero
```

- "`/dev/sda`" has the major device number 8 and the minor device number 0. This is read/write accessible by users belonging to the `disk` group.
- "`/dev/sr0`" has the major device number 11 and the minor device number 0. This is read/write accessible by users belonging to the `cdrom` group.
- "`/dev/ttyS0`" has the major device number 4 and the minor device number 64. This is read/write accessible by users belonging to the `dialout` group.
- "`/dev/zero`" has the major device number 1 and the minor device number 5. This is read/write accessible by anyone.

On the modern Linux system, the filesystem under "`/dev/`" is automatically populated by the udev(7) mechanism.

1.2.11 Special device files

There are some special device files.

device file	action	description of response
`/dev/null`	read	return "end-of-file (EOF) character"
`/dev/null`	write	return nothing (a bottomless data dump pit)
`/dev/zero`	read	return "the `\0` (NUL) character" (not the same as the number zero ASCII)
`/dev/random`	read	return random characters from a true random number generator, delivering real entropy (slow)
`/dev/urandom`	read	return random characters from a cryptographically secure pseudorandom number generator
`/dev/full`	write	return the disk-full (ENOSPC) error

Table 1.10: List of special device files

These are frequently used in conjunction with the shell redirection (see Section 1.5.8).

1.2.12 procfs and sysfs

The procfs and sysfs mounted on "`/proc`" and "`/sys`" are the pseudo-filesystem and expose internal data structures of the kernel to the userspace. In other word, these entries are virtual, meaning that they act as a convenient window into the operation of the operating system.

The directory "`/proc`" contains (among other things) one subdirectory for each process running on the system, which is named after the process ID (PID). System utilities that access process information, such as ps(1), get their information from this directory structure.

The directories under "`/proc/sys/`" contain interfaces to change certain kernel parameters at run time. (You may do the same through the specialized sysctl(8) command or its preload/configuration file "`/etc/sysctl.d/*.conf`".)

People frequently panic when they notice one file in particular - "/proc/kcore" - which is generally huge. This is (more or less) a copy of the content of your computer's memory. It's used to debug the kernel. It is a virtual file that points to computer memory, so don't worry about its size.

The directory under "/sys" contains exported kernel data structures, their attributes, and their linkages between them. It also contains interfaces to change certain kernel parameters at run time.

See "proc.txt(.gz)", "sysfs.txt(.gz)" and other related documents in the Linux kernel documentation ("/usr/share provided by the linux-doc-* package.

1.2.13 tmpfs

The tmpfs is a temporary filesystem which keeps all files in the virtual memory. The data of the tmpfs in the page cache on memory may be swapped out to the swap space on disk as needed.

The directory "/run" is mounted as the tmpfs in the early boot process. This enables writing to it even when the directory "/" is mounted as read-only. This is the new location for the storage of transient state files and replaces several locations described in the Filesystem Hierarchy Standard version 2.3:

- "/var/run" → "/run"
- "/var/lock" → "/run/lock"
- "/dev/shm" → "/run/shm"

See "tmpfs.txt(.gz)" in the Linux kernel documentation ("/usr/share/doc/linux-doc-*/Documentation/files provided by the linux-doc-* package.

1.3 Midnight Commander (MC)

Midnight Commander (MC) is a GNU "Swiss army knife" for the Linux console and other terminal environments. This gives newbie a menu driven console experience which is much easier to learn than standard Unix commands.

You may need to install the Midnight Commander package which is titled "mc" by the following.

```
$ sudo apt-get install mc
```

Use the mc(1) command to explore the Debian system. This is the best way to learn. Please explore few interesting locations just using the cursor keys and Enter key.

- "/etc" and its subdirectories
- "/var/log" and its subdirectories
- "/usr/share/doc" and its subdirectories
- "/usr/sbin" and "/usr/bin"

1.3.1 Customization of MC

In order to make MC to change working directory upon exit and cd to the directory, I suggest to modify "~/.bashrc" to include a script provided by the mc package.

```
. /usr/lib/mc/mc.sh
```

See mc(1) (under the "-P" option) for the reason. (If you do not understand what exactly I am talking here, you can do this later.)

1.3.2 Starting MC

MC can be started by the following.

```
$ mc
```

MC takes care of all file operations through its menu, requiring minimal user effort. Just press F1 to get the help screen. You can play with MC just by pressing cursor-keys and function-keys.

Note
In some consoles such as gnome-terminal(1), key strokes of function-keys may be stolen by the console program. You can disable these features in "Preferences" → "General" and "Shortcuts" menu for `gnome-terminal`.

If you encounter character encoding problem which displays garbage characters, adding "`-a`" to MC's command line may help prevent problems.

If this doesn't clear up your display problems with MC, see Section 8.3.1.

1.3.3 File manager in MC

The default is two directory panels containing file lists. Another useful mode is to set the right window to "information" to see file access privilege information, etc. Following are some essential keystrokes. With the gpm(8) daemon running, one can use a mouse on Linux character consoles, too. (Make sure to press the shift-key to obtain the normal behavior of cut and paste in MC.)

key	key binding
`F1`	help menu
`F3`	internal file viewer
`F4`	internal editor
`F9`	activate pull down menu
`F10`	exit Midnight Commander
`Tab`	move between two windows
`Insert` or `Ctrl-T`	mark file for a multiple-file operation such as copy
`Del`	delete file (be careful---set MC to safe delete mode)
Cursor keys	self-explanatory

Table 1.11: The key bindings of MC

1.3.4 Command-line tricks in MC

- `cd` command changes the directory shown on the selected screen.
- `Ctrl-Enter` or `Alt-Enter` copies a filename to the command line. Use this with cp(1) and mv(1) commands together with command-line editing.
- `Alt-Tab` shows shell filename expansion choices.
- One can specify the starting directory for both windows as arguments to MC; for example, "`mc /etc /root`".
- `Esc` + `n-key` → `Fn` (i.e., `Esc` + `1` → `F1`, etc.; `Esc` + `0` → `F10`)
- Pressing `Esc` before the key has the same effect as pressing the `Alt` and the key together.; i.e., type `Esc` + `c` for `Alt-C`. `Esc` is called meta-key and sometimes noted as "`M-`".

1.3.5 The internal editor in MC

The internal editor has an interesting cut-and-paste scheme. Pressing F3 marks the start of a selection, a second F3 marks the end of selection and highlights the selection. Then you can move your cursor. If you press F6, the selected area is moved to the cursor location. If you press F5, the selected area is copied and inserted at the cursor location. F2 saves the file. F10 gets you out. Most cursor keys work intuitively.

This editor can be directly started on a file using one of the following commands.

```
$ mc -e filename_to_edit
```

```
$ mcedit filename_to_edit
```

This is not a multi-window editor, but one can use multiple Linux consoles to achieve the same effect. To copy between windows, use Alt-F*n* keys to switch virtual consoles and use "File→Insert file" or "File→Copy to file" to move a portion of a file to another file.

This internal editor can be replaced with any external editor of choice.

Also, many programs use the environment variables "`$EDITOR`" or "`$VISUAL`" to decide which editor to use. If you are uncomfortable with vim(1) or nano(1) initially, you may set these to "`mcedit`" by adding the following lines to "`~/.bashrc`".

```
export EDITOR=mcedit
export VISUAL=mcedit
```

I do recommend setting these to "`vim`" if possible.

If you are uncomfortable with vim(1), you can keep using mcedit(1) for most system maintenance tasks.

1.3.6 The internal viewer in MC

MC is a very smart viewer. This is a great tool for searching words in documents. I always use this for files in the "`/usr/share/doc`" directory. This is the fastest way to browse through masses of Linux information. This viewer can be directly started using one of the following commands.

```
$ mc -v path/to/filename_to_view
```

```
$ mcview path/to/filename_to_view
```

1.3.7 Auto-start features of MC

Press Enter on a file, and the appropriate program handles the content of the file (see Section 9.4.11). This is a very convenient MC feature.

file type	**reaction to enter key**
executable file	execute command
man file	pipe content to viewer software
html file	pipe content to web browser
"`*.tar.gz`" and "`*.deb`" file	browse its contents as if subdirectory

Table 1.12: The reaction to the enter key in MC

In order to allow these viewer and virtual file features to function, viewable files should not be set as executable. Change their status using chmod(1) or via the MC file menu.

1.3.8 Virtual filesystem of MC

MC can be used to access files over the Internet. Go to the menu by pressing `F9`, "`Enter`" and "h" to activate the Shell filesystem. Enter a URL in the form "`sh://[user@]machine[:options]/[remote-dir]`", which retrieves a remote directory that appears like a local one using `ssh`.

1.4 The basic Unix-like work environment

Although MC enables you to do almost everything, it is very important for you to learn how to use the command line tools invoked from the shell prompt and become familiar with the Unix-like work environment.

1.4.1 The login shell

Since the login shell may be used by some system initialization programs, it is prudent to keep it as bash(1) and avoid switching the login shell with chsh(1).

If you want to use a different interactive shell prompt, set it from GUI terminal emulator configuration or start it from `~/.bashrc`, e.g., by placing "`exec /usr/bin/zsh -i -l`" or "`exec /usr/bin/fish -i -l`" in it.

package	popcon	size	POSIX shell	description
`bash`	V:886, I:999	7308	Yes	Bash: the GNU Bourne Again SHell (de facto standard)
`bash-completion`	V:37, I:962	1952	N/A	programmable completion for the bash shell
`dash`	V:929, I:999	207	Yes	Debian Almquist Shell, good for shell script
`zsh`	V:39, I:68	2571	Yes	Z shell: the standard shell with many enhancements
`tcsh`	V:4, I:14	1366	No	TENEX C Shell: an enhanced version of Berkeley csh
`mksh`	V:4.7, I:7.5	7713	Yes	A version of the Korn shell
`csh`	V:1.0, I:5.2	348	No	OpenBSD C Shell, a version of Berkeley csh
`sash`	V:0.3, I:4.7	1245	Yes	Stand-alone shell with builtin commands (Not meant for standard "`/usr/bin/sh`")
`ksh`	V:0.3, I:7.9	65	Yes	the real, AT&T version of the Korn shell
`rc`	V:0.07, I:0.70	182	No	implementation of the AT&T Plan 9 rc shell
`posh`	V:0.02, I:0.22	195	Yes	Policy-compliant Ordinary SHell (`pdksh` derivative)

Table 1.13: List of shell programs

Tip
Although POSIX-like shells share the basic syntax, they can differ in behavior for things as basic as shell variables and glob expansions. Please check their documentation for details.

In this tutorial chapter, the interactive shell always means `bash`.

1.4.2 Customizing bash

You can customize bash(1) behavior by "`~/.bashrc`".

For example, try the following.

```
# enable bash-completion
if ! shopt -oq posix; then
  if [ -f /usr/share/bash-completion/bash_completion ]; then
    . /usr/share/bash-completion/bash_completion
  elif [ -f /etc/bash_completion ]; then
    . /etc/bash_completion
  fi
fi

# CD upon exiting MC
. /usr/lib/mc/mc.sh

# set CDPATH to a good one
CDPATH=.:/usr/share/doc:~:~/Desktop:~
export CDPATH

PATH="${PATH+$PATH:}/usr/sbin:/sbin"
# set PATH so it includes user's private bin if it exists
if [ -d ~/bin ] ; then
  PATH="~/bin${PATH+:$PATH}"
fi
export PATH

EDITOR=vim
export EDITOR
```

Tip
You can find more `bash` customization tips, such as Section 9.3.6, in Chapter 9.

Tip
The `bash-completion` package enables programmable completion for `bash`.

1.4.3 Special key strokes

In the Unix-like environment, there are few key strokes which have special meanings. Please note that on a normal Linux character console, only the left-hand `Ctrl` and `Alt` keys work as expected. Here are few notable key strokes to remember.

Tip
The terminal feature of `Ctrl-S` can be disabled using stty(1).

1.4.4 Mouse operations

Mouse operations for text on Debian system mix 2 styles with some twists:

- Traditional Unix style mouse operations:

key	description of key binding
Ctrl-U	erase line before cursor
Ctrl-H	erase a character before cursor
Ctrl-D	terminate input (exit shell if you are using shell)
Ctrl-C	terminate a running program
Ctrl-Z	temporarily stop program by moving it to the background job
Ctrl-S	halt output to screen
Ctrl-Q	reactivate output to screen
Ctrl-Alt-Del	reboot/halt the system, see inittab(5)
Left-Alt-key (optionally, Windows-key)	meta-key for Emacs and the similar UI
Up-arrow	start command history search under bash
Ctrl-R	start incremental command history search under bash
Tab	complete input of the filename to the command line under bash
Ctrl-V Tab	input Tab without expansion to the command line under bash

Table 1.14: List of key bindings for bash

- use 3 buttons (click)
- use PRIMARY
- used by X applications such as `xterm` and text applications in Linux console

- Modern GUI style mouse operations:
  - use 2 buttons (drag + click)
  - use PRIMARY and CLIPBOARD
  - used in Modern GUI applications such as `gnome-terminal`

action	response
Left-click-and-drag mouse	select range as PRIMARY selection
Left-click	select the start of range for PRIMARY selection
Right-click (traditional)	select the end of range for PRIMARY selection
Right-click (modern)	context dependent menu (cut/copy/paste)
Middle-click or Shift-Ins	insert PRIMARY selection at the cursor
Ctrl-X	cut PRIMARY selection to CLIPBOARD
Ctrl-C (Shift-Ctrl-C in terminal)	copy PRIMARY selection to CLIPBOARD
Ctrl-V	paste CLIPBOARD at the cursor

Table 1.15: List of mouse operations and related key actions on Debian

Here, the PRIMARY selection is the highlighted text range. Within the terminal program, `Shift-Ctrl-C` is used instead to avoid terminating a running program.

The center wheel on the modern wheel mouse is considered middle mouse button and can be used for middle-click. Clicking left and right mouse buttons together serves as the middle-click under the 2 button mouse system situation.

In order to use a mouse in Linux character consoles, you need to have gpm(8) running as daemon.

1.4.5 The pager

The less(1) command is the enhanced pager (file content browser). It reads the file specified by its command argument or its standard input. Hit "h" if you need help while browsing with the `less` command. It can do much more than more(1) and can be supercharged by executing "`eval $(lesspipe)`" or "`eval $(lessfile)`" in the shell startup script. See more in "`/usr/share/doc/less/LESSOPEN`". The "`-R`" option allows raw character output and enables ANSI color escape sequences. See less(1).

Tip
In the `less` command, type "h" to see the help screen, type "`/`" or "?" to search a string, and type "`-i`" to the change case sensitivity.

1.4.6 The text editor

You should become proficient in one of variants of Vim or Emacs programs which are popular in the Unix-like system.

I think getting used to Vim commands is the right thing to do, since Vi-editor is always there in the Linux/Unix world. (Actually, original `vi` or new `nvi` are programs you find everywhere. I choose Vim instead for newbie since it offers you help through `F1` key while it is similar enough and more powerful.)

If you choose either Emacs or XEmacs instead as your choice of the editor, that is another good choice indeed, particularly for programming. Emacs has a plethora of other features as well, including functioning as a newsreader, directory editor, mail program, etc. When used for programming or editing shell scripts, it intelligently recognizes the format of what you are working on, and tries to provide assistance. Some people maintain that the only program they need on Linux is Emacs. Ten minutes learning Emacs now can save hours later. Having the GNU Emacs manual for reference when learning Emacs is highly recommended.

All these programs usually come with tutoring program for you to learn them by practice. Start Vim by typing "`vim`" and press F1-key. You should at least read the first 35 lines. Then do the online training course by moving cursor to "`|tutor|`" and pressing `Ctrl-]`.

Note
Good editors, such as Vim and Emacs, can handle UTF-8 and other exotic encoding texts correctly. It is a good idea to use the GUI environment in the UTF-8 locale and to install required programs and fonts to it. Editors have options to set the file encoding independent of the GUI environment. Please refer to their documentation on multibyte text.

1.4.7 Setting a default text editor

Debian comes with a number of different editors. We recommend to install the `vim` package, as mentioned above.

Debian provides unified access to the system default editor via command "`/usr/bin/editor`" so other programs (e.g., reportbug(1)) can invoke it. You can change it by the following.

```
$ sudo update-alternatives --config editor
```

The choice "`/usr/bin/vim.basic`" over "`/usr/bin/vim.tiny`" is my recommendation for newbies since it supports syntax highlighting.

Tip
Many programs use the environment variables "`$EDITOR`" or "`$VISUAL`" to decide which editor to use (see Section 1.3.5 and Section 9.4.11). For the consistency on the Debian system, set these to "`/usr/bin/editor`". (Historically, "`$EDITOR`" was "`ed`" and "`$VISUAL`" was "`vi`".)

1.4.8 Using vim

The recent vim(1) starts itself in the sane "`nocompatible`" option and enters into the `NORMAL` mode.[1]

Please use the "`vimtutor`" program to learn `vim` through an interactive tutorial course.

[1] Even the older `vim` can starts in the sane "`nocompatible`" mode by starting it with the "`-N`" option.

mode	key strokes	action
NORMAL	:help|only	display the help file
NORMAL	:e filename.ext	open new buffer to edit filename.ext
NORMAL	:w	overwrite current buffer to the original file
NORMAL	:w filename.ext	write current buffer to filename.ext
NORMAL	:q	quit vim
NORMAL	:q!	force to quit vim
NORMAL	:only	close all other split open windows
NORMAL	:set nocompatible?	check if vim is in the sane nocompatible mode
NORMAL	:set nocompatible	set vim to the sane nocompatible mode
NORMAL	i	enter the INSERT mode
NORMAL	R	enter the REPLACE mode
NORMAL	v	enter the VISUAL mode
NORMAL	V	enter the linewise VISUAL mode
NORMAL	Ctrl-V	enter the blockwise VISUAL mode
except TERMINAL-JOB	ESC-key	enter the NORMAL mode
NORMAL	:term	enter the TERMINAL-JOB mode
TERMINAL-NORMAL	i	enter the TERMINAL-JOB mode
TERMINAL-JOB	Ctrl-W N (or Ctrl-\ Ctrl-N)	enter the TERMINAL-NORMAL mode
TERMINAL-JOB	Ctrl-W :	enter the Ex-mode in TERMINAL-NORMAL mode

Table 1.16: List of basic Vim key strokes

The vim program changes its behavior to typed key strokes based on **mode**. Typing in key strokes to the buffer is mostly done in INSERT-mode and REPLACE-mode. Moving cursor is mostly done in NORMAL-mode. Interactive selection is done in VISUAL-mode. Typing ":" in NORMAL-mode changes its **mode** to Ex-mode. Ex-mode accepts commands.

Tip
The Vim comes with the **Netrw** package. Netrw supports reading files, writing files, browsing directories over a network, and local browsing! Try Netrw with "vim ." (a period as the argument) and read its manual at ":help netrw".

For the advanced configuration of vim, see Section 9.2.

1.4.9 Recording the shell activities

The output of the shell command may roll off your screen and may be lost forever. It is a good practice to log shell activities into the file for you to review them later. This kind of record is essential when you perform any system administration tasks.

Tip
The new Vim (version>=8.2) can be used to record the shell activities cleanly using TERMINAL-JOB-mode. See Section 1.4.8.

The basic method of recording the shell activity is to run it under script(1).

For example, try the following

```
$ script
Script started, file is typescript
```

Do whatever shell commands under `script`.

Press `Ctrl-D` to exit `script`.

```
$ vim typescript
```

See Section 9.1.1 .

1.4.10 Basic Unix commands

Let's learn basic Unix commands. Here I use "Unix" in its generic sense. Any Unix clone OSs usually offer equivalent commands. The Debian system is no exception. Do not worry if some commands do not work as you wish now. If `alias` is used in the shell, its corresponding command outputs are different. These examples are not meant to be executed in this order.

Try all following commands from the non-privileged user account.

Note
Unix has a tradition to hide filenames which start with "`.`". They are traditionally files that contain configuration information and user preferences.
For `cd` command, see builtins(7).
The default pager of the bare bone Debian system is more(1) which cannot scroll back. By installing the `less` package using command line "`apt-get install less`", less(1) becomes default pager and you can scroll back with cursor keys.
The "`[`" and "`]`" in the regular expression of the "`ps aux | grep -e "[e]xim4*"`" command above enable `grep` to avoid matching itself. The "`4*`" in the regular expression means 0 or more repeats of character "4" thus enables `grep` to match both "`exim`" and "`exim4`". Although "`*`" is used in the shell filename glob and the regular expression, their meanings are different. Learn the regular expression from grep(1).

Please traverse directories and peek into the system using the above commands as training. If you have questions on any of console commands, please make sure to read the manual page.

For example, try the following

```
$ man man
$ man bash
$ man builtins
$ man grep
$ man ls
```

The style of man pages may be a little hard to get used to, because they are rather terse, particularly the older, very traditional ones. But once you get used to it, you come to appreciate their succinctness.

Please note that many Unix-like commands including ones from GNU and BSD display brief help information if you invoke them in one of the following ways (or without any arguments in some cases).

```
$ commandname --help
$ commandname -h
```

1.5 The simple shell command

Now you have some feel on how to use the Debian system. Let's look deep into the mechanism of the command execution in the Debian system. Here, I have simplified reality for the newbie. See bash(1) for the exact explanation.

A simple command is a sequence of components.

command	description
`pwd`	display name of current/working directory
`whoami`	display current user name
`id`	display current user identity (name, uid, gid, and associated groups)
`file` *foo*	display a type of file for the file "*foo*"
`type -p` *commandname*	display a file location of command "*commandname*"
`which` *commandname*	, ,
`type` *commandname*	display information on command "*commandname*"
`apropos` *key-word*	find commands related to "*key-word*"
`man -k` *key-word*	, ,
`whatis` *commandname*	display one line explanation on command "*commandname*"
`man -a` *commandname*	display explanation on command "*commandname*" (Unix style)
`info` *commandname*	display rather long explanation on command "*commandname*" (GNU style)
`ls`	list contents of directory (non-dot files and directories)
`ls -a`	list contents of directory (all files and directories)
`ls -A`	list contents of directory (almost all files and directories, i.e., skip "`..`" and "`.`")
`ls -la`	list all contents of directory with detail information
`ls -lai`	list all contents of directory with inode number and detail information
`ls -d`	list all directories under the current directory
`tree`	display file tree contents
`lsof` *foo*	list open status of file "*foo*"
`lsof -p` *pid*	list files opened by the process ID: "*pid*"
`mkdir` *foo*	make a new directory "*foo*" in the current directory
`rmdir` *foo*	remove a directory "*foo*" in the current directory
`cd` *foo*	change directory to the directory "*foo*" in the current directory or in the directory listed in the variable "`$CDPATH`"
`cd /`	change directory to the root directory
`cd`	change directory to the current user's home directory
`cd` */foo*	change directory to the absolute path directory "*/foo*"
`cd ..`	change directory to the parent directory
`cd ~`*foo*	change directory to the home directory of the user "*foo*"
`cd -`	change directory to the previous directory
`</etc/motd pager`	display contents of "`/etc/motd`" using the default pager
`touch` *junkfile*	create a empty file "*junkfile*"
`cp` *foo bar*	copy a existing file "*foo*" to a new file "*bar*"
`rm` *junkfile*	remove a file "*junkfile*"
`mv` *foo bar*	rename an existing file "*foo*" to a new name "*bar*" ("*bar*" must not exist)
`mv` *foo bar*	move an existing file "*foo*" to a new location "*bar/foo*" (the directory "*bar*" must exist)
`mv` *foo bar/baz*	move an existing file "*foo*" to a new location with a new name "*bar/baz*" (the directory "*bar*" must exist but the directory "*bar/baz*" must not exist)
`chmod 600` *foo*	make an existing file "*foo*" to be non-readable and non-writable by the other people (non-executable for all)
`chmod 644` *foo*	make an existing file "*foo*" to be readable but non-writable by the other people (non-executable for all)
`chmod 755` *foo*	make an existing file "*foo*" to be readable but non-writable by the other people (executable for all)
`find . -name` *pattern*	find matching filenames using shell "*pattern*" (slower)
`locate -d . pattern`	find matching filenames using shell "*pattern*" (quicker using regularly generated database)
`grep -e "`*pattern*`" *.html`	find a "*pattern*" in all files ending with "`.html`" in current directory and display them all
`top`	display process information using full screen, type "q" to quit
`ps aux | pager`	display information on all the running processes using BSD style output
`ps -ef | pager`	display information on all the running processes using Unix system-V style output
`ps aux | grep -e "[e]xim4*"`	display all processes running "`exim`" and "`exim4`"

1. Variable assignments (optional)
2. Command name
3. Arguments (optional)
4. Redirections (optional: > , >> , < , << , etc.)
5. Control operator (optional: && , || , *newline* , ; , & , (,))

1.5.1 Command execution and environment variable

The values of some environment variables change the behavior of some Unix commands.

Default values of environment variables are initially set by the PAM system and then some of them may be reset by some application programs.

- The PAM system such as `pam_env` may set environment variables by "`/etc/pam.conf`", "`/etc/environment`" and "`/etc/default/locale`".
- The modern GUI environment run from the user mode systemd unit (see systemd.unit(5)) sets its environment variables by a file in the "`~/.config/environment.d/`" directory.
- The user specific program initialization may reset environment variables by "`~/.profile`", "`~/.bash_profile`" and "`~/.bashrc`".

1.5.2 The "$LANG" variable

The default locale is defined in the "`$LANG`" environment variable and is configured as "`LANG=xx_YY.UTF-8`" by the installer or by the subsequent GUI configuration, e.g., "Settings" → "Region & Language" → "Language" / "Formats" for GNOME.

Note
I recommend you to configure the system environment just by the "`$LANG`" variable for now and to stay away from "`$LC_*`" variables unless it is absolutely needed.

The full locale value given to "`$LANG`" variable consists of 3 parts: "`xx_YY.ZZZZ`".

locale value	meaning
`xx`	ISO 639 language codes (lower case) such as "en"
`YY`	ISO 3166 country codes (upper case) such as "US"
`ZZZZ`	codeset, always set to "UTF-8"

Table 1.18: The 3 parts of locale value

Typical command execution uses a shell line sequence as the following.

```
$ echo $LANG
en_US.UTF-8
$ date -u
Wed 19 May 2021 03:18:43 PM UTC
$ LANG=fr_FR.UTF-8 date -u
mer. 19 mai 2021 15:19:02 UTC
```

Here, the program date(1) is executed with different locale values.

locale recommendation	Language (area)
en_US.UTF-8	English (USA)
en_GB.UTF-8	English (Great Britain)
fr_FR.UTF-8	French (France)
de_DE.UTF-8	German (Germany)
it_IT.UTF-8	Italian (Italy)
es_ES.UTF-8	Spanish (Spain)
ca_ES.UTF-8	Catalan (Spain)
sv_SE.UTF-8	Swedish (Sweden)
pt_BR.UTF-8	Portuguese (Brazil)
ru_RU.UTF-8	Russian (Russia)
zh_CN.UTF-8	Chinese (P.R. of China)
zh_TW.UTF-8	Chinese (Taiwan R.O.C.)
ja_JP.UTF-8	Japanese (Japan)
ko_KR.UTF-8	Korean (Republic of Korea)
vi_VN.UTF-8	Vietnamese (Vietnam)

Table 1.19: List of locale recommendations

- For the first command, "$LANG" is set to the system default locale value "en_US.UTF-8".
- For the second command, "$LANG" is set to the French UTF-8 locale value "fr_FR.UTF-8".

Most command executions usually do not have preceding environment variable definition. For the above example, you can alternatively execute as the following.

```
$ LANG=fr_FR.UTF-8
$ date -u
mer. 19 mai 2021 15:19:24 UTC
```

Tip
When filing a bug report, running and checking the command under "en_US.UTF-8" locale is a good idea if you use non-English environment.

For fine details of the locale configuration, see Section 8.1.

1.5.3 The "$PATH" variable

When you type a command into the shell, the shell searches the command in the list of directories contained in the "$PATH" environment variable. The value of the "$PATH" environment variable is also called the shell's search path.

In the default Debian installation, the "$PATH" environment variable of user accounts may not include "/usr/sbin" and "/usr/sbin". For example, the ifconfig command needs to be issued with full path as "/usr/sbin/ifconfig". (Similar ip command is located in "/usr/bin".)

You can change the "$PATH" environment variable of Bash shell by "~/.bash_profile" or "~/.bashrc" files.

1.5.4 The "$HOME" variable

Many commands stores user specific configuration in the home directory and changes their behavior by their contents. The home directory is identified by the environment variable "$HOME".

value of "$HOME"	program execution situation
/	program run by the init process (daemon)
/root	program run from the normal root shell
/home/*normal_user*	program run from the normal user shell
/home/*normal_user*	program run from the normal user GUI desktop menu
/home/*normal_user*	program run as root with "sudo program"
/root	program run as root with "sudo -H program"

Table 1.20: List of "$HOME" values

Tip
Shell expands "~/" to current user's home directory, i.e., "$HOME/". Shell expands "~foo/" to foo's home directory, i.e., "/home/foo/".

See Section 12.1.5 if $HOME isn't available for your program.

1.5.5 Command line options

Some commands take arguments. Arguments starting with "-" or "--" are called options and control the behavior of the command.

```
$ date
Thu 20 May 2021 01:08:08 AM JST
$ date -R
Thu, 20 May 2021 01:08:12 +0900
```

Here the command-line argument "-R" changes date(1) behavior to output RFC2822 compliant date string.

1.5.6 Shell glob

Often you want a command to work with a group of files without typing all of them. The filename expansion pattern using the shell **glob**, (sometimes referred as **wildcards**), facilitate this need.

shell glob pattern	description of match rule
*	filename not started with "."
.*	filename started with "." (in Bash 5.2 and later, "." and ".." are excluded by default)
?	exactly one character
[…]	exactly one character with any character enclosed in brackets
[a-z]	exactly one character with any character between "a" and "z"
[!…]	exactly one character other than any character enclosed in brackets (excluding "!")

Table 1.21: Shell glob patterns

Note
A "/" in a pathname cannot be matched by the shell glob pattern listed in the above table.

For example, try the following

```
$ mkdir foo; cd foo; touch b1.txt b2.txt b3.c b4.h .b5.txt ..b6.txt
$ mkdir bar; cd bar; touch b1.txt b2.txt b3.c b4.h .b5.txt ..b6.txt ; cd -
$ echo b*.txt
b1.txt b2.txt
$ echo b*
b1.txt b2.txt b3.c b4.h bar
$ echo */*.txt
bar/b1.txt bar/b2.txt
$ echo *.[hc]
b3.c b4.h
$ echo .*
.b5.txt ..b6.txt
$ echo b[!1-3]*
b4.h bar
$ echo b*/b[!1-3]*
bar/b4.h
$ cd ..; rm -rf junk
```

See glob(7).

Note
Unlike normal filename expansion by the shell, the shell pattern "*" tested in find(1) with "-name" test etc., matches the initial "." of the filename. (New POSIX feature)

Note
BASH can be tweaked to change its glob behavior with its shopt builtin options such as "dotglob", "noglob", "nocaseglob", "nullglob", "extglob", etc. See bash(1).

1.5.7 Return value of the command

Each command returns its exit status (variable: "$?") as the return value.

command exit status	numeric return value	logical return value
success	zero, 0	**TRUE**
error	non-zero, -1	**FALSE**

Table 1.22: Command exit codes

For example, try the following.

```
$ [ 1 = 1 ] ; echo $?
0
$ [ 1 = 2 ] ; echo $?
1
```

Note
Please note that, in the logical context for the shell, **success** is treated as the logical **TRUE** which has 0 (zero) as its value. This is somewhat non-intuitive and needs to be reminded here.

command idiom	description
`command &`	**background** execution of `command` in the subshell
`command1 | command2`	**pipe** the standard output of `command1` to the standard input of `command2` (**concurrent** execution)
`command1 2>&1 | command2`	**pipe** both standard output and standard error of `command1` to the standard input of `command2` (**concurrent** execution)
`command1 ; command2`	execute `command1` and `command2` **sequentially**
`command1 && command2`	execute `command1`; if successful, execute `command2` **sequentially** (return success if both `command1` **and** `command2` are successful)
`command1 || command2`	execute `command1`; if not successful, execute `command2` **sequentially** (return success if `command1` **or** `command2` are successful)
`command > foo`	redirect standard output of `command` to a file `foo` (overwrite)
`command 2> foo`	redirect standard error of `command` to a file `foo` (overwrite)
`command >> foo`	redirect standard output of `command` to a file `foo` (append)
`command 2>> foo`	redirect standard error of `command` to a file `foo` (append)
`command > foo 2>&1`	redirect both standard output and standard error of `command` to a file `foo`
`command < foo`	redirect standard input of `command` to a file `foo`
`command << delimiter`	redirect standard input of `command` to the following lines until "`delimiter`" is met (here document)
`command <<- delimiter`	redirect standard input of `command` to the following lines until "`delimiter`" is met (here document, the leading tab characters are stripped from input lines)

Table 1.23: Shell command idioms

1.5.8 Typical command sequences and shell redirection

Let's try to remember following shell command idioms typed in one line as a part of shell command.

The Debian system is a multi-tasking system. Background jobs allow users to run multiple programs in a single shell. The management of the background process involves the shell builtins: `jobs`, `fg`, `bg`, and `kill`. Please read sections of bash(1) under "SIGNALS", and "JOB CONTROL", and builtins(1).

For example, try the following

```
$ </etc/motd pager
```

```
$ pager </etc/motd
```

```
$ pager /etc/motd
```

```
$ cat /etc/motd | pager
```

Although all 4 examples of shell redirections display the same thing, the last example runs an extra `cat` command and wastes resources with no reason.

The shell allows you to open files using the `exec` builtin with an arbitrary file descriptor.

```
$ echo Hello >foo
$ exec 3<foo 4>bar  # open files
$ cat <&3 >&4       # redirect stdin to 3, stdout to 4
$ exec 3<&- 4>&-    # close files
$ cat bar
Hello
```

The file descriptor 0-2 are predefined.

device	description	file descriptor
stdin	standard input	0
stdout	standard output	1
stderr	standard error	2

Table 1.24: Predefined file descriptors

1.5.9 Command alias

You can set an alias for the frequently used command.

For example, try the following

```
$ alias la='ls -la'
```

Now, "la" works as a short hand for "ls -la" which lists all files in the long listing format.

You can list any existing aliases by alias (see bash(1) under "SHELL BUILTIN COMMANDS").

```
$ alias
...
alias la='ls -la'
```

You can identity exact path or identity of the command by type (see bash(1) under "SHELL BUILTIN COMMANDS").

For example, try the following

```
$ type ls
ls is hashed (/bin/ls)
$ type la
la is aliased to ls -la
$ type echo
echo is a shell builtin
$ type file
file is /usr/bin/file
```

Here ls was recently searched while "file" was not, thus "ls" is "hashed", i.e., the shell has an internal record for the quick access to the location of the "ls" command.

Tip
See Section 9.3.6.

1.6 Unix-like text processing

In Unix-like work environment, text processing is done by piping text through chains of standard text processing tools. This was another crucial Unix innovation.

1.6.1 Unix text tools

There are few standard text processing tools which are used very often on the Unix-like system.

- No regular expression is used:
  - cat(1) concatenates files and outputs the whole content.

  - tac(1) concatenates files and outputs in reverse.
  - cut(1) selects parts of lines and outputs.
  - head(1) outputs the first part of files.
  - tail(1) outputs the last part of files.
  - sort(1) sorts lines of text files.
  - uniq(1) removes duplicate lines from a sorted file.
  - tr(1) translates or deletes characters.
  - diff(1) compares files line by line.
- Basic regular expression (**BRE**) is used as default:
  - ed(1) is a primitive line editor.
  - sed(1) is a stream editor.
  - grep(1) matches text with patterns.
  - vim(1) is a screen editor.
  - emacs(1) is a screen editor. (somewhat extended **BRE**)
- Extended regular expression (**ERE**) is used:
  - awk(1) does simple text processing.
  - egrep(1) matches text with patterns.
  - `tcl`(3tcl) can do every conceivable text processing: See re_syntax(3). Often used with `tk`(3tk).
  - perl(1) can do every conceivable text processing. See perlre(1).
  - pcre2grep(1) from the `pcre2-util` package matches text with Perl Compatible Regular Expressions (PCRE) pattern.
  - python(1) with the `re` module can do every conceivable text processing. See "`/usr/share/doc/python/html/inde`

If you are not sure what exactly these commands do, please use "`man command`" to figure it out by yourself.

Note
Sort order and range expression are locale dependent. If you wish to obtain traditional behavior for a command, use **C** locale or **C.UTF-8** locale instead of normal **UTF-8** ones (see Section 8.1).

Note
Perl regular expressions (perlre(1)), Perl Compatible Regular Expressions (PCRE), and Python regular expressions offered by the `re` module have many common extensions to the normal **ERE**.

1.6.2 Regular expressions

Regular expressions are used in many text processing tools. They are analogous to the shell globs, but they are more complicated and powerful.

The regular expression describes the matching pattern and is made up of text characters and **metacharacters**.

A **metacharacter** is just a character with a special meaning. There are 2 major styles, **BRE** and **ERE**, depending on the text tools as described above.

The regular expression of **`emacs`** is basically **BRE** but has been extended to treat "+"and "?" as the **metacharacters** as in **ERE**. Thus, there are no needs to escape them with "\" in the regular expression of `emacs`.

grep(1) can be used to perform the text search using the regular expression.

For example, try the following

BRE	ERE	description of the regular expression
`\ . [ ] ^ $ *`	`\ . [ ] ^ $ *`	common **metacharacters**
`\+ \? \( \) \{ \} |`		BRE only "\" escaped **metacharacters**
	`+ ? ( ) { } |`	ERE only non-"\" escaped **metacharacters**
c	c	match **non-metacharacter** "c"
`\c`	`\c`	match a literal character "c" even if "c" is **metacharacter** by itself
`.`	`.`	match any character including newline
`^`	`^`	position at the beginning of a string
`$`	`$`	position at the end of a string
`\<`	`\<`	position at the beginning of a word
`\>`	`\>`	position at the end of a word
`[abc…]`	`[abc…]`	match any characters in "abc…"
`[^abc…]`	`[^abc…]`	match any characters except in "abc…"
`r*`	`r*`	match zero or more regular expressions identified by "r"
`r\+`	`r+`	match one or more regular expressions identified by "r"
`r\?`	`r?`	match zero or one regular expressions identified by "r"
`r1|r2`	`r1|r2`	match one of the regular expressions identified by "r1" or "r2"
`\(r1|r2\)`	`(r1|r2)`	match one of the regular expressions identified by "r1" or "r2" and treat it as a **bracketed** regular expression

Table 1.25: Metacharacters for BRE and ERE

```
$ egrep 'GNU.*LICENSE|Yoyodyne' /usr/share/common-licenses/GPL
GNU GENERAL PUBLIC LICENSE
GNU GENERAL PUBLIC LICENSE
Yoyodyne, Inc., hereby disclaims all copyright interest in the program
```

Tip
See Section 9.3.6.

1.6.3 Replacement expressions

For the replacement expression, some characters have special meanings.

replacement expression	description of the text to replace the replacement expression
&	what the regular expression matched (use `\&` in `emacs`)
\n	what the n-th **bracketed** regular expression matched ("n" being number)

Table 1.26: The replacement expression

For Perl replacement string, "$&" is used instead of "&" and "$n" is used instead of "\n".

For example, try the following

```
$ echo zzz1abc2efg3hij4 | \
sed -e 's/\(1[a-z]*\)[0-9]*\(.*\)$/=&=/'
zzz=1abc2efg3hij4=
$ echo zzz1abc2efg3hij4 | \
sed -E -e 's/(1[a-z]*)[0-9]*(.*)$/=&=/'
zzz=1abc2efg3hij4=
$ echo zzz1abc2efg3hij4 | \
perl -pe 's/(1[a-z]*)[0-9]*(.*)$/=$&=/'
zzz=1abc2efg3hij4=
$ echo zzz1abc2efg3hij4 | \
sed -e 's/\(1[a-z]*\)[0-9]*\(.*\)$/\2===\1/'
zzzefg3hij4===1abc
$ echo zzz1abc2efg3hij4 | \
sed -E -e 's/(1[a-z]*)[0-9]*(.*)$/\2===\1/'
zzzefg3hij4===1abc
$ echo zzz1abc2efg3hij4 | \
perl -pe 's/(1[a-z]*)[0-9]*(.*)$/$2===$1/'
zzzefg3hij4===1abc
```

Here please pay extra attention to the style of the **bracketed** regular expression and how the matched strings are used in the text replacement process on different tools.

These regular expressions can be used for cursor movements and text replacement actions in some editors too.

The back slash "\" at the end of line in the shell commandline escapes newline as a white space character and continues shell command line input to the next line.

Please read all the related manual pages to learn these commands.

1.6.4 Global substitution with regular expressions

The ed(1) command can replace all instances of "FROM_REGEX" with "TO_TEXT" in "file".

```
$ ed file <<EOF
,s/FROM_REGEX/TO_TEXT/g
w
q
EOF
```

The sed(1) command can replace all instances of "FROM_REGEX" with "TO_TEXT" in "file".

```
$ sed -i -e 's/FROM_REGEX/TO_TEXT/g' file
```

The vim(1) command can replace all instances of "FROM_REGEX" with "TO_TEXT" in "file" by using ex(1) commands.

```
$ vim '+%s/FROM_REGEX/TO_TEXT/gc' '+update' '+q' file
```

Tip
The "c" flag in the above ensures interactive confirmation for each substitution.

Multiple files ("file1", "file2", and "file3") can be processed with regular expressions similarly with vim(1) or perl(1).

```
$ vim '+argdo %s/FROM_REGEX/TO_TEXT/gce|update' '+q' file1 file2 file3
```

Tip
The ”e” flag in the above prevents the ”No match” error from breaking a mapping.

```
$ perl -i -p -e 's/FROM_REGEX/TO_TEXT/g;' file1 file2 file3
```

In the perl(1) example, ”`-i`” is for the in-place editing of each target file, and ”`-p`” is for the implicit loop over all given files.

Tip
Use of argument ”`-i.bak`” instead of ”`-i`” keeps each original file by adding ”`.bak`” to its filename. This makes recovery from errors easier for complex substitutions.

Note
ed(1) and vim(1) are **BRE**; perl(1) is **ERE**.

1.6.5 Extracting data from text file table

Let’s consider a text file called ”`DPL`” in which some pre-2004 Debian project leader’s names and their initiation date are listed in a space-separated format.

```
Ian     Murdock   August  1993
Bruce   Perens    April   1996
Ian     Jackson   January 1998
Wichert Akkerman  January 1999
Ben     Collins   April   2001
Bdale   Garbee    April   2002
Martin  Michlmayr March   2003
```

Tip
See ”A Brief History of Debian” for the latest Debian leadership history.

Awk is frequently used to extract data from these types of files.

For example, try the following

```
$ awk '{ print $3 }' <DPL                   # month started
August
April
January
January
April
April
March
$ awk '($1=="Ian") { print }' <DPL          # DPL called Ian
Ian     Murdock   August  1993
Ian     Jackson   January 1998
$ awk '($2=="Perens") { print $3,$4 }' <DPL # When Perens started
April 1996
```

Shells such as Bash can be also used to parse this kind of file.

For example, try the following

```
$ while read first last month year; do
    echo $month
  done <DPL
... same output as the first Awk example
```

Here, the `read` builtin command uses characters in "`$IFS`" (internal field separators) to split lines into words.

If you change "`$IFS`" to "`:`", you can parse "`/etc/passwd`" with shell nicely.

```
$ oldIFS="$IFS"   # save old value
$ IFS=':'
$ while read user password uid gid rest_of_line; do
    if [ "$user" = "bozo" ]; then
      echo "$user's ID is $uid"
    fi
  done < /etc/passwd
bozo's ID is 1000
$ IFS="$oldIFS"   # restore old value
```

(If Awk is used to do the equivalent, use "`FS=':'`" to set the field separator.)

IFS is also used by the shell to split results of parameter expansion, command substitution, and arithmetic expansion. These do not occur within double or single quoted words. The default value of IFS is *space*, *tab*, and *newline* combined.

Be careful about using this shell IFS tricks. Strange things may happen, when shell interprets some parts of the script as its **input**.

```
$ IFS=":,"                        # use ":" and "," as IFS
$ echo IFS=$IFS,   IFS="$IFS"     # echo is a Bash builtin
IFS=  , IFS=:,
$ date -R                         # just a command output
Sat, 23 Aug 2003 08:30:15 +0200
$ echo $(date -R)                 # sub shell --> input to main shell
Sat  23 Aug 2003 08 30 36 +0200
$ unset IFS                       # reset IFS to the default
$ echo $(date -R)
Sat, 23 Aug 2003 08:30:50 +0200
```

1.6.6 Script snippets for piping commands

The following scripts do nice things as a part of a pipe.

A one-line shell script can loop over many files using find(1) and xargs(1) to perform quite complicated tasks. See Section 10.1.5 and Section 9.4.9.

When using the shell interactive mode becomes too complicated, please consider to write a shell script (see Section 12.1).

script snippet (type in one line)	effect of command
`find /usr -print`	find all files under "`/usr`"
`seq 1 100`	print 1 to 100
`| xargs -n 1 command`	run command repeatedly with each item from pipe as its argument
`| xargs -n 1 echo`	split white-space-separated items from pipe into lines
`| xargs echo`	merge all lines from pipe into a line
`| grep -e regex_pattern`	extract lines from pipe containing *regex_pattern*
`| grep -v -e regex_pattern`	extract lines from pipe not containing *regex_pattern*
`| cut -d: -f3 -`	extract third field from pipe separated by "`:`" (passwd file etc.)
`| awk '{ print $3 }'`	extract third field from pipe separated by whitespaces
`| awk -F'\t' '{ print $3 }'`	extract third field from pipe separated by tab
`| col -bx`	remove backspace and expand tabs to spaces
`| expand -`	expand tabs
`| sort| uniq`	sort and remove duplicates
`| tr 'A-Z' 'a-z'`	convert uppercase to lowercase
`| tr -d '\n'`	concatenate lines into one line
`| tr -d '\r'`	remove CR
`| sed 's/^/# /'`	add "#" to the start of each line
`| sed 's/\.ext//g'`	remove "`.ext`"
`| sed -n -e 2p`	print the second line
`| head -n 2 -`	print the first 2 lines
`| tail -n 2 -`	print the last 2 lines

Table 1.27: List of script snippets for piping commands

Chapter 2

Debian package management

Note
This chapter is written assuming the latest stable release is codename: `trixie`.
The data source of the APT system is collectively referred as **the source list** in this document . This can be defined anywhere in the "`/etc/apt/sources.list`" file, "`/etc/apt/sources.list.d/*.list`" files, or "`/etc/apt/sources.list.d/*.sources`" files.

2.1 Debian package management prerequisites

2.1.1 Debian package management system

Debian is a volunteer organization which builds **consistent** distributions of pre-compiled binary packages of free software and distributes them from its archive.

The Debian archive is offered by many remote mirror sites for access through HTTP and FTP methods. It is also available as CD-ROM/DVD.

The current Debian package management system which can utilize all these resources is Advanced Packaging Tool (APT).

The Debian package management system, **when used properly**, offers the user to install **consistent sets of binary packages** to the system from the archive. Currently, there are 78072 packages available for the amd64 architecture.

The Debian package management system has a rich history and many choices for the front end user program and back end archive access method to be used. Currently, we recommend the following.

- apt(8) for all interactive command line operations, including package installation, removal and dist-upgrades.
- apt-get(8) for calling Debian package management system from scripts. It is also a fallback option when `apt` is not available (often with older Debian systems).
- aptitude(8) for an interactive text interface to manage the installed packages and to search the available packages.

2.1.2 Package configuration

Here are some key points for package configuration on the Debian system.

- For the modern Desktop environment, rebooting of the system after package configuration change and package upgrade is a good idea to ensure the system to function properly.

package	popcon	size	description
dpkg	V:906, I:1000	4944	low level package management system for Debian (file based)
apt	V:895, I:1000	4956	APT front-end to manage packages with CLI: apt/apt-get/apt-cache
aptitude	V:31, I:160	4621	APT front-end to interactively manage packages with full screen console: aptitude(8)
tasksel	V:38, I:985	351	APT front-end to install selected tasks: tasksel(8)
unattended-upgrades	V:130, I:182	320	enhancement package for APT to enable automatic installation of security upgrades
gnome-software	V:158, I:263	4732	Software Center for GNOME (GUI APT front-end)
synaptic	V:31, I:269	7264	graphical package manager (GTK APT front-end)
apt-utils	V:420, I:998	1181	APT utility programs: apt-extracttemplates(1), apt-ftparchive(1), and apt-sortpkgs(1)
apt-listchanges	V:403, I:894	562	package change history notification tool
apt-listbugs	V:4.7, I:6.5	512	lists critical bugs before each APT installation
apt-file	V:14, I:53	76	APT package searching utility — command-line interface
apt-rdepends	V:0.4, I:4.2	39	recursively lists package dependencies

Table 2.1: List of Debian package management tools

- The manual configuration by the system administrator is respected. In other words, the package configuration system makes no intrusive configuration for the sake of convenience.
- Each package comes with its own configuration script with standardized user interface called debconf(7) to help initial installation process of the package.
- Debian Developers try their best to make your upgrade experience flawless with package configuration scripts.
- Full functionalities of packaged software are available to the system administrator. But ones with security risks are disabled in the default installation.
- If you manually activate a service with some security risks, you are responsible for the risk containment.
- Esoteric configuration may be manually enabled by the system administrator. This may create interference with popular generic helper programs for the system configuration.

2.1.3 Basic precautions



Warning
Do not install packages from random mixture of suites. It probably breaks the package consistency which requires deep system management knowledge, such as compiler ABI, library version, interpreter features, etc.

The newbie Debian system administrator should stay with the **`stable`** release of Debian while applying only security updates. Until you understand the Debian system very well, you should follow the following precautions.

- Do not include **`testing`** or **`unstable`** in **the source list**.
- Do not mix standard Debian with other non-Debian archives such as Ubuntu in **the source list** .
- Do not create "`/etc/apt/preferences`".
- Do not change default behavior of package management tools through configuration files without knowing their full impacts.

- Do not install random packages by "`dpkg -i random_package`".
- Do not ever install random packages by "`dpkg --force-all -i random_package`".
- Do not erase or alter files in "`/var/lib/dpkg/`".
- Do not overwrite system files by installing software programs directly compiled from source.
  - Install them into "`/usr/local`" or "`/opt`", if needed.

The non-compatible effects caused by violating above precautions to the Debian package management system may leave your system unusable.

The serious Debian system administrator who runs mission critical servers, should use extra precautions.

- Do not install any packages including security updates from Debian without thoroughly testing them with your particular configuration under safe conditions.
  - You as the system administrator are responsible for your system in the end.
  - The long stability history of the Debian system is no guarantee by itself.

2.1.4 Life with eternal upgrades



Caution
For your **production server**, the `stable` suite with the security updates is recommended. The same can be said for desktop PCs on which you can spend limited administration efforts.

Despite my warnings above, I know many readers of this document may wish to run the newer `testing` or `unstable` suites.

Enlightenment with the following saves a person from the eternal karmic struggle of upgrade hell and let him reach Debian nirvana.

This list is targeted for the **self-administered** Desktop environment.

- Use the `testing` suite since it is practically the rolling release automatically managed by the Debian archive QA infrastructure such as the Debian continuous integration, the source only upload practices, and the library transition tracking. The packages in the `testing` suite are updated frequently enough to offer all the latest features.
- Set the codename corresponding to the `testing` suite ("`forky`" during the `trixie-as-stable` release cycle) in **the source list**.
- Manually update this codename in **the source list** to the new one only after assessing situation by yourself for about a month after the major suite release. The Debian user and developer mailing list are good sources of information for this, too.

The use of the `unstable` suite isn't recommended. The `unstable` suite is **good for debugging packages** as a developer but tends to expose you to unnecessary risks for the normal Desktop usage. Even though the `unstable` suite of the Debian system looks very stable for most of the times, there have been some package problems and a few of them were not so trivial to resolve.

Here are some basic precautionary measure ideas to ensure quick and easy recovery from bugs in Debian packages.

- Make the rescue system available by following Section 3.2.
- Make the system **dual bootable** by installing the `stable` suite of the Debian system to another partition

- Consider installing `apt-listbugs` to check the Debian Bug Tracking System (BTS) information before the upgrade
- Learn the package system infrastructure enough to work around the problem



Caution
If you can not do any one of these precautionary actions, you are probably not ready for the `testing` and `unstable` suites.

2.1.5 Debian archive basics

Tip
Official policy of the Debian archive is defined at Debian Policy Manual, Chapter 2 - The Debian Archive.

Let's look into the Debian archive from a system user's perspective.

For a system user, the Debian archive is accessed using the APT system.

The APT system specifies its data source as **the source list** and it is described in sources.list(5).

For the `trixie` system with the typical HTTP access, **the source list** is provided in the modern deb822-style in "`/etc/apt/sources.list.d/debian.sources`" as the following:

```
Types: deb deb-src
URIs: http://deb.debian.org/debian/
Suites: trixie trixie-updates
Components: main non-free-firmware contrib non-free

Types: deb deb-src
URIs: http://security.debian.org/debian-security/
Suites: trixie-security
Components: main non-free-firmware contrib non-free
```

Tip
If **the source list** is provided in the older deprecated one-line-style in "`/etc/apt/sources.list`" or "`/etc/apt/sources.list.d/*.list`" files, update them with:

```
$ sudo apt modernize-sources
```

Key points of **the source list** in deb822-style are followings.

- It's definition files are in "`/etc/apt/sources.list.d/*.sources`" files.
- Each block of lines separated by a blank line defines the data source for the APT system.
- The "`Types:`" stanza defines the list of types such as "`deb`" and "`deb-src`".
- The "`URIs:`" stanza defines the list of root URIs of the Debian archive.
- The "`Suites:`" stanza defines the list of distribution names using either the suite name or the codename.
- The "`Components:`" stanza defines the list of valid archive area names of the Debian archive.

The definition for "deb-src" can safely be omitted if it is just for aptitude which does not access source related meta data. It speeds up the updates of the archive meta data.

The URL can be "https://", "http://", "ftp://", "file://", ...

Lines starting with "#" are comments and ignored.

Here, I tend to use codename "trixie" or "forky" instead of suite name "stable" or "testing" to avoid surprises when the next stable is released.

Tip

If "sid" is used in the above example instead of "trixie", the "deb: http://security.debian.org/ ..." line or its deb822 equivalent content for security updates in **the source list** is not required. This is because there is no security update archive for "sid" (unstable).

Here is the list of URL of the Debian archive sites and suite name or codename used in the configuration file after the trixie release.

archive URL	suite name	codename	purpose of repository
http://deb.debian.org/debian/	stable	trixie	Quasi-static stable release after extensive checks
http://deb.debian.org/debian/	stable-updates	trixie-updates	Updates for the stable suite (important)
http://security.debian.org/debian-security/	stable-security	trixie-security	Security updates for the stable suite (important)
http://deb.debian.org/debian/	testing	forky	Dynamic testing release after decent checks and short waits
http://deb.debian.org/debian/	unstable	sid	Dynamic unstable release after minimal checks and no waits
http://deb.debian.org/debian/	experimental	N/A	Pre-release experiments by developers (optional, only for developer)
http://deb.debian.org/debian/	stable-proposed-updates	trixie-proposed-updates	Updates for the next stable point release (optional)
http://deb.debian.org/debian/	stable-backports	trixie-backports	Random collection of recompiled packages mostly from the testing release (optional)
http://security.debian.org/debian-security/	testing-security	forky-security	This is not actively supported nor used by the security team

Table 2.2: List of Debian archive sites

Caution

Only pure **stable** release with security updates provides the best stability. Running mostly **stable** release mixed with some packages from **testing** or **unstable** release is riskier than running pure **unstable** release for library version mismatch etc. If you really need the latest version of some programs under **stable** release, please use packages from stable-updates and backports (see Section 2.7.4) services. These services must be used with extra care.

Caution

You should basically list only one of stable, testing, or unstable suites in the "deb" line. If you list any combination of stable, testing, and unstable suites in the "deb" line, APT programs slow down while only the latest archive is effective. Multiple listing makes sense for these when the "/etc/apt/preferences" file is used with clear objectives (see Section 2.7.7).

Tip
For the Debian system with the `stable` suite, it is a good idea to include the content with "`http://security.debian.org/`" in **the source list** to enable security updates as in the example above.

Note
The security bugs for the `stable` archive are fixed by the Debian security team. This activity has been quite rigorous and reliable. Those for the `testing` archive may be fixed by the Debian testing security team. For several reasons, this activity is not as rigorous as that for `stable` and you may need to wait for the migration of fixed `unstable` packages to the `testing` archive. Those for the `unstable` archive are fixed by the individual maintainer. Actively maintained `unstable` packages are usually in a fairly good shape by leveraging latest upstream security fixes. See Debian security FAQ for how Debian handles security bugs.

area	number of packages	criteria of package component
`main`	76587	DFSG compliant and no dependency to `non-free`
`non-free-firmware`	78	not DFSG compliant, firmware required for reasonable system installation experience
`contrib`	362	DFSG compliant but having dependency to `non-free`
`non-free`	1045	not DFSG compliant and not in `non-free-firmware`

Table 2.3: List of Debian archive area

Here the number of packages in the above is for the amd64 architecture. The `main` area provides the Debian system (see Section 2.1.6).

The Debian archive organization can be studied best by pointing your browser to the each archive URL appended with `dists` or `pool`.

The distribution is referred by two ways, the suite or codename. The word distribution is alternatively used as the synonym to the suite in many documentations. The relationship between the suite and the codename can be summarized as the following.

Timing	suite = `stable`	suite = `testing`	suite = `unstable`
after the `trixie` release	codename = `trixie`	codename = `forky`	codename = `sid`
after the `forky` release	codename = `forky`	codename = `duke`	codename = `sid`

Table 2.4: The relationship between suite and codename

The history of codenames are described in Debian FAQ: 6.2.1 Which other codenames have been used in the past?

In the stricter Debian archive terminology, the word "section" is specifically used for the categorization of packages by the application area. (Although, the word "main section" may sometimes be used to describe the Debian archive area named as "main".)

Every time a new upload is done by a Debian developer (DD) to the `unstable` archive (via incoming processing), the DD is required to ensure uploaded packages to be compatible with the latest set of packages in the latest `unstable` archive.

If DD breaks this compatibility intentionally for important library upgrade etc, there is usually announcement to the debian-devel mailing list etc.

Before a set of packages are moved by the Debian archive maintenance script from the `unstable` archive to the `testing` archive, the archive maintenance script not only checks the maturity (about 2-10 days old) and the status of the RC bug reports for the packages but also tries to ensure them to be compatible with the latest set of packages in the `testing` archive. This process makes the `testing` archive very current and usable.

Through the gradual archive freeze process led by the release team, the `testing` archive is matured to make it completely consistent and bug free with some manual interventions. Then the new `stable` release is created by assigning the codename for the old `testing` archive to the new `stable` archive and creating the new codename for the new `testing` archive. The initial contents of the new `testing` archive is exactly the same as that of the newly released `stable` archive.

Both the `unstable` and the `testing` archives may suffer temporary glitches due to several factors.

- Broken package upload to the archive (mostly for `unstable`)
- Delay of accepting the new packages to the archive (mostly for `unstable`)
- Archive synchronization timing issue (both for `testing` and `unstable`)
- Manual intervention to the archive such as package removal (more for `testing`) etc.

So if you ever decide to use these archives, you should be able to fix or work around these kinds of glitches.



Caution
For about few months after a new `stable` release, most desktop users should use the `stable` archive with its security updates even if they usually use `unstable` or `testing` archives. For this transition period, both `unstable` and `testing` archives are not good for most people. Your system is difficult to keep in good working condition with the `unstable` archive since it suffers surges of major upgrades for core packages. The `testing` archive is not useful either since it contains mostly the same content as the `stable` archive without its security support (Debian testing-security-announce 2008-12). After a month or so, `unstable` or `testing` archives may become useful if you are careful.

Tip
When tracking the `testing` archive, a problem caused by a removed package is usually worked around by installing corresponding package from the `unstable` archive which is uploaded for bug fix.

See Debian Policy Manual for archive definitions.

- "Sections"
- "Priorities"
- "Base system"
- "Essential packages"

2.1.6 Debian is 100% free software

Debian is 100% free software because of the followings:

- Debian installs only free software by default to respect user's freedoms.
- Debian provides only free software in `main`.
- Debian recommends running only free software from `main`.
- No packages in `main` depend nor recommend packages in `non-free` nor `non-free-firmware` nor `contrib`.

Some people wonder if the following 2 facts contradict or not.

- ”Debian will remain 100% free”. (First term of Debian Social Contract)
- Debian servers host some `non-free-firmware`, `non-free` and `contrib` packages.

These do not contradict, because of the followings.

- The Debian system is 100% free and its packages are hosted by Debian servers in the `main` area.
- Packages outside of the Debian system are hosted by Debian servers in the `non-free`, `non-free-firmware` and `contrib` areas.

These are precisely explained in the 4th and 5th terms of Debian Social Contract:

- Our priorities are our users and free software
  - We will be guided by the needs of our users and the free software community. We will place their interests first in our priorities. We will support the needs of our users for operation in many different kinds of computing environments. We will not object to non-free works that are intended to be used on Debian systems, or attempt to charge a fee to people who create or use such works. We will allow others to create distributions containing both the Debian system and other works, without any fee from us. In furtherance of these goals, we will provide an integrated system of high-quality materials with no legal restrictions that would prevent such uses of the system.
- Works that do not meet our free software standards
  - We acknowledge that some of our users require the use of works that do not conform to the Debian Free Software Guidelines. We have created ”`non-free`”, ”`non-free-firmware`” and ”`contrib`” areas in our archive for these works. The packages in these areas are not part of the Debian system, although they have been configured for use with Debian. We encourage CD manufacturers to read the licenses of the packages in these areas and determine if they can distribute the packages on their CDs. Thus, although non-free works are not a part of Debian, we support their use and provide infrastructure for non-free packages (such as our bug tracking system and mailing lists). The Debian official media may include firmware that is otherwise not part of the Debian system to enable use of Debian with hardware that requires such firmware.

Note
The actual text of the 5th term in the current Debian Social Contract 1.2 is slightly different from the above text. This editorial deviation is intentional one to make this user document consistent without changing the real content of the Social Contract.

Users should be aware of the risks of using packages in the `non-free`, `non-free-firmware` and `contrib` areas:

- lack of freedom for such software packages
- lack of support from Debian on such software packages (Debian can’t support software properly without having access to its source code.)
- contamination of your 100% free Debian system

The Debian Free Software Guidelines are the free software standards for Debian. Debian interprets ”software” in the widest scope including document, firmware, logo, and artwork data in the package. This makes Debian’s free software standards very strict ones.

Typical `non-free`, `non-free-firmware` and `contrib` packages include freely distributable packages of following types:

- Document packages under GNU Free Documentation License with invariant sections such as ones for GCC and Make. (mostly found in the `non-free/doc` section.)

- Firmware packages containing sourceless binary data such as ones listed in Section 9.10.5 as `non-free-firmware`. (mostly found in the `non-free-firmware/kernel` section.)
- Game and font packages with restriction on commercial use and/or content modification.

Please note that the number of `non-free`, `non-free-firmware` and `contrib` packages is less than 2% of that of `main` packages. Enabling access to the `non-free`, `non-free-firmware` and `contrib` areas does not obscure the source of packages. Interactive full screen use of aptitude(8) provides you with full visibility and control over what packages are installed from which area to keep your system as free as you wish.

2.1.7 Package dependencies

The Debian system offers a consistent set of binary packages through its versioned binary dependency declaration mechanism in the control file fields. Here is a bit over simplified definition for them.

- ”Depends”
  - This declares an absolute dependency and all of the packages listed in this field must be installed at the same time or in advance.
- ”Pre-Depends”
  - This is like Depends, except that it requires completed installation of the listed packages in advance.
- ”Recommends”
  - This declares a strong, but not absolute, dependency. Most users would not want the package unless all of the packages listed in this field are installed.
- ”Suggests”
  - This declares a weak dependency. Many users of this package may benefit from installing packages listed in this field but can have reasonable functions without them.
- ”Enhances”
  - This declares a weak dependency like Suggests but works in the opposite direction.
- ”Breaks”
  - This declares a package incompatibility usually with some version specification. Generally the resolution is to upgrade all of the packages listed in this field.
- ”Conflicts”
  - This declares an absolute incompatibility. All of the packages listed in this field must be removed to install this package.
- ”Replaces”
  - This is declared when files installed by this package replace files in the listed packages.
- ”Provides”
  - This is declared when this package provide all of the files and functionality in the listed packages.

Note
Please note that defining ”Provides”, ”Conflicts” and ”Replaces” simultaneously to an virtual package is the sane configuration. This ensures that only one real package providing this virtual package can be installed at any one time.

The official definition including source dependency can be found in the Policy Manual: Chapter 7 - Declaring relationships between packages.

2.1.8 The event flow of the package management

Here is a summary of the simplified event flow of the package management by APT.

- **Update** (”`apt update`”, ”`aptitude update`” or ”`apt-get update`”):
  1. Fetch archive metadata from remote archive
  2. Reconstruct and update local metadata for use by APT
- **Upgrade** (”`apt upgrade`” and ”`apt full-upgrade`”, or ”`aptitude safe-upgrade`” and ”`aptitude full-upgrad` or ”`apt-get upgrade`” and ”`apt-get dist-upgrade`”):
  1. Choose candidate version which is usually the latest available version for all installed packages (see Section 2.7.7 for exception)
  2. Make package dependency resolution
  3. Fetch selected binary packages from remote archive if candidate version is different from installed version
  4. Unpack fetched binary packages
  5. Run **preinst** script
  6. Install binary files
  7. Run **postinst** script
- **Install** (”`apt install …`”, `aptitude install …`” or ”`apt-get install …`”):
  1. Choose packages listed on the command line
  2. Make package dependency resolution
  3. Fetch selected binary packages from remote archive
  4. Unpack fetched binary packages
  5. Run **preinst** script
  6. Install binary files
  7. Run **postinst** script
- **Remove** (”`apt remove …`”, ”`aptitude remove …`” or ”`apt-get remove …`”):
  1. Choose packages listed on the command line
  2. Make package dependency resolution
  3. Run **prerm** script
  4. Remove installed files **except** configuration files
  5. Run **postrm** script
- **Purge** (”`apt purge`”, ”`aptitude purge …`” or ”`apt-get purge …`”):
  1. Choose packages listed on the command line
  2. Make package dependency resolution
  3. Run **prerm** script
  4. Remove installed files **including** configuration files
  5. Run **postrm** script

Here, I intentionally skipped technical details for the sake of big picture.

2.1.9 First response to package management troubles

You should read the fine official documentation. The first document to read is the Debian specific "`/usr/share/doc/packa`
Other documentation in "`/usr/share/doc/package_name/`" should be consulted too. If you set shell as Section 1.4.2, type the following.

```
$ cd package_name
$ pager README.Debian
$ mc
```

You may need to install the corresponding documentation package named with "`-doc`" suffix for detailed information.

If you are experiencing problems with a specific package, make sure to check out the Debian bug tracking system (BTS) sites, first.

web site	command
Home page of the Debian bug tracking system (BTS)	`sensible-browser "https://bugs.debian.org/"`
The bug report of a known package name	`sensible-browser "https://bugs.debian.org/package_name"`
The bug report of known bug number	`sensible-browser "https://bugs.debian.org/bug_number"`

Table 2.5: List of key web site to resolving problems with a specific package

Search Google with search words including "`site:debian.org`", "`site:wiki.debian.org`", "`site:lists.debian.o`
etc.

When you file a bug report, please use reportbug(1) command.

2.1.10 How to pick Debian packages

When you encounter more than 2 similar packages and wonder which one to install without "trial and error" efforts, you should use some **common sense**. I consider following points are good indications of preferred packages.

- Essential: yes > no
- Area: main > contrib > non-free
- Priority: required > important > standard > optional > extra
- Tasks: packages listed in tasks such as "Desktop environment"
- Packages selected by the dependency package (e.g., `gcc-16` by `gcc`)
- Popcon: higher in the vote and install number
- Changelog: regular updates by the maintainer
- BTS: No RC bugs (no critical, no grave, and no serious bugs)
- BTS: responsive maintainer to bug reports
- BTS: higher number of the recently fixed bugs
- BTS: lower number of remaining non-wishlist bugs

Debian being a volunteer project with distributed development model, its archive contains many packages with different focus and quality. You must make your own decision what to do with them.

2.1.11 How to cope with conflicting requirements

Whatever suite of Debian system you may decide to use, you may still wish to run versions of programs which aren't available in that suite. Even if you find binary packages of such programs in other Debian suites or in other non-Debian resources, their requirements may conflict with your current Debian system.

Although you can tweak package management system with **apt-pinning** technique etc. as described in Section 2.7.7 to install such out-of-sync binary packages, such tweaking approaches have only limited use cases since they may break those programs and your system.

Before brutally installing such out-of-sync packages, you should seek all available alternative safer technical solutions which are compatible with your current Debian syetem.

- Install such programs using corresponding sandboxed upstream binary packages (see Section 7.7).
  - Many mostly GUI programs such as LibreOffice and GNOME applications are available as Flatpak, Snap, or AppImage packages.
- Create a chroot or similar environment and run such programs in it (see Section 9.11).
  - CLI commands can be executed easily under its compatible chroot (see Section 9.11.4).
  - Multiple full desktop environments can be tried easily without reboot (see Section 9.11.5).
- Build desired versions of binary packages which are compatible with your current Debian syetem by yourself.
  - This is a non-trivial task (see Section 2.7.13).

2.2 Basic package management operations

Repository based package management operations on the Debian system can be performed by many APT-based package management tools available on the Debian system. Here, we explain 3 basic package management tools: `apt`, `apt-get` / `apt-cache` and `aptitude`.

For the package management operation which involves package installation or updates package metadata, you need to have root privilege.

2.2.1 `apt` vs. `apt-get` / `apt-cache` vs. `aptitude`

Although `aptitude` is a very nice interactive tool which the author mainly uses, you should know some cautionary facts:

- The `aptitude` command is not recommended for the release-to-release system upgrade on the `stable` Debian system after the new release.
  - The use of "`apt full-upgrade`" or "`apt-get dist-upgrade`" is recommended for it. See Bug #411280.
- The `aptitude` command sometimes suggests mass package removals for the system upgrade on the `testing` or `unstable` Debian system.
  - This situation has frightened many system administrators. Don't panic.
  - This seems to be caused mostly by the version skew among packages depended or recommended by a meta-package such as `gnome-core`.
  - This can be resolved by selecting "Cancel pending actions" in the `aptitude` command menu, exiting `aptitude`, and using "`apt full-upgrade`".

The `apt-get` and `apt-cache` commands are the most **basic** APT-based package management tools.

- `apt-get` and `apt-cache` offer only the commandline user interface.
- `apt-get` is most suitable for the **major system upgrade** between releases, etc.
- `apt-get` offers a **robust** package dependency resolver.
- `apt-get` is less demanding on hardware resources. It consumes less memory and runs faster.
- `apt-cache` offers a **standard** regex based search on the package name and description.
- `apt-get` and `apt-cache` can manage multiple versions of packages using `/etc/apt/preferences` but it is quite cumbersome.

The `apt` command is a high-level commandline interface for package management. It is basically a wrapper of `apt-get`, `apt-cache` and similar commands, originally intended as an end-user interface and enables some options better suited for interactive usage by default.

- `apt` provides a friendly progress bar when installing packages using `apt install`.
- `apt` will **remove** cached `.deb` packages by default after successful installation of downloaded packages.

Tip
Users are recommended to use the new apt(8) command for **interactive** usage and use the apt-get(8) and apt-cache(8) commands in the shell script.

The `aptitude` command is the most **versatile** APT-based package management tool.

- `aptitude` offers the fullscreen interactive text user interface.
- `aptitude` offers the commandline user interface, too.
- `aptitude` is most suitable for the **daily interactive package management** such as inspecting installed packages and searching available packages.
- `aptitude` is more demanding on hardware resources. It consumes more memory and runs slower.
- `aptitude` offers an **enhanced** regex based search on all of the package metadata.
- `aptitude` can manage multiple versions of packages without using `/etc/apt/preferences` and it is quite intuitive.

2.2.2 Basic package management operations with the commandline

Here are basic package management operations with the commandline using apt(8), aptitude(8) and apt-get(8) /apt-cache(8).

`apt` / `apt-get` and `aptitude` can be mixed without major troubles.

The "`aptitude why` *regex*" can list more information by "`aptitude -v why` *regex*". Similar information can be obtained by "`apt rdepends` *package*" or "`apt-cache rdepends` *package*".

When `aptitude` command is started in the commandline mode and faces some issues such as package conflicts, you can switch to the full screen interactive mode by pressing "e"-key later at the prompt.

Note
Although the `aptitude` command comes with rich features such as its enhanced package resolver, this complexity has caused (or may still causes) some regressions such as Bug #411123, Bug #514930, and Bug #570377. In case of doubt, please use the `apt`, `apt-get` and `apt-cache` commands over the `aptitude` command.

You may provide command options right after "`aptitude`".

See aptitude(8) and "aptitude user's manual" at "`/usr/share/doc/aptitude/README`" for more.

apt syntax	**aptitude syntax**	**apt-get/apt-cache syntax**	**description**
`apt update`	`aptitude update`	`apt-get update`	update package archive metadata
`apt install foo`	`aptitude install foo`	`apt-get install foo`	install candidate version of ”`foo`” package with its dependencies
`apt upgrade`	`aptitude safe-upgrade`	`apt-get upgrade`	install candidate version of installed packages without removing any other packages
`apt full-upgrade`	`aptitude full-upgrade`	`apt-get dist-upgrade`	install candidate version of installed packages while removing other packages if needed
`apt remove foo`	`aptitude remove foo`	`apt-get remove foo`	remove ”`foo`” package while leaving its configuration files
`apt autoremove`	N/A	`apt-get autoremove`	remove auto-installed packages which are no longer required
`apt purge foo`	`aptitude purge foo`	`apt-get purge foo`	purge ”`foo`” package with its configuration files
`apt clean`	`aptitude clean`	`apt-get clean`	clear out the local repository of retrieved package files completely
`apt autoclean`	`aptitude autoclean`	`apt-get autoclean`	clear out the local repository of retrieved package files for outdated packages
`apt show foo`	`aptitude show foo`	`apt-cache show foo`	display detailed information about ”`foo`” package
`apt search` *`regex`*	`aptitude search` *`regex`*	`apt-cache search` *`regex`*	search packages which match *regex*
N/A	`aptitude why` *`regex`*	N/A	explain the reason why *regex* matching packages should be installed
N/A	`aptitude why-not` *`regex`*	N/A	explain the reason why *regex* matching packages can not be installed
`apt list --manual-installed`	`aptitude search '~i!~M'`	`apt-mark showmanual`	list manually installed packages

Table 2.6: Basic package management operations with the commandline using apt(8), aptitude(8) and apt-get(8) /apt-cache(8)

command option	**description**
`-s`	simulate the result of the command
`-d`	download only but no install/upgrade
`-D`	show brief explanations before the automatic installations and removals

Table 2.7: Notable command options for aptitude(8)

2.2.3 Interactive use of aptitude

For the interactive package management, you start `aptitude` in interactive mode from the console shell prompt as follows.

```
$ sudo aptitude -u
Password:
```

This updates the local copy of the archive information and display the package list in the full screen with menu. Aptitude places its configuration at "`~/.aptitude/config`".

Tip
If you want to use root's configuration instead of user's one, use "`sudo -H aptitude …`" instead of "`sudo aptitude …`" in the above expression.

Tip
`Aptitude` automatically sets **pending actions** as it is started interactively. If you do not like it, you can reset it from menu: "Action" → "Cancel pending actions".

2.2.4 Key bindings of aptitude

Notable key strokes to browse status of packages and to set "planned action" on them in this full screen mode are the following.

key	key binding
`F10` or `Ctrl-t`	menu
`?`	display **help** for keystroke (more complete listing)
`F10` → Help → User's Manual	display User's Manual
`u`	update package archive information
`+`	mark the package for the **upgrade** or the **install**
`-`	mark the package for the **remove** (keep configuration files)
`_`	mark the package for the **purge** (remove configuration files)
`=`	place the package on **hold**
`U`	mark all upgradable packages (function as **full-upgrade**)
`g`	start **downloading** and **installing** selected packages
`q`	quit current screen and save changes
`x`	quit current screen and discard changes
`Enter`	view information about a package
`C`	view a package's changelog
`l`	change the limit for the displayed packages
`/`	search for the first match
`\`	repeat the last search

Table 2.8: List of key bindings for aptitude

The file name specification of the command line and the menu prompt after pressing "`l`" and "`//`" take the aptitude regex as described below. Aptitude regex can explicitly match a package name using a string started by "~n" and followed by the package name.

Tip
You need to press "U" to get all the installed packages upgraded to the **candidate version** in the visual interface. Otherwise only the selected packages and certain packages with versioned dependency to them are upgraded to the **candidate version**.

2.2.5 Package views under aptitude

In the interactive full screen mode of aptitude(8), packages in the package list are displayed as the next example.

```
idA   libsmbclient                              -2220kB 3.0.25a-1  3.0.25a-2
```

Here, this line means from the left as the following.

- The ”current state” flag (the first letter)
- The ”planned action” flag (the second letter)
- The ”automatic” flag (the third letter)
- The Package name
- The change in disk space usage attributed to ”planned action”
- The current version of the package
- The candidate version of the package

Tip
The full list of flags are given at the bottom of **Help** screen shown by pressing ”?”.

The **candidate version** is chosen according to the current local preferences (see apt_preferences(5) and Section 2.7.7).

Several types of package views are available under the menu ”`Views`”.

view	**description of view**
`Package View`	see Table 2.10 (default)
`Audit Recommendations`	list packages which are recommended by some installed packages but not yet installed
`Flat Package List`	list packages without categorization (for use with regex)
`Debtags Browser`	list packages categorized according to their debtags entries
`Source Package View`	list packages grouped by source packages

Table 2.9: List of views for aptitude

Note
Please help us improving tagging packages with debtags!

The standard ”`Package View`” categorizes packages somewhat like `dselect` with few extra features.

Tip
`Tasks` view can be used to cherry pick packages for your task.

category	description of view
`Upgradable Packages`	list packages organized as `section` → `area` → `package`
`New Packages`	,,
`Installed Packages`	,,
`Not Installed Packages`	,,
`Obsolete and Locally Created Packages`	,,
`Virtual Packages`	list packages with the same function
`Tasks`	list packages with different functions generally needed for a task

Table 2.10: The categorization of standard package views

2.2.6 Search method options with aptitude

Aptitude offers several options for you to search packages using its regex formula.

- Shell commandline:
  - "`aptitude search '`*`aptitude_regex`*`'`" to list installation status, package name and short description of matching packages
  - "`aptitude show '`*`package_name`*`'`" to list detailed description of the package
- Interactive full screen mode:
  - "`l`" to limit package view to matching packages
  - "`/`" for search to a matching package
  - "`\`" for backward search to a matching package
  - "n" for find-next
  - "`N`" for find-next (backward)

Tip
The string for *package_name* is treated as the exact string match to the package name unless it is started explicitly with "~" to be the regex formula.

2.2.7 The aptitude regex formula

The aptitude regex formula is mutt-like extended **ERE** (see Section 1.6.2) and the meanings of the `aptitude` specific special match rule extensions are as follows.

- The regex part is the same **ERE** as the one used in typical Unix-like text tools using "^", "`.*`", "$" etc. as in egrep(1), awk(1) and perl(1).
- The dependency *type* is one of (depends, predepends, recommends, suggests, conflicts, replaces, provides) specifying the package interrelationship.
- The default dependency *type* is "depends".

Tip
When *regex_pattern* is a null string, place "`~T`" immediately after the command.

Here are some short cuts.

description of the extended match rule	**regex formula**
match on package name	`~n`*`regex_name`*
match on description	`~d`*`regex_description`*
match on task name	`~t`*`regex_task`*
match on debtag	`~G`*`regex_debtag`*
match on maintainer	`~m`*`regex_maintainer`*
match on package section	`~s`*`regex_section`*
match on package version	`~V`*`regex_version`*
match archive	`~A{trixie,forky,sid}`
match origin	`~O{debian,…}`
match priority	`~p{extra,important,optional,required,standard}`
match essential packages	`~E`
match virtual packages	`~v`
match new packages	`~N`
match with pending action	`~a{install,upgrade,downgrade,remove,purge,hold,keep}`
match installed packages	`~i`
match installed packages with **A**-mark (auto installed packages)	`~M`
match installed packages without **A**-mark (administrator selected packages)	`~i!~M`
match installed and upgradable packages	`~U`
match removed but not purged packages	`~c`
match removed, purged or can-be-removed packages	`~g`
match packages declaring a broken dependency	`~b`
match packages declaring broken dependency of *type*	`~B`*`type`*
match *pattern* packages declaring dependency of *type*	`~D[`*`type`*`:]`*`pattern`*
match *pattern* packages declaring broken dependency of *type*	`~DB[`*`type`*`:]`*`pattern`*
match packages to which the *pattern* matching package declares dependency *type*	`~R[`*`type`*`:]`*`pattern`*
match packages to which the *pattern* matching package declares broken dependency *type*	`~RB[`*`type`*`:]`*`pattern`*
match packages to which some other installed packages depend on	`~R~i`
match packages to which no other installed packages depend on	`!~R~i`
match packages to which some other installed packages depend or recommend on	`~R~i|~Rrecommends:~i`
match *pattern* package with filtered version	`~S filter `*`pattern`*
match all packages (true)	`~T`
match no packages (false)	`~F`

Table 2.11: List of the aptitude regex formula

- "~P*term*" == "~Dprovides:*term*"
- "~C*term*" == "~Dconflicts:*term*"
- "…~W term" == "(…|term)"

Users familiar with `mutt` pick up quickly, as mutt was the inspiration for the expression syntax. See "SEARCHING, LIMITING, AND EXPRESSIONS" in the "User's Manual" "`/usr/share/doc/aptitude/README`".

Note
With the `lenny` version of aptitude(8), the new **long form** syntax such as "`?broken`" may be used for regex matching in place for its old **short form** equivalent "~b". Now space character " " is considered as one of the regex terminating character in addition to tilde character "~". See "User's Manual" for the new **long form** syntax.

2.2.8 Dependency resolution of aptitude

The selection of a package in `aptitude` not only pulls in packages which are defined in its "`Depends:`" list but also defined in the "`Recommends:`" list if the menu "`F10` → Options → Preferences → Dependency handling" is set accordingly. These auto installed packages are removed automatically if they are no longer needed under `aptitude`.

The flag controlling the "auto install" behavior of the `aptitude` command can also be manipulated using the aptmark(8) command from the `apt` package.

2.2.9 Package activity logs

You can check package activity history in the log files.

file	content
`/var/log/dpkg.log`	Log of `dpkg` level activity for all package activities
`/var/log/apt/term.log`	Log of generic APT activity
`/var/log/aptitude`	Log of `aptitude` command activity

Table 2.12: The log files for package activities

In reality, it is not so easy to get meaningful understanding quickly out from these logs. See Section 9.3.9 for easier way.

2.3 Examples of aptitude operations

Here are few examples of aptitude(8) operations.

2.3.1 Seeking interesting packages

You can seek packages which satisfy your needs with `aptitude` from the package description or from the list under "Tasks".

2.3.2 Listing packages with regex matching on package names

The following command lists packages with regex matching on package names.

```
$ aptitude search '~n(pam|nss).*ldap'
p libnss-ldap - NSS module for using LDAP as a naming service
p libpam-ldap - Pluggable Authentication Module allowing LDAP interfaces
```

This is quite handy for you to find the exact name of a package.

2.3.3 Browsing with the regex matching

The regex "`~dipv6`" in the "New Flat Package List" view with "`l`" prompt, limits view to packages with the matching description and let you browse their information interactively.

2.3.4 Purging removed packages for good

You can purge all remaining configuration files of removed packages.

Check results of the following command.

```
# aptitude search '~c'
```

If you think listed packages are OK to be purged, execute the following command.

```
# aptitude purge '~c'
```

You may want to do the similar in the interactive mode for fine grained control.

You provide the regex "`~c`" in the "New Package View" view with "`l`" prompt. This limits the package view only to regex matched packages, i.e., "removed but not purged". All these regex matched packages can be shown by pressing "`[`" at top level headings.

Then you press "_" at top level headings such as "Not Installed Packages". Only regex matched packages under the heading are marked to be purged by this. You can exclude some packages to be purged by pressing "=" interactively for each of them.

This technique is quite handy and works for many other command keys.

2.3.5 Tidying auto/manual install status

Here is how I tidy auto/manual install status for packages (after using non-aptitude package installer etc.).

1. Start `aptitude` in interactive mode as root.
2. Type "u", "`U`", "`f`" and "g" to update and upgrade package list and packages.
3. Type "`l`" to enter the package display limit as "`~i(~R~i|~Rrecommends:~i)`" and type "`M`" over "`Installed Packages`" as auto installed.
4. Type "`l`" to enter the package display limit as "`~prequired|~pimportant|~pstandard|~E`" and type "`m`" over "`Installed Packages`" as manual installed.
5. Type "`l`" to enter the package display limit as "`~i!~M`" and remove unused package by typing "`-`" over each of them after exposing them by typing "`[`" over "`Installed Packages`".
6. Type "`l`", to enter the package display limit as "`~i`"; then type "`m`" over "`Tasks`", to mark that packages as manual installed.

7. Exit `aptitude`.
8. Start "`apt-get -s autoremove | less`" as root to check what are not used.
9. Restart `aptitude` in interactive mode and mark needed packages as "`m`".
10. Restart "`apt-get -s autoremove | less`" as root to recheck REMOVED contain only expected packages.
11. Start "`apt-get autoremove | less`" as root to autoremove unused packages.

The "`m`" action over "`Tasks`" is an optional one to prevent mass package removal situation in future.

2.3.6 System wide upgrade

Note
When moving to a new release etc, you should consider to perform a clean installation of new system even though Debian is upgradable as described below. This provides you a chance to remove garbages collected and exposes you to the best combination of latest packages. Of course, you should make a full backup of system to a safe place (see Section 10.2) before doing this. I recommend to make a dual boot configuration using different partition to have the smoothest transition.

You can perform system wide upgrade to a newer release by changing contents of **the source list** pointing to a new release and running the "`apt update; apt dist-upgrade`" command.

To upgrade from `stable` to `testing` or `unstable` during the `trixie`-as-`stable` release cycle, you replace "`trixie`" in **the source list** example of Section 2.1.5 with "`forky`" or "`sid`".

In reality, you may face some complications due to some package transition issues, mostly due to package dependencies. The larger the difference of the upgrade, the more likely you face larger troubles. For the transition from the old `stable` to the new `stable` after its release, you can read its new Release Notes and follow the exact procedure described in it to minimize troubles.

When you decide to move from `stable` to `testing` before its formal release, there are no Release Notes to help you. The difference between `stable` and `testing` could have grown quite large after the previous `stable` release and makes upgrade situation complicated.

You should make precautionary moves for the full upgrade while gathering latest information from mailing list and using common senses.

1. Read previous "Release Notes".
2. Backup entire system (especially data and configuration information).
3. Have bootable media handy for broken bootloader.
4. Inform users on the system well in advance.
5. Record upgrade activity with script(1).
6. Apply "unmarkauto" to required packages, e.g., "`aptitude unmarkauto vim`", to prevent removal.
7. Minimize installed packages to reduce chance of package conflicts, e.g., remove desktop task packages.
8. Remove the "`/etc/apt/preferences`" file (disable **apt-pinning**).
9. Try to upgrade step wise: `oldstable` → `stable` → `testing` → `unstable`.
10. Update **the source list** to point to new archive only and run "`aptitude update`".
11. Install, optionally, new **core packages** first, e.g., "`aptitude install perl`".

12. Run the "`apt-get -s dist-upgrade`" command to assess impact.
13. Run the "`apt-get dist-upgrade`" command at last.



Caution
It is not wise to skip major Debian release when upgrading between `stable` releases.



Caution
In previous "Release Notes", GCC, Linux Kernel, initrd-tools, Glibc, Perl, APT tool chain, etc. have required some special attention for system wide upgrade.



Caution
"Release Notes" may not cover all possible cases. If you change low level configurations, your next upgrade may fail badly as "... segfault after upgrade ...".

For daily upgrade in `unstable`, see Section 2.4.3.

2.4 Advanced package management operations

2.4.1 Advanced package management operations with commandline

Here are list of other package management operations for which `aptitude` is too high-level or lacks required functionalities.

Note
For a package with the multi-arch feature, you may need to specify the architecture name for some commands. For example, use "`dpkg -L libglib2.0-0:amd64`" to list contents of the `libglib2.0-0` package for the `amd64` architecture.



Caution
Lower level package tools such as "`dpkg -i ...`" and "`debi ...`" should be carefully used by the system administrator. It does not automatically take care required package dependencies. Dpkg's commandline options "`--force-all`" and similar (see dpkg(1)) are intended to be used by experts only. Using them without fully understanding their effects may break your whole system.

Please note the following.

- All system configuration and installation commands require to be run from root.
- Unlike `aptitude` which uses regex (see Section 1.6.2), other package management commands use pattern like shell glob (see Section 1.5.6).
- apt-file(1) provided by the `apt-file` package must run "`apt-file update`" in advance.
- configure-debian(8) provided by the `configure-debian` package runs dpkg-reconfigure(8) as its backend.

command	action
`COLUMNS=120 dpkg -l` *`package_name_pattern`*	list status of an installed package for the bug report
`dpkg -L` *`package_name`*	list contents of an installed package
`dpkg -L` *`package_name`* `| egrep '/usr/share/man/man.*/.+'`	list manpages for an installed package
`dpkg -S` *`file_name_pattern`*	list installed packages which have matching file name
`apt-file search` *`file_name_pattern`*	list packages in archive which have matching file name
`apt-file list` *`package_name_pattern`*	list contents of matching packages in archive
`dpkg-reconfigure` *`package_name`*	reconfigure the exact package
`dpkg-reconfigure -plow` *`package_name`*	reconfigure the exact package with the most detailed question
`configure-debian`	reconfigure packages from the full screen menu
`dpkg --audit`	audit system for partially installed packages
`dpkg --configure -a`	configure all partially installed packages
`apt-cache policy` *`binary_package_name`*	show available version, priority, and archive information of a binary package
`apt-cache madison` *`package_name`*	show available version, archive information of a package
`apt-cache showsrc` *`binary_package_name`*	show source package information of a binary package
`apt-get build-dep` *`package_name`*	install required packages to build package
`aptitude build-dep` *`package_name`*	install required packages to build package
`apt-get source` *`package_name`*	download a source (from standard archive)
`dget` *`URL for dsc file`*	download a source packages (from other archive)
`dpkg-source -x` *`package_name_version-debian.revision`*`.dsc`	build a source tree from a set of source packages (”`*.orig.tar.gz`” and ”`*.debian.tar.gz`”/”`*.diff.gz`”)
`debuild binary`	build package(s) from a local source tree
`make-kpkg kernel_image`	build a kernel package from a kernel source tree
`make-kpkg --initrd kernel_image`	build a kernel package from a kernel source tree with initramfs enabled
`dpkg -i` *`package_name_version-debian.revision_arch`*`.deb`	install a local package to the system
`apt install /path/to/`*`package_filename`*`.deb`	install a local package to the system, meanwhile try to resolve dependency automatically
`debi` *`package_name_version-debian.revision_arch`*`.dsc`	install local package(s) to the system
`dpkg --get-selections '*' >selection.txt`	save `dpkg` level package selection state information
`dpkg --set-selections <selection.txt`	set `dpkg` level package selection state information
`echo` *`package_name`* `hold | dpkg --set-selections`	set `dpkg` level package selection state for a package to **hold** (equivalent to ”`aptitude hold` *`package_name`*”)

Table 2.13: List of advanced package management operations

- dpkg-reconfigure(8) runs package scripts using debconf(1) as its backend.
- "`apt-get build-dep`", "`apt-get source`" and "`apt-cache showsrc`" commands require "`deb-src`" entry in **the source list**.
- dget(1), debuild(1), and debi(1) require `devscripts` package.
- See (re)packaging procedure using "`apt-get source`" in Section 2.7.13.
- `make-kpkg` command requires the `kernel-package` package (see Section 9.10).
- See Section 12.9 for general packaging.

2.4.2 Verification of installed package files

The installation of `debsums` enables verification of installed package files against MD5sum values in the "`/var/lib/dpkg/i` file with debsums(1). See Section 10.3.5 for how MD5sum works.

Note
Because MD5sum database may be tampered by the intruder, debsums(1) is of limited use as a security tool. It is only good for checking local modifications by the administrator or damage due to media errors.

2.4.3 Safeguarding for package problems

Many users prefer to follow the **testing** (or **unstable**) releases of the Debian system for its new features and packages. This makes the system more prone to be hit by the critical package bugs.

The installation of the `apt-listbugs` package safeguards your system against critical bugs by checking Debian BTS automatically for critical bugs when upgrading with APT system.

The installation of the `apt-listchanges` package provides important news in "`NEWS.Debian`" when upgrading with APT system.

2.4.4 Searching on the package meta data

Although visiting Debian site https://packages.debian.org/ facilitates easy ways to search on the package meta data these days, let's look into more traditional ways.

The grep-dctrl(1), grep-status(1), and grep-available(1) commands can be used to search any file which has the general format of a Debian package control file.

The "`dpkg -S` *`file_name_pattern`*" can be used to search package names which contain files with the matching name installed by `dpkg`. But this overlooks files created by the maintainer scripts.

If you need to make more elaborate search on the dpkg meta data, you need to run "`grep -e regex_pattern *`" command in the "`/var/lib/dpkg/info/`" directory. This makes you search words mentioned in package scripts and installation query texts.

If you wish to look up package dependency recursively, you should use apt-rdepends(8).

2.5 Debian package management internals

Let's learn how the Debian package management system works internally. This should help you to create your own solution to some package problems.

2.5.1 Archive meta data

Meta data files for each distribution are stored under "`dist/`*`codename`*" on each Debian mirror sites, e.g., "`http://deb.deb`
Its archive structure can be browsed by the web browser. There are 6 types of key meta data.

file	location	content
`Release`	top of distribution	archive description and integrity information
`Release.gpg`	top of distribution	signature file for the "`Release`" file signed with the archive key
`Contents-`*`architecture`*	top of distribution	list of all files for all the packages in the pertinent archive
`Release`	top of each distribution/area/architecture combination	archive description used for the rule of apt_preferences(5)
`Packages`	top of each distribution/area/binary-architecture combination	concatenated `debian/control` for binary packages
`Sources`	top of each distribution/area/source combination	concatenated `debian/control` for source packages

Table 2.14: The content of the Debian archive meta data

In the recent archive, these meta data are stored as the compressed and differential files to reduce network traffic.

2.5.2 Top level "Release" file and authenticity

Tip
The top level "`Release`" file is used for signing the archive under the **secure APT** system.

Each suite of the Debian archive has a top level "`Release`" file, e.g., "`http://deb.debian.org/debian/dists/unstab`
as follows.

```
Origin: Debian
Label: Debian
Suite: unstable
Codename: sid
Date: Sat, 14 May 2011 08:20:50 UTC
Valid-Until: Sat, 21 May 2011 08:20:50 UTC
Architectures: alpha amd64 armel hppa hurd-i386 i386 ia64 kfreebsd-amd64 kfreebsd-i386 mips ↩
     mipsel powerpc s390 sparc
Components: main contrib non-free
Description: Debian x.y Unstable - Not Released
MD5Sum:
 bdc8fa4b3f5e4a715dd0d56d176fc789 18876880 Contents-alpha.gz
 9469a03c94b85e010d116aeeab9614c0 19441880 Contents-amd64.gz
 3d68e206d7faa3aded660dc0996054fe 19203165 Contents-armel.gz
...
```

Note
Here, you can find my rationale to use the "suite", and "codename" in Section 2.1.5. The "distribution" is used when referring to both "suite" and "codename". All archive "area" names offered by the archive are listed under "Components".

The integrity of the top level "Release" file is verified by cryptographic infrastructure called the secure apt as described in apt-secure(8).

- The cryptographic signature file "Release.gpg" is created from the authentic top level "Release" file and the secret Debian archive key.
- The public Debian archive keys are locally installed by the latest debian-archive-keyring package.
- The **secure APT** system automatically verifies the integrity of the downloaded top level "Release" file cryptographically by this "Release.gpg" file and the locally installed public Debian archive keys.
- The integrity of all the "Packages" and "Sources" files are verified by using MD5sum values in its top level "Release" file. The integrity of all package files are verified by using MD5sum values in the "Packages" and "Sources" files. See debsums(1) and Section 2.4.2.
- Since the cryptographic signature verification is a much more CPU intensive process than the MD5sum value calculation, use of MD5sum value for each package while using cryptographic signature for the top level "Release" file provides the good security with the performance (see Section 10.3).

If **the source list** entry specifies the "signed-by" option, the integrity of its downloaded top level "Release" file is verified using specified public key. This is useful when **the source list** contains non-Debian archives.

Tip
The use of apt-key(8) command for APT key management is deprecated.

Also, you can manually verify the integrity of the "Release" file with the "Release.gpg" file and the public Debian archive key posted on ftp-master.debian.org using gpg.

2.5.3 Archive level "Release" files

Tip
The archive level "Release" files are used for the rule of apt_preferences(5).

There are archive level "Release" files for all archive locations specified by **the source list**, such as "http://deb.debian.
or "http://deb.debian.org/debian/dists/sid/main/binary-amd64/Release" as follows.

```
Archive: unstable
Origin: Debian
Label: Debian
Component: main
Architecture: amd64
```



Caution
For "Archive:" stanza, suite names ("stable", "testing", "unstable", ...) are used in the Debian archive while codenames ("trusty", "xenial", "artful", ...) are used in the Ubuntu archive.

For some archives, such as experimental, and trixie-backports, which contain packages which should not be installed automatically, there is an extra line, e.g., "http://deb.debian.org/debian/dists/experimental/main/b
as follows.

```
Archive: experimental
Origin: Debian
Label: Debian
NotAutomatic: yes
Component: main
Architecture: amd64
```

Please note that for normal archives without "`NotAutomatic: yes`", the default Pin-Priority value is 500, while for special archives with "`NotAutomatic: yes`", the default Pin-Priority value is 1 (see apt_preferences(5) and Section 2.7.7).

2.5.4 Fetching of the meta data for the package

When APT tools, such as `aptitude`, `apt-get`, `synaptic`, `apt-file`, `auto-apt`, ... are used, we need to update the local copies of the meta data containing the Debian archive information. These local copies have following file names corresponding to the specified `distribution`, `area`, and `architecture` names in **the source list** (see Section 2.1.5).

- "`/var/lib/apt/lists/deb.debian.org_debian_dists_`*`distribution`*`_Release`"
- "`/var/lib/apt/lists/deb.debian.org_debian_dists_`*`distribution`*`_Release.gpg`"
- "`/var/lib/apt/lists/deb.debian.org_debian_dists_`*`distribution_area`*`_binary-`*`architecture`*`_Pac`
- "`/var/lib/apt/lists/deb.debian.org_debian_dists_`*`distribution_area`*`_source_Sources`"
- "`/var/cache/apt/apt-file/deb.debian.org_debian_dists_`*`distribution`*`_Contents-`*`architecture`*`.g`
  (for `apt-file`)

First 4 types of files are shared by all the pertinent APT commands and updated from command line by "`apt-get update`" or "`aptitude update`". The "`Packages`" meta data are updated if the "`deb`" is specified in **the source list**. The "`Sources`" meta data are updated if the "`deb-src`" is specified in **the source list**.

The "`Packages`" and "`Sources`" meta data contain "`Filename:`" stanza pointing to the file location of the binary and source packages. Currently, these packages are located under the "`pool/`" directory tree for the improved transition over the releases.

Local copies of "`Packages`" meta data can be interactively searched with the help of `aptitude`. The specialized search command grep-dctrl(1) can search local copies of "`Packages`" and "`Sources`" meta data.

Local copy of "`Contents-`*`architecture`*" meta data can be updated by "`apt-file update`" and its location is different from other 4 ones. See apt-file(1). (The `auto-apt` uses different location for local copy of "`Contents-`*`architectur`*
as default.)

2.5.5 The package state for APT

In addition to the remotely fetched meta data, the APT tool after `lenny` stores its locally generated installation state information in the "`/var/lib/apt/extended_states`" which is used by all APT tools to track all auto installed packages.

2.5.6 The package state for aptitude

In addition to the remotely fetched meta data, the `aptitude` command stores its locally generated installation state information in the "`/var/lib/aptitude/pkgstates`" which is used only by it.

2.5.7 Local copies of the fetched packages

All the remotely fetched packages via APT mechanism are stored in the "`/var/cache/apt/archives`" until they are cleaned.

This cache file cleaning policy for `aptitude` can be set under "`Options`" → "`Preferences`" and it may be forced by its menu "`Clean package cache`" or "`Clean obsolete files`" under "`Actions`".

2.5.8 Debian package file names

Debian package files have particular name structures.

package type	name structure
The binary package (a.k.a `deb`)	*`package-name_upstream-version-debian.revision_architec`
The binary package for debian-installer (a.k.a `udeb`)	*`package-name_upstream-version-debian.revision_architec`
The source package (upstream source)	*`package-name_upstream-version-debian.revision`*`.orig.tar`
The `1.0` source package (Debian changes)	*`package-name_upstream-version-debian.revision`*`.diff.gz`
The `3.0 (quilt)` source package (Debian changes)	*`package-name_upstream-version-debian.revision`*`.debian.ta`
The source package (description)	*`package-name_upstream-version-debian.revision`*`.dsc`

Table 2.15: The name structure of Debian packages

Tip
Here only the basic source package formats are described. See more on dpkg-source(1).

name component	usable characters (ERE regex)	existence
`package-name`	`[a-z0-9][-a-z0-9.+]+`	required
`epoch``:`	`[0-9]+:`	optional
`upstream-version`	`[-a-zA-Z0-9.+:]+`	required
`debian.revision`	`[a-zA-Z0-9.+~]+`	optional

Table 2.16: The usable characters for each component in the Debian package names

Note
You can check package version order by dpkg(1), e.g., "`dpkg --compare-versions 7.0 gt 7.~pre1 ; echo $?`".

Note
The debian-installer (d-i) uses `udeb` as the file extension for its binary package instead of normal `deb`. An `udeb` package is a stripped down `deb` package which removes few non-essential contents such as documentation to save space while relaxing the package policy requirements. Both `deb` and `udeb` packages share the same package structure. The "u" stands for micro.

2.5.9 The dpkg command

dpkg(1) is the lowest level tool for the Debian package management. This is very powerful and needs to be used with care.

While installing package called "*package_name*", `dpkg` process it in the following order.

1. Unpack the deb file ("`ar -x`" equivalent)
2. Execute "*package_name*`.preinst`" using debconf(1)
3. Install the package content to the system ("`tar -x`" equivalent)
4. Execute "*package_name*`.postinst`" using debconf(1)

The `debconf` system provides standardized user interaction with I18N and L10N (Chapter 8) supports.

file	description of contents
`/var/lib/dpkg/info/`*package_name*`.conffiles`	list of configuration files. (user modifiable)
`/var/lib/dpkg/info/`*package_name*`.list`	list of files and directories installed by the package
`/var/lib/dpkg/info/`*package_name*`.md5sums`	list of MD5 hash values for files installed by the package
`/var/lib/dpkg/info/`*package_name*`.preinst`	package script to be run before the package installation
`/var/lib/dpkg/info/`*package_name*`.postinst`	package script to be run after the package installation
`/var/lib/dpkg/info/`*package_name*`.prerm`	package script to be run before the package removal
`/var/lib/dpkg/info/`*package_name*`.postrm`	package script to be run after the package removal
`/var/lib/dpkg/info/`*package_name*`.config`	package script for `debconf` system
`/var/lib/dpkg/alternatives/`*package_name*	the alternative information used by the `update-alternatives` command
`/var/lib/dpkg/available`	the availability information for all the package
`/var/lib/dpkg/diversions`	the diversions information used by dpkg(1) and set by dpkg-divert(8)
`/var/lib/dpkg/statoverride`	the stat override information used by dpkg(1) and set by dpkg-statoverride(8)
`/var/lib/dpkg/status`	the status information for all the packages
`/var/lib/dpkg/status-old`	the first-generation backup of the "`var/lib/dpkg/status`" file
`/var/backups/dpkg.status*`	the second-generation backup and older ones of the "`var/lib/dpkg/status`" file

Table 2.17: The notable files created by `dpkg`

The "`status`" file is also used by the tools such as dpkg(1), "`dselect update`" and "`apt-get -u dselect-upgrade`".

The specialized search command grep-dctrl(1) can search the local copies of "`status`" and "`available`" meta data.

Tip
In the debian-installer environment, the `udpkg` command is used to open `udeb` packages. The `udpkg` command is a stripped down version of the `dpkg` command.

2.5.10 The update-alternatives command

The Debian system has mechanism to install somewhat overlapping programs peacefully using update-alternatives(1). For example, you can make the `vi` command select to run `vim` while installing both `vim` and `nvi` packages.

```
$ ls -l $(type -p vi)
lrwxrwxrwx 1 root root 20 2007-03-24 19:05 /usr/bin/vi -> /etc/alternatives/vi
$ sudo update-alternatives --display vi
...
$ sudo update-alternatives --config vi
  Selection    Command
 -----------------------------------------------
      1        /usr/bin/vim
*+    2        /usr/bin/nvi

Enter to keep the default[*], or type selection number: 1
```

The Debian alternatives system keeps its selection as symlinks in "`/etc/alternatives/`". The selection process uses corresponding file in "`/var/lib/dpkg/alternatives/`".

2.5.11 The dpkg-statoverride command

Stat overrides provided by the dpkg-statoverride(8) command are a way to tell dpkg(1) to use a different owner or mode for a **file** when a package is installed. If "`--update`" is specified and file exists, it is immediately set to the new owner and mode.



Caution
The direct alteration of owner or mode for a **file** owned by the package using `chmod` or `chown` commands by the system administrator is reset by the next upgrade of the package.

Note
I use the word **file** here, but in reality this can be any filesystem object that `dpkg` handles, including directories, devices, etc.

2.5.12 The dpkg-divert command

File **diversions** provided by the dpkg-divert(8) command are a way of forcing dpkg(1) not to install a file into its default location, but to a **diverted** location. The use of `dpkg-divert` is meant for the package maintenance scripts. Its casual use by the system administrator is deprecated.

2.6 Recovery from a broken system

When running `testing` or `unstable` system, the administrator is expected to recover from broken package management situation.



Caution
Some methods described here are high risk actions. You have been warned!

2.6.1 Incompatibility with old user configuration

If a desktop GUI program experienced instability after significant upstream version upgrade, you should suspect interference with old local configuration files created by it. If it is stable under a newly created user account, this hypothesis is confirmed. (This is a bug of packaging and usually avoided by the packager.)

To recover stability, you should move corresponding local configuration files and restart the GUI program. You may need to read old configuration file contents to recover configuration information later. (Do not erase them too quickly.)

2.6.2 Caching errors of the package data

Caching errors of the package data cause intriguing errors, such as "GPG error: ... invalid: BADSIG ..." with APT.

You should remove all cached data by "`sudo rm -rf /var/lib/apt/*` " and try again. (If `apt-cacher-ng` is used, you should also run "`sudo rm -rf /var/cache/apt-cacher-ng/*` ".)

2.6.3 Rescue with the dpkg command

Since `dpkg` is very low level package tool, it can function without network connection.

Let's assume `foo` package was broken and needs to be fixed.

You may still find cached copies of older bug free version of `foo` package in the package cache directory: "`/var/cache/apt`
(If not, you can download it from archive of https://snapshot.debian.org/ or copy it from package cache of a functioning machine.)

If you can boot the system, you may install it by the following command.

```
# dpkg -i /path/to/foo_old_version_arch.deb
```

If attempting to install a package this way fails due to some dependency violations and you really need to do this as the last resort, you can override dependency using `dpkg`'s "`--ignore-depends`", "`--force-depends`" and other options. If you do this, you need to make serious effort to restore proper dependency later. See dpkg(8) for details.

Note
If your system is seriously broken, you should make a full backup of system to a safe place (see Section 10.2) and should perform a clean installation. This is less time consuming and produces better results in the end.

Tip
If system breakage is minor, you may alternatively downgrade the whole system as in Section 2.7.11 using the higher level APT system.

2.6.4 Failed installation due to missing dependencies

If you force to install a package by "`sudo dpkg -i ...`" to a system without all dependency packages installed, the package installation will fail as partially installed.

You should install all dependency packages by repeatedly using "`sudo dpkg -i ...`" or by using:

```
# apt --fix-broken install
```

Then, configure all partially installed packages with the following command.

```
# dpkg --configure -a
```

2.6.5 Different packages with overlapped files

Archive level package management systems, such as aptitude(8) or apt-get(1), do not even try to install packages with overlapped files using package dependencies (see Section 2.1.7).

Errors by the package maintainer or deployment of inconsistently mixed source of archives (see Section 2.7.6) by the system administrator may create a situation with incorrectly defined package dependencies. When you install a package with overlapped files using aptitude(8) or apt-get(1) under such a situation, dpkg(1) which unpacks package ensures to return error to the calling program without overwriting existing files.



Caution
The use of third party packages introduces significant system risks via maintainer scripts which are run with root privilege and can do anything to your system. The dpkg(1) command only protects against overwriting by the unpacking.

You can work around such broken installation by removing the old offending package, *`old-package`*, first.

```
$ sudo dpkg -P old-package
```

2.6.6 Fixing broken package script

When a command in the package script returns error for some reason and the script exits with error, the package management system aborts their action and ends up with partially installed packages. When a package contains bugs in its removal scripts, the package may become impossible to remove and quite nasty.

For the package script problem of "*`package_name`*", you should look into following package scripts.

- "`/var/lib/dpkg/info/`*`package_name`*`.preinst`"
- "`/var/lib/dpkg/info/`*`package_name`*`.postinst`"
- "`/var/lib/dpkg/info/`*`package_name`*`.prerm`"
- "`/var/lib/dpkg/info/`*`package_name`*`.postrm`"

Edit the offending package script from the root using following techniques.

- disable the offending line by preceding "#"
- force to return success by appending the offending line with "`|| true`"

Then, configure all partially installed packages with the following command.

```
# dpkg --configure -a
```

2.6.7 Recovering package selection data

If "`/var/lib/dpkg/status`" becomes corrupt for any reason, the Debian system loses package selection data and suffers severely. Look for the old "`/var/lib/dpkg/status`" file at "`/var/lib/dpkg/status-old`" or "`/var/backups/`".

Keeping "`/var/backups/`" in a separate partition may be a good idea since this directory contains lots of important system data.

For serious breakage, I recommend to make fresh re-install after making backup of the system. Even if everything in "`/var/`" is gone, you can still recover some information from directories in "`/usr/share/doc/`" to guide your new installation.

Reinstall minimal (desktop) system.

```
# mkdir -p /path/to/old/system
```

Mount old system at "/path/to/old/system/".

```
# cd /path/to/old/system/usr/share/doc
# ls -1 >~/ls1.txt
# cd /usr/share/doc
# ls -1 >>~/ls1.txt
# cd
# sort ls1.txt | uniq | less
```

Then you are presented with package names to install. (There may be some non-package names such as "texmf".)

2.7 Tips for the package management

For simplicity, **the source list** examples in this section are presented as "/etc/apt/sources.list" in one-line-style after the trixie release.

2.7.1 Who uploaded the package?

Although the maintainer name listed in "/var/lib/dpkg/available" and "/usr/share/doc/package_name/change provide some information on "who is behind the packaging activity", the actual uploader of the package is somewhat obscure. who-uploads(1) in the devscripts package identifies the actual uploader of Debian source packages.

2.7.2 Limiting download bandwidth for APT

If you want to limit the download bandwidth for APT to e.g. 800Kib/sec (=100kiB/sec), you should configure APT with its configuration parameter as the following.

```
APT::Acquire::http::Dl-Limit "800";
```

2.7.3 Automatic download and upgrade of packages

The apt package comes with its own cron script "/etc/cron.daily/apt" to support the automatic download of packages. This script can be enhanced to perform the automatic upgrade of packages by installing the unattended-upgrad package. These can be customized by parameters in "/etc/apt/apt.conf.d/02backup" and "/etc/apt/apt.conf.d as described in "/usr/share/doc/unattended-upgrades/README".

The unattended-upgrades package is mainly intended for the security upgrade for the stable system. If the risk of breaking an existing stable system by the automatic upgrade is smaller than that of the system broken by the intruder using its security hole which has been closed by the security update, you should consider using this automatic upgrade with configuration parameters as the following.

```
APT::Periodic::Update-Package-Lists "1";
APT::Periodic::Download-Upgradeable-Packages "1";
APT::Periodic::Unattended-Upgrade "1";
```

If you are running an testing or unstable system, you do not want to use the automatic upgrade since it certainly breaks system some day. Even for such testing or unstable case, you may still want to download packages in advance to save time for the interactive upgrade with configuration parameters as the following.

```
APT::Periodic::Update-Package-Lists "1";
APT::Periodic::Download-Upgradeable-Packages "1";
APT::Periodic::Unattended-Upgrade "0";
```

2.7.4 Updates and Backports

There are stable-updates ("`trixie`-updates" during the `trixie`-as-`stable` release cycle) and backports.debian.org archives which provide upgrade packages for `stable`.

In order to use these archives, you list all required archives in the "`/etc/apt/sources.list`" file as the following.

```
deb http://deb.debian.org/debian/ trixie main non-free-firmware contrib non-free
deb http://security.debian.org/debian-security trixie-security main non-free-firmware ↩
    contrib non-free
deb http://deb.debian.org/debian/ trixie-updates main non-free-firmware contrib non-free
deb http://deb.debian.org/debian/ trixie-backports main non-free-firmware contrib non-free
```

There is no need to set Pin-Priority value explicitly in the "`/etc/apt/preferences`" file. When newer packages become available, the default configuration provides most reasonable upgrades (see Section 2.5.3).

- All installed older packages are upgraded to newer ones from `trixie-updates`.
- Only manually installed older packages from `trixie-backports` are upgraded to newer ones from `trixie-backports`

Whenever you wish to install a package named "*`package-name`*" with its dependency from `trixie-backports` archive manually, you use following command while switching target release with "`-t`" option.

```
$ sudo apt-get install -t trixie-backports package-name
```



Warning
Do not install too many packages from backports.debian.org archives. It may cause package dependency complications. See Section 2.1.11 for alternative solutions.

2.7.5 External package archives



Warning
You should be aware that the external package gains the root privilege to your system. You should only use the trusted external package archive. See Section 2.1.11 for alternative solutions.

You can use secure APT with Debian-compatible external package archive by adding it to **the source list** and its archive key file into the "`/etc/apt/trusted.gpg.d/`" directory. See sources.list(5), apt-secure(8) and apt-key(8).

2.7.6 Packages from mixed source of archives without apt-pinning



Caution
Installing packages from mixed source of archives is not supported by the official Debian distribution except for officially supported particular combinations of archives such as `stable` with security updates and stable-updates.

Here is an example of operations to include specific newer upstream version packages found in `unstable` while tracking `testing` for single occasion.

1. Change the "`/etc/apt/sources.list`" file temporarily to single "`unstable`" entry.
2. Run "`aptitude update`".
3. Run "`aptitude install package-name`".
4. Recover the original "`/etc/apt/sources.list`" file for `testing`.
5. Run "`aptitude update`".

You do not create the "`/etc/apt/preferences`" file nor need to worry about **apt-pinning** with this manual approach. But this is very cumbersome.



Caution
When using mixed source of archives, you must ensure compatibility of packages by yourself since the Debian does not guarantee it. If package incompatibility exists, you may break system. You must be able to judge these technical requirements. The use of mixed source of random archives is completely optional operation and its use is not something I encourage you to use.

General rules for installing packages from different archives are the following.

- Non-binary packages ("`Architecture: all`") are **safer** to install.
  - documentation packages: no special requirements
  - interpreter program packages: compatible interpreter must be available
- Binary packages (non "`Architecture: all`") usually face many road blocks and are **unsafe** to install.
  - library version compatibility (including "`libc`")
  - related utility program version compatibility
  - Kernel ABI compatibility
  - C++ ABI compatibility
  - …

Note
In order to make a package to be **safer** to install, some commercial non-free binary program packages may be provided with completely statically linked libraries. You should still check ABI compatibility issues etc. for them.

Note
Except to avoid broken package for a short term, installing binary packages from non-Debian archives is generally bad idea. You should seek all available alternative safer technical solutions which are compatible with your current Debian syetem (see Section 2.1.11).

2.7.7 Tweaking candidate version with apt-pinning



Warning
Use of **apt-pinning** technique by a novice user is sure call for major troubles. You must avoid using this technique except when you absolutely need it. See Section 2.1.11 for alternative solutions.

Without the "`/etc/apt/preferences`" file, APT system chooses the latest available version as the **candidate version** using the version string. This is the normal state and most recommended usage of APT system. All officially supported combinations of archives do not require the "`/etc/apt/preferences`" file since some archives which should not be used as the automatic source of upgrades are marked as **NotAutomatic** and dealt properly.

Tip
The version string comparison rule can be verified with, e.g., "`dpkg --compare-versions ver1.1 gt ver1.1~1; echo $?`" (see dpkg(1)).

When you install packages from mixed source of archives (see Section 2.7.6) regularly, you can automate these complicated operations by creating the "`/etc/apt/preferences`" file with proper entries and tweaking the package selection rule for **candidate version** as described in apt_preferences(5). This is called **apt-pinning**.

When using **apt-pinning**, you must ensure compatibility of packages by yourself since the Debian does not guarantee it. The **apt-pinning** is completely optional operation and its use is not something I encourage you to use.

Archive level Release files (see Section 2.5.3) are used for the rule of apt_preferences(5). Thus **apt-pinning** works only with "suite" name for normal Debian archives and security Debian archives. (This is different from Ubuntu archives.) For example, you can do "`Pin: release a=unstable`" but can not do "`Pin: release a=sid`" in the "`/etc/apt/preferences`" file.

When you use non-Debian archive as a part of **apt-pinning**, you should check what they are intended for and also check their credibility. For example, Ubuntu and Debian are not meant to be mixed.

Note
Even if you do not create the "`/etc/apt/preferences`" file, you can do fairly complex system operations (see Section 2.6.3 and Section 2.7.6) without **apt-pinning**.

Here is a simplified explanation of **apt-pinning** technique.

The APT system chooses the highest Pin-Priority **upgrading** package from available package sources defined in the "`/etc/apt/sources.list`" file as the **candidate version** package. If the Pin-Priority of the package is larger than 1000, this version restriction for **upgrading** is dropped to enable downgrading (see Section 2.7.11).

Pin-Priority value of each package is defined by "Pin-Priority" entries in the "`/etc/apt/preferences`" file or uses its default value.

Pin-Priority	apt-pinning effects to the package
1001	install the package even if this constitutes a downgrade of the package
990	used as the default for the **target release** archive
500	used as the default for the **normal** archive
100	used as the default for the **NotAutomatic** and **ButAutomaticUpgrades** archive
100	used for the **installed** package
1	used as the default for the **NotAutomatic** archive
-1	**never install** the package even if recommended

Table 2.18: List of notable Pin-Priority values for **apt-pinning** technique.

The **target release** archive can be set by the command line option, e.g., "`apt-get install -t testing some-packag`

The **NotAutomatic** and **ButAutomaticUpgrades** archive is set by archive server having its archive level Release file (see Section 2.5.3) containing both "`NotAutomatic: yes`" and "`ButAutomaticUpgrades: yes`". The **NotAutomatic** archive is set by archive server having its archive level Release file containing only "`NotAutomatic: yes`".

The **apt-pinning situation** of *package* from multiple archive sources is displayed by "`apt-cache policy` *package*".

- A line started with "`Package pin:`" lists the package version of **pin** if association just with *package* is defined, e.g., "`Package pin: 0.190`".
- No line with "`Package pin:`" exists if no association just with *package* is defined.
- The Pin-Priority value associated just with *package* is listed right side of all version strings, e.g., "`0.181 700`".
- "`0`" is listed right side of all version strings if no association just with *package* is defined, e.g., "`0.181 0`".
- The Pin-Priority values of archives (defined as "`Package: *`" in the "`/etc/apt/preferences`" file) are listed left side of all archive paths, e.g., "`100 http://deb.debian.org/debian/ trixie-backports/main Packages`".

2.7.8 Blocking packages installed by "Recommends"



Warning
Use of **apt-pinning** technique by a novice user is sure call for major troubles. You must avoid using this technique except when you absolutely need it. See Section 2.1.11 for alternative solutions.

If you wish not to pull in particular packages automatically by "Recommends", you must create the "`/etc/apt/preferences`" file and explicitly list all those packages at the top of it as the following.

```
Package: package-1
Pin: version *
Pin-Priority: -1

Package: package-2
Pin: version *
Pin-Priority: -1
```

2.7.9 Tracking `testing` with some packages from `unstable`



Warning
Use of **apt-pinning** technique by a novice user is sure call for major troubles. You must avoid using this technique except when you absolutely need it. See Section 2.1.11 for alternative solutions.

Here is an example of **apt-pinning** technique to include specific newer upstream version packages found in `unstable` regularly upgraded while tracking `testing`. You list all required archives in the "`/etc/apt/sources.list`" file as the following.

```
deb http://deb.debian.org/debian/ testing main contrib non-free
deb http://deb.debian.org/debian/ unstable main contrib non-free
deb http://security.debian.org/debian-security testing-security main contrib
```

Set the "`/etc/apt/preferences`" file as the following.

```
Package: *
Pin: release a=unstable
Pin-Priority: 100
```

When you wish to install a package named "*`package-name`*" with its dependencies from `unstable` archive under this configuration, you issue the following command which switches target release with "`-t`" option (Pin-Priority of `unstable` becomes 990).

```
$ sudo apt-get install -t unstable package-name
```

With this configuration, usual execution of "`apt-get upgrade`" and "`apt-get dist-upgrade`" (or "`aptitude safe-upgrade`" and "`aptitude full-upgrade`") upgrades packages which were installed from `testing` archive using current `testing` archive and packages which were installed from `unstable` archive using current `unstable` archive.



Caution
Be careful not to remove "`testing`" entry from the "`/etc/apt/sources.list`" file. Without "`testing`" entry in it, APT system upgrades packages using newer `unstable` archive.

Tip
I usually edit the "`/etc/apt/sources.list`" file to comment out "`unstable`" archive entry right after above operation. This avoids slow update process of having too many entries in the "`/etc/apt/sources.list`" file although this prevents upgrading packages which were installed from `unstable` archive using current `unstable` archive.

Tip
If "`Pin-Priority: 1`" is used instead of "`Pin-Priority: 100`" in the "`/etc/apt/preferences`" file, already installed packages having Pin-Priority value of 100 are not upgraded by `unstable` archive even if "`testing`" entry in the "`/etc/apt/sources.list`" file is removed.

If you wish to track particular packages in `unstable` automatically without initial "`-t unstable`" installation, you must create the "`/etc/apt/preferences`" file and explicitly list all those packages at the top of it as the following.

```
Package: package-1
Pin: release a=unstable
Pin-Priority: 700

Package: package-2
Pin: release a=unstable
Pin-Priority: 700
```

These set Pin-Priority value for each specific package. For example, in order to track the latest `unstable` version of this "Debian Reference" in English, you should have following entries in the "`/etc/apt/preferences`" file.

```
Package: debian-reference-en
Pin: release a=unstable
Pin-Priority: 700

Package: debian-reference-common
Pin: release a=unstable
Pin-Priority: 700
```

Tip
This **apt-pinning** technique is valid even when you are tracking `stable` archive. Documentation packages have been always safe to install from `unstable` archive in my experience, so far.

2.7.10 Tracking `unstable` with some packages from `experimental`



Warning
Use of **apt-pinning** technique by a novice user is sure call for major troubles. You must avoid using this technique except when you absolutely need it. See Section 2.1.11 for alternative solutions.

Here is another example of **apt-pinning** technique to include specific newer upstream version packages found in `experimental` while tracking `unstable`. You list all required archives in the "`/etc/apt/sources.list`" file as the following.

```
deb http://deb.debian.org/debian/ unstable main contrib non-free
deb http://deb.debian.org/debian/ experimental main contrib non-free
deb http://security.debian.org/ testing-security main contrib
```

The default Pin-Priority value for `experimental` archive is always 1 (<<100) since it is **NotAutomatic** archive (see Section 2.5.3). There is no need to set Pin-Priority value explicitly in the "`/etc/apt/preferences`" file just to use `experimental` archive unless you wish to track particular packages in it automatically for next upgrading.

2.7.11 Emergency downgrading



Warning
Use of **apt-pinning** technique by a novice user is sure call for major troubles. You must avoid using this technique except when you absolutely need it. See Section 2.1.11 for alternative solutions.



Caution
Downgrading is not officially supported by the Debian by design. It should be done only as a part of emergency recovery process. Despite of this situation, it is known to work well in many incidents. For critical systems, you should backup all important data on the system after the recovery operation and re-install the new system from the scratch.

You may be lucky to downgrade from newer archive to older archive to recover from broken system upgrade by manipulating **candidate version** (see Section 2.7.7). This is lazy alternative to tedious actions of many "`dpkg -i` *`broken-package_old-version`*`.deb`" commands (see Section 2.6.3).

Search lines in the "`/etc/apt/sources.list`" file tracking `unstable` as the following.

```
deb http://deb.debian.org/debian/ sid main contrib non-free
```

Replace it with the following to track `testing`.

```
deb http://deb.debian.org/debian/ forky main contrib non-free
```

Set the "`/etc/apt/preferences`" file as the following.

```
Package: *
Pin: release a=testing
Pin-Priority: 1010
```

Run "`apt-get update; apt-get dist-upgrade`" to force downgrading of packages across the system.

Remove this special "`/etc/apt/preferences`" file after this emergency downgrading.

Tip
It is a good idea to remove (not purge!) as much packages to minimize dependency problems. You may need to manually remove and install some packages to get system downgraded. Linux kernel, bootloader, udev, PAM, APT, and networking related packages and their configuration files require special attention.

2.7.12 The equivs package

If you are to compile a program from source to replace the Debian package, it is best to make it into a real local debianized package (`*.deb`) and use private archive.

If you choose to compile a program from source and to install them under "`/usr/local`" instead, you may need to use `equivs` as a last resort to satisfy the missing package dependency.

```
Package: equivs
Priority: optional
Section: admin
Description: Circumventing Debian package dependencies
 This package provides a tool to create trivial Debian packages.
 Typically these packages contain only dependency information, but they
 can also include normal installed files like other packages do.
 .
 One use for this is to create a metapackage: a package whose sole
 purpose is to declare dependencies and conflicts on other packages so
 that these will be automatically installed, upgraded, or removed.
 .
 Another use is to circumvent dependency checking: by letting dpkg
 think a particular package name and version is installed when it
 isn't, you can work around bugs in other packages' dependencies.
 (Please do still file such bugs, though.)
```

2.7.13 Porting a package to the stable system



Caution
There is no guarantee for the procedure described here to work without extra manual efforts for system differences.

For partial upgrades of the `stable` system, rebuilding a package within its environment using the source package is desirable. This avoids massive package upgrades due to their dependencies.

Add the following entries to the "`/etc/apt/sources.list`" of a `stable` system.

```
deb-src http://deb.debian.org/debian unstable  main contrib non-free
```

Install required packages for the compilation and download the source package as the following.

```
# apt-get update
# apt-get dist-upgrade
# apt-get install fakeroot devscripts build-essential
# apt-get build-dep foo
$ apt-get source foo
$ cd foo*
```

Update some tool chain packages such as `dpkg`, and `debhelper` from the backport packages if they are required for the backporting.

Execute the following.

```
$ dch -i
```

Bump package version, e.g. one appended with "+bp1" in "debian/changelog"

Build packages and install them to the system as the following.

```
$ debuild
$ cd ..
# debi foo*.changes
```

2.7.14 Proxy server for APT

Since mirroring whole subsection of Debian archive wastes disk space and network bandwidth, deployment of a local proxy server for APT is desirable consideration when you administer many systems on LAN. APT can be configure to use generic web (http) proxy servers such as squid (see Section 6.5) as described in apt.conf(5) and in "/usr/share/doc/apt/examples/configure-index.gz". The "$http_proxy" environment variable can be used to override proxy server setting in the "/etc/apt/apt.conf" file.

There are proxy tools specially for Debian archive. You should check BTS before using them.

package	popcon	size	description
apt-cacher	V:0.28, I:0.33	265	Caching proxy for Debian package and source files (Perl program)
apt-cacher-ng	V:3.9, I:4.1	2063	Caching proxy for distribution of software packages (compiled C++ program)

Table 2.19: List of the proxy tools specially for Debian archive



Caution
When Debian reorganizes its archive structure, these specialized proxy tools tend to require code rewrites by the package maintainer and may not be functional for a while. On the other hand, generic web (http) proxy servers are more robust and easier to cope with such changes.

2.7.15 More readings for the package management

You can learn more on the package management from following documentations.

- Primary documentations on the package management:
  - aptitude(8), dpkg(1), tasksel(8), apt(8), apt-get(8), apt-config(8), apt-secure(8), sources.list(5), apt.conf(5), and apt_preferences(5);
  - "/usr/share/doc/apt-doc/guide.html/index.html" and "/usr/share/doc/apt-doc/offline.html/in from the apt-doc package; and
  - "/usr/share/doc/aptitude/html/en/index.html" from the aptitude-doc-en package.
- Official and detailed documentations on the Debian archive:
  - "Debian Policy Manual Chapter 2 - The Debian Archive",
  - "Debian Developer's Reference, Chapter 4 - Resources for Debian Developers 4.6 The Debian archive", and
  - "The Debian GNU/Linux FAQ, Chapter 6 - The Debian FTP archives".
- Tutorial for building of a Debian package for Debian users:
  - "Guide for Debian Maintainers".

Chapter 3

The system initialization

It is wise for you as the system administrator to know roughly how the Debian system is started and configured. Although the exact details are in the source files of the packages installed and their documentations, it is a bit overwhelming for most of us.

Here is a rough overview of the key points of the Debian system initialization. Since the Debian system is a moving target, you should refer to the latest documentation.

- Debian Linux Kernel Handbook is the primary source of information on the Debian kernel.
- bootup(7) describes the system bootup process based on `systemd` . (Recent Debian)
- boot(7) describes the system bootup process based on UNIX System V Release 4. (Older Debian)

3.1 An overview of the boot strap process

The computer system undergoes several phases of boot strap processes from the power-on event until it offers the fully functional operating system (OS) to the user.

For simplicity, I limit discussion to the typical PC platform with the default installation.

The typical boot strap process is like a four-stage rocket. Each stage rocket hands over the system control to the next stage one.

- Section 3.1.1
- Section 3.1.2
- Section 3.1.3
- Section 3.1.4

Of course, these can be configured differently. For example, if you compiled your own kernel, you may be skipping the step with the mini-Debian system. So please do not assume this is the case for your system until you check it yourself.

3.1.1 Stage 1: the UEFI

The Unified Extensible Firmware Interface (UEFI) defines a boot manager as part of the UEFI specification. When a computer is powered on, the boot manager is the 1st stage of the boot process which checks the boot configuration and based on its settings, then executes the specified OS boot loader or operating system kernel (usually boot loader).

The boot configuration is defined by variables stored in NVRAM, including variables that indicate the file system paths to OS loaders or OS kernels.

An EFI system partition (ESP) is a data storage device partition that is used in computers adhering to the UEFI specification. Accessed by the UEFI firmware when a computer is powered up, it stores UEFI applications and the files these applications need to run, including operating system boot loaders. (On the legacy PC system, BIOS stored in the MBR may be used instead.)

3.1.2 Stage 2: the boot loader

The boot loader is the 2nd stage of the boot process which is started by the UEFI. It loads the system kernel image and the initrd image to the memory and hands control over to them. This initrd image is the root filesystem image and its support depends on the bootloader used.

The Debian system normally uses the Linux kernel as the default system kernel. The initrd image for the current 5.x Linux kernel is technically the initramfs (initial RAM filesystem) image.

There are many boot loaders and configuration options available.

package	popcon	size	initrd	bootloader	description
grub-efi-amd64	I:466	142	Supported	GRUB UEFI	This is smart enough to understand disk partitions and filesystems such as vfat, ext4, ... (UEFI)
grub-pc	V:16, I:512	479	Supported	GRUB 2	This is smart enough to understand disk partitions and filesystems such as vfat, ext4, ... (BIOS)
grub-rescue-pc	V:0.05, I:0.58	7269	Supported	GRUB 2	This is GRUB 2 bootable rescue images (CD and floppy) (PC/BIOS version)
grml-rescueboot	V:0.1, I:1.0	39	N/A	GRUB 2 plug-in	grml-rescueboot adds live Linux ISO images to the grub2 boot menu
syslinux	V:3, I:31	325	Supported	Isolinux	This understands the ISO9660 filesystem. This is used by the boot CD.
syslinux	V:3, I:31	325	Supported	Syslinux	This understands the MSDOS filesystem (FAT). This is used by the boot floppy.
loadlin	V:0.03, I:0.39	87	Supported	Loadlin	New system is started from the FreeDOS/MSDOS system.
mbr	V:0.2, I:2.7	47	Not supported	MBR by Neil Turton	This is free software which substitutes MSDOS MBR. This only understands disk partitions.

Table 3.1: List of boot loaders

For UEFI system, GRUB2 first reads the ESP partition and uses UUID specified for `search.fs_uuid` in "`/boot/efi/EFI/`" to determine the partition of the GRUB2 menu configuration file "`/boot/grub/grub.cfg`".

The key part of the GRUB2 menu configuration file looks like:

```
menuentry 'Debian GNU/Linux' ... {
        load_video
        insmod gzio
        insmod part_gpt
        insmod ext2
        search --no-floppy --fs-uuid --set=root fe3e1db5-6454-46d6-a14c-071208ebe4b1
        echo    'Loading Linux 5.10.0-6-amd64 ...'
```

```
        linux   /boot/vmlinuz-5.10.0-6-amd64 root=UUID=fe3e1db5-6454-46d6-a14c-071208ebe4b1 ↩
            ro quiet
        echo    'Loading initial ramdisk ...'
        initrd  /boot/initrd.img-5.10.0-6-amd64
}
```

For this part of `/boot/grub/grub.cfg`, this menu entry means the following.

setting	value
GRUB2 modules loaded	`gzio`, `part_gpt`, `ext2`
root file system partition used	partition identified by `UUID=fe3e1db5-6454-46d6-a14c-071208ebe4b1`
kernel image path in the root file system	`/boot/vmlinuz-5.10.0-6-amd64`
kernel boot parameter used	"`root=UUID=fe3e1db5-6454-46d6-a14c-071208ebe4b1 ro quiet`"
initrd image path in the root file system	`/boot/initrd.img-5.10.0-6-amd64`

Table 3.2: The meaning of the menu entry of the above part of `/boot/grub/grub.cfg`

On Debian system, "`/boot/grub/grub.cfg`" is managed by the installed GRUB package (e.g. `grub-efi-amd64`) and direct user modification to this file is deprecated. You should customize configuration files in "`/etc/grub.d/`" and "`/etc/default/grub`", instead. Then configure the GRUB configuration files and update NVRAM variables to automatically boot into Debian by:

```
# dpkg-reconfigure grub-efi-amd64
```

Tip
You can show kernel boot log messages by removing "`quiet`" from the "`GRUB_CMDLINE_LINUX_DEFAULT`" value in "`/etc/default/grub`".

Tip
You can add GRUB splash background by placing its image file in "`/boot/grub/`".

See "`info grub`", grub-install(8), grub-mkconfig(8).

3.1.3 Stage 3: the mini-Debian system

The mini-Debian system is the 3rd stage of the boot process which is started by the boot loader. It runs the system kernel with its root filesystem on the memory. This is an optional preparatory stage of the boot process.

Note
The term "the mini-Debian system" is coined by the author to describe this 3rd stage boot process for this document. This system is commonly referred as the initrd or initramfs system. Similar system on the memory is used by the Debian Installer.

The "`/init`" program is executed as the first program in this root filesystem on the memory. It is a program which initializes the kernel in user space and hands control over to the next stage. This mini-Debian system offers flexibility to the boot process such as adding kernel modules before the main boot process or mounting the root filesystem as an encrypted one.

- The "`/init`" program is a shell script program if initramfs was created by `initramfs-tools`.
  - You can interrupt this part of the boot process to gain root shell by providing "`break=init`" etc. to the kernel boot parameter. See the "`/init`" script for more break conditions. This shell environment is sophisticated enough to make a good inspection of your machine's hardware.
  - Commands available in this mini-Debian system are stripped down ones and mainly provided by a GNU tool called busybox(1).
- The "`/init`" program is a binary `systemd` program if initramfs was created by `dracut`.
  - Commands available in this mini-Debian system are stripped down systemd(1) environment.



Caution
You need to use "`-n`" option for `mount` command when you are on the readonly root filesystem.

3.1.4 Stage 4: the normal Debian system

The normal Debian system is the 4th stage of the boot process which is started by the mini-Debian system. The system kernel for the mini-Debian system continues to run in this environment. The root filesystem is switched from the one on the memory to the one on the physical storage device.

The init program is executed as the first program with PID=1 to perform the main boot process of starting many programs. The default file path for the init program is "`/usr/sbin/init`" but it can be changed by the kernel boot parameter as "`init=/path/to/init_program`".

"`/usr/sbin/init`" is symlinked to "`/lib/systemd/systemd`" after Debian 8 Jessie (released in 2015).

Tip
The actual init command on your system can be verified by the "`ps --pid 1 -f`" command.

Tip
See Debian wiki: BootProcessSpeedup for the latest tips to speed up the boot process.

3.2 Rescue system



Warning
Do not perform system administration tasks around the boot strap process without having a rescue system.

Availability of a rescue system enables us to perform challenging tasks such as:

- Booting a system from a broken boot loader installation
- Fixing a broken boot loader installation
- Extracting data from a broken unbootable system

package	popcon	size	description
systemd	V:920, I:980	11386	event-based init(8) daemon for concurrency (alternative to sysvinit)
cloud-init	V:4.1, I:6.3	3267	initialization system for infrastructure cloud instances
systemd-sysv	V:915, I:983	76	the manual pages and links needed for systemd to replace sysvinit
init-system-helpers	V:723, I:988	132	helper tools for switching between sysvinit and systemd
initscripts	V:15, I:54	197	scripts for initializing and shutting down the system
sysvinit-core	V:3.1, I:3.6	365	System-V-like init(8) utilities
sysv-rc	V:29, I:57	85	System-V-like runlevel change mechanism
sysvinit-utils	V:725, I:1000	100	System-V-like utilities (startpar(8), bootlogd(8), …)
lsb-base	V:205, I:300	12	Linux Standard Base 3.2 init script functionality
insserv	V:31, I:57	132	tool to organize boot sequence using LSB init.d script dependencies
kexec-tools	V:1.4, I:5.3	320	kexec tool for kexec(8) reboots (warm reboot)
systemd-bootchart	V:0.17, I:0.52	131	boot process performance analyser
mingetty	V:0.2, I:2.3	36	console-only getty(8)
mgetty	V:0.13, I:0.47	318	smart modem getty(8) replacement

Table 3.3: List of boot utilities for the Debian system

- Editing filesystems, disk partitions, and LVM volumes involving the root filesystem

Typically, a rescue system is provided as a ISO image file and written to a removable storage media such as:

- USB flash drive prepared as Section 9.7.2
- CD / DVD prepared as Section 9.7.7

For simplicity, USB flash drive cases are mentioned as examples below but CD or DVD may be used as well.

Tip
You may need to change some UEFI NVRAM variables to boot arbitrary boot loaders on the removable storage media.

3.2.1 GRUB UEFI rescue system on USB

The GRUB UEFI rescue system with menu can be started by turning on system power with the ”GRUB UEFI rescue system on USB” inserted.

This ”GRUB UEFI rescue system on USB” is prepared by writing the Super Grub2 Disk ISO image to USB flash drive in advance.

In case of broken boot loader configuration by the installation of another operating system etc., you can fix this by:

- Start the GRUB UEFI rescue system to discover bootable installed systems automatically.
- Start the installed Debian system from the GRUB menu.
- At Linux root shell prompt:

```
# dpkg-reconfigure grub-efi-amd64
```

Note
The bootable GRUB rescue ISO image can be generated by following "`info grub-mkrescue`", too. It offers a CLI GRUB shell prompt but doesn't offer automatic discovery of bootable installed systems.

3.2.2 Linux live rescue system on USB

The Linux live rescue system can be started by turning on system power with the "Linux live rescue system on USB" inserted.

This "Linux live rescue system on USB" can be prepared by writing one of Linux live ISO images based on Debian to a USB flash drive in advance. Here are some examples of such Linux live images.

- Debian Live images
- Kali Linux Live
- Grml Live Linux

Here are some use cases of this Linux live rescue system:

- Fix the broken boot loader configuration caused by the installation of another operating system etc.:
  - Start the Linux live rescue system.
  - Mount the partition containing the root filesystem of the unbootable installed Debian system to "`/mnt`".
  - At Linux root shell prompt:

```
# chroot /mnt; dpkg-reconfigure grub-efi-amd64
```

- Fix the broken `dpkg` package:
  - Start the Linux live rescue system.
  - Mount the partition containing the root filesystem of the installed Debian system with the broken `dpkg` package to "`/mnt`".
  - At Linux root shell prompt:

```
# dpkg --root /mnt -i /mnt/var/cache/apt/archives/dpkg_old_version_amd64.deb
```

- Perform normally prohibited changes such as filesystem operations to the installed system (see Section 9.6).



Warning
Your GUI screen of the Linux live system may be locked after the inactivity.

Tip

- It's a good idea to set your password as soon as you start a Linux live system.
- You may set your password for example by "`sudo passwd` *`user`*" from the user's shell prompt on the Linux virtual consoles (see Section 1.1.6).
- Some Linux live systems may set their default "*user*/*password*": "Debian Live" = "user/live", "Kali Linux Live" = "kali/kali"

3.2.3 Linux live rescue system from GRUB

The Linux live rescue system can be started from the GRUB menu entry. The GRUB configuration for this is prepared by the following:

- Install the `grml-rescueboot` package
- Find a Linux live ISO image which has "`/boot/grub/loopback.cfg`" in its image.
  - ISO images mentioned in Section 3.2.2 have "`/boot/grub/loopback.cfg`" in their image.
- Copy some of these Linux live ISO images to the "`/boot/grml/`" directory.
- Update GRUB menu by:

```
# dpkg-reconfigure grub-efi-amd64
```

Upon turning on system power, the GRUB displays menu with candidates for Linux live rescue systems.

3.3 Systemd

3.3.1 Systemd init

When the Debian system starts, `/usr/sbin/init` symlinked to `/usr/lib/systemd` is started as the init system process (`PID=1`) owned by root (`UID=0`). See systemd(1).

The `systemd` init process spawns processes in parallel based on the unit configuration files (see systemd.unit(5)) which are written in declarative style instead of SysV-like procedural style.

The spawned processes are placed in individual Linux control groups named after the unit which they belong to in the private systemd hierarchy (see cgroups and Section 4.7.5).

Units for the system mode are loaded from the "System Unit Search Path" described in systemd.unit(5). The main ones are as follows in the order of priority:

- "`/etc/systemd/system/*`": System units created by the administrator
- "`/run/systemd/system/*`": Runtime units
- "`/lib/systemd/system/*`": System units installed by the distribution package manager

Their inter-dependencies are specified by the directives "`Wants=`", "`Requires=`", "`Before=`", "`After=`", ... (see "MAPPING OF UNIT PROPERTIES TO THEIR INVERSES" in systemd.unit(5)). The resource controls are also defined (see systemd.resource-control(5)).

The suffix of the unit configuration file encodes their types as:

- ***.service** describes the process controlled and supervised by `systemd`. See systemd.service(5).
- ***.device** describes the device exposed in the sysfs(5) as udev(7) device tree. See systemd.device(5).
- ***.mount** describes the file system mount point controlled and supervised by `systemd`. See systemd.mount(5).
- ***.automount** describes the file system auto mount point controlled and supervised by `systemd`. See systemd.automount(5
- ***.swap** describes the swap device or file controlled and supervised by `systemd`. See systemd.swap(5).
- ***.path** describes the path monitored by `systemd` for path-based activation. See systemd.path(5).
- ***.socket** describes the socket controlled and supervised by `systemd` for socket-based activation. See systemd.socket(5).

- ***.timer** describes the timer controlled and supervised by `systemd` for timer-based activation. See systemd.timer(5).
- ***.slice** manages resources with the cgroups(7). See systemd.slice(5).
- ***.scope** is created programmatically using the bus interfaces of `systemd` to manages a set of system processes. See systemd.scope(5).
- ***.target** groups other unit configuration files to create the synchronization point during start-up. See systemd.target(5).

Upon system start up (i.e., init), the `systemd` process tries to start the "`/lib/systemd/system/default.target` (normally symlinked to "`graphical.target`"). First, some special target units (see systemd.special(7)) such as "`local-fs.target`", "`swap.target`" and "`cryptsetup.target`" are pulled in to mount the filesystems. Then, other target units are also pulled in by the target unit dependencies. For details, read bootup(7).

`systemd` offers backward compatibility features. SysV-style boot scripts in "`/etc/init.d/rc[0123456S].d/[KS]`*name*" are still parsed and telinit(8) is translated into systemd unit activation requests.



Caution
Emulated runlevel 2 to 4 are all symlinked to the same "`multi-user.target`".

3.3.2 Systemd login

When a user logins to the Debian system via gdm3(8), sshd(8), etc., `/lib/systemd/system --user` is started as the user service manager process owned by the corresponding user. See systemd(1).

The `systemd` user service manager process spawns processes in parallel based on the declarative unit configuration files (see systemd.unit(5) and `user@.service`(5)).

Units for the user mode are loaded from the "User Unit Search Path" described in systemd.unit(5). The main ones are as follows in the order of priority:

- "`~/.config/systemd/user/*`": User configuration units
- "`/etc/systemd/user/*`": User units created by the administrator
- "`/run/systemd/user/*`": Runtime units
- "`/lib/systemd/user/*`": User units installed by the distribution package manager

These are managed in the same way as Section 3.3.1.

3.4 The kernel message

The kernel error message displayed to the console can be configured by setting its threshold level.

```
# dmesg -n3
```

error level value	error level name	meaning
0	KERN_EMERG	system is unusable
1	KERN_ALERT	action must be taken immediately
2	KERN_CRIT	critical conditions
3	KERN_ERR	error conditions
4	KERN_WARNING	warning conditions
5	KERN_NOTICE	normal but significant condition
6	KERN_INFO	informational
7	KERN_DEBUG	debug-level messages

Table 3.4: List of kernel error levels

Operation	Command snippets
View log for system services and kernel from the last boot	"`journalctl -b --system`"
View log for services of the current user from the last boot	"`journalctl -b --user`"
View job log of "`$unit`" from the last boot	"`journalctl -b -u $unit`"
View job log of "`$unit`" ("`tail -f`" style) from the last boot	"`journalctl -b -u $unit -f`"

Table 3.5: List of typical `journalctl` command snippets

3.5 The system message

Under `systemd`, both kernel and system messages are logged by the journal service `systemd-journald.service` (a.k.a `journald`) either into a persistent binary data below "`/var/log/journal`" or into a volatile binary data below "`/run/log/journal/`". These binary log data are accessed by the journalctl(1) command. For example, you can display log from the last boot as:

```
$ journalctl -b
```

Under `systemd`, the system logging utility rsyslogd(8) may be uninstalled. If it is installed, it changes its behavior to read the volatile binary log data (instead of pre-systemd default "`/dev/log`") and to create traditional permanent ASCII system log data. This can be customized by "`/etc/default/rsyslog`" and "`/etc/rsyslog.conf`" for both the log file and on-screen display. See rsyslogd(8) and rsyslog.conf(5). See also Section 9.3.2.

3.6 System management

The `systemd` offers not only init system but also generic system management operations with the systemctl(1) command.

Here, "`$unit`" in the above examples may be a single unit name (suffix such as `.service` and `.target` are optional) or, in many cases, multiple unit specifications (shell-style globs "`*`", "`?`", "`[]`" using fnmatch(3) which will be matched against the primary names of all units currently in memory).

System state changing commands in the above examples are typically preceded by the "`sudo`" to attain the required administrative privilege.

The output of the "`systemctl status $unit|$PID|$device`" uses color of the dot ("●") to summarize the unit state at a glance.

- White "●" indicates an "inactive" or "deactivating" state.
- Red "●" indicates a "failed" or "error" state.

Operation	Command snippets
List all available unit types	"systemctl list-units --type=help"
List all target units in memory	"systemctl list-units --type=target"
List all service units in memory	"systemctl list-units --type=service"
List all device units in memory	"systemctl list-units --type=device"
List all mount units in memory	"systemctl list-units --type=mount"
List all socket units in memory	"systemctl list-sockets"
List all timer units in memory	"systemctl list-timers"
Start "$unit"	"systemctl start $unit"
Stop "$unit"	"systemctl stop $unit"
Reload service-specific configuration	"systemctl reload $unit"
Stop and start all "$unit"	"systemctl restart $unit"
Start "$unit" and stop all others	"systemctl isolate $unit"
Switch to "graphical" (GUI system)	"systemctl isolate graphical"
Switch to "multi-user" (CLI system)	"systemctl isolate multi-user"
Switch to "rescue" (single user CLI system)	"systemctl isolate rescue"
Send kill signal to "$unit"	"systemctl kill $unit"
Check if "$unit" service is active	"systemctl is-active $unit"
Check if "$unit" service is failed	"systemctl is-failed $unit"
Check status of "$unit\|$PID|device"	"systemctl status $unit\|$PID|$device"
Show properties of "$unit\|$job"	"systemctl show $unit\|$job"
Reset failed "$unit"	"systemctl reset-failed $unit"
List dependency of all unit services	"systemctl list-dependencies --all"
List unit files installed on the system	"systemctl list-unit-files"
Enable "$unit" (add symlink)	"systemctl enable $unit"
Disable "$unit" (remove symlink)	"systemctl disable $unit"
Unmask "$unit" (remove symlink to "/dev/null")	"systemctl unmask $unit"
Mask "$unit" (add symlink to "/dev/null")	"systemctl mask $unit"
Get default-target setting	"systemctl get-default"
Set default-target to "graphical" (GUI system)	"systemctl set-default graphical"
Set default-target to "multi-user" (CLI system)	"systemctl set-default multi-user"
Show job environment	"systemctl show-environment"
Set job environment "variable" to "value"	"systemctl set-environment variable=value"
Unset job environment "variable"	"systemctl unset-environment variable"
Reload all unit files and daemons	"systemctl daemon-reload"
Shut down the system	"systemctl poweroff"
Shut down and reboot the system	"systemctl reboot"
Suspend the system	"systemctl suspend"
Hibernate the system	"systemctl hibernate"

Table 3.6: List of typical systemctl command snippets

- Green "●" indicates an "active", "reloading" or "activating" state.

3.7 Other system monitors

Here are a list of other monitoring command snippets under `systemd`. Please read the pertinent manpages including cgroups(7).

Operation	Command snippets
Show time spent for each initialization steps	"`systemd-analyze time`"
List of all units by the time to initialize	"`systemd-analyze blame`"
Load and detect errors in "`$unit`" file	"`systemd-analyze verify $unit`"
Show terse runtime status information of the user of the caller's session	"`loginctl user-status`"
Show terse runtime status information of the caller's session	"`loginctl session-status`"
Track boot process by the cgroups	"`systemd-cgls`"
Track boot process by the cgroups	"`ps xawf -eo pid,user,cgroup,args`"
Track boot process by the cgroups	Read sysfs under "`/sys/fs/cgroup/`"

Table 3.7: List of other monitoring command snippets under `systemd`

3.8 System configuration

3.8.1 The hostname

The kernel maintains the system **hostname**. The system unit started by `systemd-hostnamed.service` sets the system hostname at boot time to the name stored in "`/etc/hostname`". This file should contain **only** the system hostname, not a fully qualified domain name.

To print out the current hostname run hostname(1) without an argument.

3.8.2 The filesystem

The mount options of normal disk and network filesystems are set in "`/etc/fstab`". See fstab(5) and Section 9.6.7.

The configuration of the encrypted filesystem is set in "`/etc/crypttab`". See crypttab(5)

The configuration of software RAID with mdadm(8) is set in "`/etc/mdadm/mdadm.conf`". See mdadm.conf(5).



Warning
After mounting all the filesystems, temporary files in "`/tmp`", "`/var/lock`", and "`/var/run`" are cleaned for each boot up.

3.8.3 Network interface initialization

Network interfaces are typically initialized in "`networking.service`" for the `lo` interface and "`NetworkManager.service`" for other interfaces on modern Debian desktop system under `systemd`.

See Chapter 5 for how to configure them.

3.8.4 Cloud system initialization

The cloud system instance may be launched as a clone of "Debian Official Cloud Images" or similar images. For such system instance, personalities such as hostname, filesystem, networking, locale, SSH keys, users and groups may be configured using functionalities provided by `cloud-init` and `netplan.io` packages with multiple data sources such as files placed in the original system image and external data provided during its launch. These packages enable the declarative system configuration using YAML data.

See more at "Cloud Computing with Debian and its descendants", "Cloud-init documentation" and Section 5.4.

3.8.5 Customization example to tweak sshd service

With default installation, many network services (see Chapter 6) are started as daemon processes after `network.target` at boot time by `systemd`. The "`sshd`" is no exception. Let's change this to on-demand start of "`sshd`" as a customization example.

First, disable system installed service unit.

```
$ sudo systemctl stop sshd.service
$ sudo systemctl mask sshd.service
```

The on-demand socket activation system of the classic Unix services was through the `inetd` (or `xinetd`) superserver. Under `systemd`, the equivalent can be enabled by adding ***.socket** and ***.service** unit configuration files.

`sshd.socket` for specifying a socket to listen on

```
[Unit]
Description=SSH Socket for Per-Connection Servers

[Socket]
ListenStream=22
Accept=yes

[Install]
WantedBy=sockets.target
```

`sshd@.service` as the matching service file of `sshd.socket`

```
[Unit]
Description=SSH Per-Connection Server

[Service]
ExecStart=-/usr/sbin/sshd -i
StandardInput=socket
```

Then reload.

```
$ sudo systemctl daemon-reload
```

3.9 The udev system

The udev system provides mechanism for the automatic hardware discovery and initialization (see udev(7)) since Linux kernel 2.6. Upon discovery of each device by the kernel, the udev system starts a user process which uses information from the sysfs filesystem (see Section 1.2.12), loads required kernel modules supporting it using the modprobe(8) program (see Section 3.10), and creates corresponding device nodes.

Tip
If "`/lib/modules/`*`kernel-version`*`/modules.dep`" was not generated properly by depmod(8) for some reason, modules may not be loaded as expected by the udev system. Execute "`depmod -a`" to fix it.
For mounting rules in "`/etc/fstab`", device nodes do not need to be static ones. You can use UUID to mount devices instead of device names such as "`/dev/sda`". See Section 9.6.3.

Since the udev system is somewhat a moving target, I leave details to other documentations and describe the minimum information here.



Warning
Don't try to run long running programs such as backup script with RUN in udev rules as mentioned in udev(7). Please create a proper systemd.service(5) file and activate it, instead. See Section 10.2.3.2.

3.10 The kernel module initialization

The modprobe(8) program enables us to configure running Linux kernel from user process by adding and removing kernel modules. The udev system (see Section 3.9) automates its invocation to help the kernel module initialization.

There are non-hardware modules and special hardware driver modules as the following which need to be pre-loaded by listing them in the "`/etc/modules`" file (see modules(5)).

- TUN/TAP modules providing virtual Point-to-Point network device (TUN) and virtual Ethernet network device (TAP),
- netfilter modules providing netfilter firewall capabilities (iptables(8), Section 5.7), and
- watchdog timer driver modules.

The configuration files for the modprobe(8) program are located under the "`/etc/modprobes.d/`" directory as explained in modprobe.conf(5). (If you want to avoid some kernel modules to be auto-loaded, consider to blacklist them in the "`/etc/modprobes.d/blacklist`" file.)

The "`/lib/modules/`*`version`*`/modules.dep`" file generated by the depmod(8) program describes module dependencies used by the modprobe(8) program.

Note
If you experience module loading issues with boot time module loading or with modprobe(8), "`depmod -a`" may resolve these issues by reconstructing "`modules.dep`".

The modinfo(8) program shows information about a Linux kernel module.

The lsmod(8) program nicely formats the contents of the "`/proc/modules`", showing what kernel modules are currently loaded.

Tip
You can identify exact hardware on your system. See Section 9.5.3.
You may configure hardware at boot time to activate expected hardware features. See Section 9.5.4.
You can probably add support for your special device by recompiling the kernel. See Section 9.10.

Chapter 4

Authentication and access controls

When a person (or a program) requests access to the system, authentication confirms the identity to be a trusted one.



Warning
Configuration errors of PAM may lock you out of your own system. You must have a rescue CD handy or setup an alternative boot partition. To recover, boot the system with them and correct things from there.

4.1 Normal Unix authentication

Normal Unix authentication is provided by the pam_unix(8) module under the PAM (Pluggable Authentication Modules). Its 3 important configuration files, with "`:`" separated entries, are the following.

file	permission	user	group	description
`/etc/passwd`	`-rw-r--r--`	`root`	`root`	(sanitized) user account information
`/etc/shadow`	`-rw-r-----`	`root`	`shadow`	secure user account information
`/etc/group`	`-rw-r--r--`	`root`	`root`	group information

Table 4.1: 3 important configuration files for pam_unix(8)

"`/etc/passwd`" contains the following.

```
 ...
user1:x:1000:1000:User1 Name,,,:/home/user1:/bin/bash
user2:x:1001:1001:User2 Name,,,:/home/user2:/bin/bash
 ...
```

As explained in passwd(5), each "`:`" separated entry of this file means the following.

- Login name
- Password specification entry
- Numerical user ID
- Numerical group ID
- User name or comment field

- User home directory
- Optional user command interpreter

The second entry of "/etc/passwd" was used for the encrypted password entry. After the introduction of "/etc/shadow", this entry is used for the password specification entry.

content	meaning
(empty)	passwordless account
x	the encrypted password is in "/etc/shadow"

Table 4.2: The second entry content of "/etc/passwd"

"/etc/shadow" contains the following.

```
 ...
user1:$1$Xop0FYH9$IfxyQwBe9b8tiyIkt2P4F/:13262:0:99999:7:::
user2:$1$vXGZLVbS$ElyErNf/agUDsm1DehJMS/:13261:0:99999:7:::
 ...
```

As explained in shadow(5), each ":" separated entry of this file means the following.

- Login name
- Encrypted password (The initial "1" indicates use of the MD5 encryption. The "*" indicates no login.)
- Date of the last password change, expressed as the number of days since Jan 1, 1970
- Number of days the user will have to wait before she will be allowed to change her password again
- Number of days after which the user will have to change her password
- Number of days before a password is going to expire during which the user should be warned
- Number of days after a password has expired during which the password should still be accepted
- Date of expiration of the account, expressed as the number of days since Jan 1, 1970
- …

"/etc/group" contains the following.

```
group1:x:20:user1,user2
```

As explained in group(5), each ":" separated entry of this file means the following.

- Group name
- Encrypted password (not really used)
- Numerical group ID
- "," separated list of user names

Note
"/etc/gshadow" provides the similar function as "/etc/shadow" for "/etc/group" but is not really used.

Note
The actual group membership of a user may be dynamically added if "`auth optional pam_group.so`" line is added to "`/etc/pam.d/common-auth`" and set it in "`/etc/security/group.conf`". See pam_group(8).

Note
The `base-passwd` package contains an authoritative list of the user and the group: "`/usr/share/doc/base-passwd/users-and-groups.html`".

4.2 Managing account and password information

Here are few notable commands to manage account information.

command	function
`getent passwd` *`user_name`*	browse account information of "*`user_name`*"
`getent shadow` *`user_name`*	browse shadowed account information of "*`user_name`*"
`getent group` *`group_name`*	browse group information of "*`group_name`*"
`passwd`	manage password for the account
`passwd -e`	set one-time password for the account activation
`chage`	manage password aging information

Table 4.3: List of commands to manage account information

You may need to have the root privilege for some functions to work. See crypt(3) for the password and data encryption.

Note
On the system set up with PAM and NSS as the Debian salsa machine, the content of local "`/etc/passwd`", "`/etc/group`" and "`/etc/shadow`" may not be actively used by the system. Above commands are valid even under such environment.

4.3 Good password

When creating an account during your system installation or with the passwd(1) command, you should choose a good password which consists of at least 6 to 8 characters including one or more characters from each of the following sets according to passwd(1).

- Lower case alphabetics
- Digits 0 through 9
- Punctuation marks



Warning
Do not choose guessable words for the password. Account name, social security number, phone number, address, birthday, name of your family members or pets, dictionary words, simple sequence of characters such as "12345" or "qwerty", … are all bad choice for the password.

4.4 Creating encrypted password

There are independent tools to generate encrypted passwords with salt.

package	popcon	size	command	function
whois	V:23, I:203	384	mkpasswd	over-featured front end to the crypt(3) library
openssl	V:846, I:996	2564	openssl passwd	compute password hashes (OpenSSL). passwd(1ssl)

Table 4.4: List of tools to generate password

4.5 PAM and NSS

Modern Unix-like systems such as the Debian system provide PAM (Pluggable Authentication Modules) and NSS (Name Service Switch) mechanism to the local system administrator to configure his system. The role of these can be summarizes as the following.

- PAM offers a flexible authentication mechanism used by the application software thus involves password data exchange.
- NSS offers a flexible name service mechanism which is frequently used by the C standard library to obtain the user and group name for programs such as ls(1) and id(1).

These PAM and NSS systems need to be configured consistently.

The notable packages of PAM and NSS systems are the following.

package	popcon	size	description
libpam-modules	V:942, I:1000	892	Pluggable Authentication Modules (basic service)
libpam-ldapd	V:7, I:13	79	Pluggable Authentication Module allowing LDAP interfaces
libpam-systemd	V:764, I:969	848	Pluggable Authentication Module to register user sessions for logind
libpam-doc	I:7.5	1492	Pluggable Authentication Modules (documentation in html and text)
libc6	V:936, I:1000	5483	GNU C Library: Shared libraries which also provides "Name Service Switch" service
glibc-doc	I:5.1	3927	GNU C Library: Manpages
glibc-doc-reference	I:3.1	14764	GNU C Library: Reference manual in info, pdf and html format (non-free)
libnss-mdns	V:299, I:507	142	NSS module for Multicast DNS name resolution
libnss-ldapd	V:8, I:16	131	NSS module for using LDAP as a naming service

Table 4.5: List of notable PAM and NSS systems

- "The Linux-PAM System Administrators' Guide" in libpam-doc is essential for learning PAM configuration.
- "System Databases and Name Service Switch" section in glibc-doc-reference is essential for learning NSS configuration.

Note
You can see more extensive and current list by "`aptitude search 'libpam-|libnss-'`" command. The acronym NSS may also mean "Network Security Service" which is different from "Name Service Switch".

Note
PAM is the most basic way to initialize environment variables for each program with the system wide default value.

Under systemd, `libpam-systemd` package is installed to manage user logins by registering user sessions in the `systemd` control group hierarchy for logind. See systemd-logind(8), logind.conf(5), and pam_systemd(8).

4.5.1 Configuration files accessed by PAM and NSS

Here are a few notable configuration files accessed by PAM and NSS.

configuration file	function
`/etc/pam.d/`*`program_name`*	set up PAM configuration for the "*`program_name`*" program; see pam(7) and pam.d(5)
`/etc/nsswitch.conf`	set up NSS configuration with the entry for each service. See nsswitch.conf(5)
`/etc/nologin`	limit the user login by the pam_nologin(8) module
`/etc/securetty`	limit the tty for the root access by the pam_securetty(8) module
`/etc/security/access.conf`	set access limit by the pam_access(8) module
`/etc/security/group.conf`	set group based restraint by the pam_group(8) module
`/etc/security/pam_env.conf`	set environment variables by the pam_env(8) module
`/etc/environment`	set additional environment variables by the pam_env(8) module with the "`readenv=1`" argument
`/etc/default/locale`	set locale by pam_env(8) module with the "`readenv=1 envfile=/etc/default/locale`" argument (Debian)
`/etc/security/limits.conf`	set resource restraint (ulimit, core, ...) by the pam_linits(8) module
`/etc/security/time.conf`	set time restraint by the pam_time(8) module
`/etc/systemd/logind.conf`	set `systemd` login manager configuration (see logind.conf(5) and systemd-logind.service(8))

Table 4.6: List of configuration files accessed by PAM and NSS

The limitation of the password selection is implemented by the PAM modules, pam_unix(8) and pam_cracklib(8). They can be configured by their arguments.

Tip
PAM modules use suffix "`.so`" for their filenames.

4.5.2 The modern centralized system management

The modern centralized system management can be deployed using the centralized Lightweight Directory Access Protocol (LDAP) server to administer many Unix-like and non-Unix-like systems on the network. The open source implementation of the Lightweight Directory Access Protocol is OpenLDAP Software.

The LDAP server provides the account information through the use of PAM and NSS with `libpam-ldapd` and `libnss-ldapd` packages for the Debian system. Several actions are required to enable this (I have not used this setup and the following is purely secondary information. Please read this in this context.).

- You set up a centralized LDAP server by running a program such as the stand-alone LDAP daemon, slapd(8).
- You change the PAM configuration files in the "`/etc/pam.d/`" directory to use "`pam_ldap.so`" instead of the default "`pam_unix.so`".
- You change the NSS configuration in the "`/etc/nsswitch.conf`" file to use "`ldap`" instead of the default ("`compat`" or "`file`").
- You must make `libpam-ldapd` to use SSL (or TLS) connection for the security of password.
- You may make `libnss-ldapd` to use SSL (or TLS) connection to ensure integrity of data at the cost of the LDAP network overhead.
- You should run nscd(8) locally to cache any LDAP search results in order to reduce the LDAP network traffic.

See documentations in nsswitch.conf(5), pam.conf(5), ldap.conf(5), and "`/usr/share/doc/libpam-doc/html/`" offered by the `libpam-doc` package and "`info libc 'Name Service Switch'`" offered by the `glibc-doc` package.

Similarly, you can set up alternative centralized systems with other methods.

- Integration of user and group with the Windows system.
  - Access Windows domain services by the `winbind` and `libpam_winbind` packages.
  - See winbindd(8) and Integrating MS Windows Networks with Samba.
- Integration of user and group with the legacy Unix-like system.
  - Access NIS (originally called YP) or NIS+ by the `nis` package.
  - See The Linux NIS(YP)/NYS/NIS+ HOWTO.

4.5.3 "Why GNU su does not support the wheel group"

This is the famous phrase at the bottom of the old "`info su`" page by Richard M. Stallman. Not to worry: the current `su` command in Debian uses PAM, so that one can restrict the ability to use `su` to the `root` group by enabling the line with "`pam_wheel.so`" in "`/etc/pam.d/su`".

4.5.4 Stricter password rule

Installing the `libpam-cracklib` package enables you to force stricter password rule.

On a typical GNOME system which automatically installs `libpam-gnome-keyring`, "`/etc/pam.d/common-password`" looks like:

```
# here are the per-package modules (the "Primary" block)
password  requisite     pam_cracklib.so retry=3 minlen=8 difok=3
password  [success=1 default=ignore]  pam_unix.so obscure use_authtok try_first_pass  ↩
    yescrypt
# here's the fallback if no module succeeds
password  requisite     pam_deny.so
# prime the stack with a positive return value if there isn't one already;
# this avoids us returning an error just because nothing sets a success code
# since the modules above will each just jump around
password  required      pam_permit.so
# and here are more per-package modules (the "Additional" block)
password  optional  pam_gnome_keyring.so
# end of pam-auth-update config
```

4.6 Security of authentication

Note
The information here **may not be sufficient** for your security needs but it should be a **good start**.

4.6.1 Secure password on the Internet

Many popular transportation layer services communicate messages including password authentication in the plain text. It is very bad idea to transmit password in the plain text over the wild Internet where it can be intercepted. You can run these services over "Transport Layer Security" (TLS) or its predecessor, "Secure Sockets Layer" (SSL) to secure entire communication including password by the encryption.

insecure service name	port	secure service name	port
`www (http)`	80	`https`	443
`smtp (mail)`	25	`ssmtp (smtps)`	465
`ftp-data`	20	`ftps-data`	989
`ftp`	21	`ftps`	990
`telnet`	23	`telnets`	992
`imap2`	143	`imaps`	993
`pop3`	110	`pop3s`	995
`ldap`	389	`ldaps`	636

Table 4.7: List of insecure and secure services and ports

The encryption costs CPU time. As a CPU friendly alternative, you can keep communication in plain text while securing just the password with the secure authentication protocol such as "Authenticated Post Office Protocol" (APOP) for POP and "Challenge-Response Authentication Mechanism MD5" (CRAM-MD5) for SMTP and IMAP. (For sending mail messages over the Internet to your mail server from your mail client, it is recently popular to use new message submission port 587 instead of traditional SMTP port 25 to avoid port 25 blocking by the network provider while authenticating yourself with CRAM-MD5.)

The traditional password-based authentication still suffers inherent security issues such as password reuse across multiple websites. See other authentication methods for the secure solution: Time-based one-time password (TOTP), Universal 2nd Factor (U2F), Web Authentication (WebAuthn, commonly referred as "passkey")

4.6.2 Secure Shell

The Secure Shell (SSH) program provides secure encrypted communications between two untrusted hosts over an insecure network with the secure authentication. It consists of the OpenSSH client, ssh(1), and the OpenSSH daemon, sshd(8). This SSH can be used to tunnel an insecure protocol communication such as POP and X securely over the Internet with the port forwarding feature.

The client tries to authenticate itself using host-based authentication, public key authentication, challenge-response authentication, or password authentication. The use of public key authentication enables the remote password-less login. See Section 6.3.

4.6.3 Extra security measures for the Internet

Even when you run secure services such as Secure Shell (SSH) and Point-to-point tunneling protocol (PPTP) servers, there are still chances for the break-ins using brute force password guessing attack etc. from the Internet. Use of the firewall policy (see Section 5.7) together with the following security tools may improve the security situation.

package	popcon	size	description
knockd	V:0.7, I:1.6	110	small port-knock daemon knockd(1) and client knock(1)
fail2ban	V:101, I:111	2275	ban IPs that cause multiple authentication errors
libpam-shield	V:0.04, I:0.05	115	lock out remote attackers trying password guessing

Table 4.8: List of tools to provide extra security measures

4.6.4 Securing the root password

To prevent people to access your machine with root privilege, you need to make following actions.

- Prevent physical access to the system storage device (HDD / SSD / ...)
- Lock UEFI/BIOS and prevent booting from the removable media
- Set password for GRUB interactive session
- Lock GRUB menu from editing

With physical access to the system storage device, resetting the password is relatively easy with following steps.

1. Move the system storage device to a PC with USB bootable UEFI/BIOS.
2. Boot system with a rescue media (see Section 3.2.2).
3. Mount root partition with read/write access.
4. Edit "`/etc/passwd`" in the root partition and make the second entry for the `root` account empty.

If you have edit access to the GRUB menu entry (see Section 3.1.2) for `grub-rescue-pc` at boot time, it is even easier with following steps.

1. Boot system with the kernel parameter changed to something like "`root=/dev/sda6 rw init=/bin/sh`".
2. Edit "`/etc/passwd`" and make the second entry for the `root` account empty.
3. Reboot system.

The root shell of the system is now accessible without password.

Note
Once one has root shell access, he can access everything on the system and reset any passwords on the system. Further more, he may compromise password for all user accounts using brute force password cracking tools such as `john` and `crack` packages (see Section 9.5.10). This cracked password may lead to compromise other systems.

The only reasonable software solution to avoid all these concerns is to use software encrypted root partition (or "`/etc`" partition) using dm-crypt and initramfs (see Section 9.9). You always need password to boot the system, though.

4.7 Other access controls

There are access controls to the system other than the password based authentication and file permissions.

Note
See Section 9.4.16 for restricting the kernel secure attention key (SAK) feature.

4.7.1 Access control lists (ACLs)

ACLs are a superset of the regular permissions as explained in Section 1.2.3.

You encounter ACLs in action on modern desktop environment. When a formatted USB storage device is auto mounted as, e.g., "`/media/penguin/USBSTICK`", a normal user `penguin` can execute:

```
 $ cd /media/penguin
 $ ls -la
total 16
drwxr-x---+ 1 root    root    16 Jan 17 22:55 .
drwxr-xr-x  1 root    root    28 Sep 17 19:03 ..
drwxr-xr-x  1 penguin penguin 18 Jan  6 07:05 USBSTICK
```

"+" in the 11th column indicates ACLs are in action. Without ACLs, a normal user `penguin` shouldn't be able to list like this since `penguin` isn't in `root` group. You can see ACLs as:

```
 $ getfacl .
# file: .
# owner: root
# group: root
user::rwx
user:penguin:r-x
group::---
mask::r-x
other::---
```

Here:

- "`user::rwx`", "`group::---`", and "`other::---`" correspond to the regular owner, group, and other permissions.
- The ACL "`user:penguin:r-x`" allows a normal user `penguin` to have "`r-x`" permissions. This enabled "`ls -la`" to list directory content.
- The ACL "`mask::r-x`" sets the upper bound of permissions.

See "POSIX Access Control Lists on Linux", acl(5), getfacl(1), and setfacl(1) for more.

4.7.2 sudo

sudo(8) is a program designed to allow a sysadmin to give limited root privileges to users and log root activity. `sudo` requires only an ordinary user's password. Install `sudo` package and activate it by setting options in "`/etc/sudoers`". See configuration example at "`/usr/share/doc/sudo/examples/sudoers`" and Section 1.1.12.

My usage of `sudo` for the single user system (see Section 1.1.12) is aimed to protect myself from my own stupidity. Personally, I consider using `sudo` a better alternative than using the system from the root account all the time. For example, the following changes the owner of "*`some_file`*" to "*`my_name`*".

```
$ sudo chown my_name some_file
```

Of course if you know the root password (as self-installed Debian users do), any command can be run under root from any user's account using "`su -c`".

4.7.3 PolicyKit

PolicyKit is an operating system component for controlling system-wide privileges in Unix-like operating systems.

Newer GUI applications are not designed to run as privileged processes. They talk to privileged processes via PolicyKit to perform administrative operations.

PolicyKit limits such operations to user accounts belonging to the `sudo` group on the Debian system.

See polkit(8).

4.7.4 Restricting access to some server services

For system security, it is a good idea to disable as much server programs as possible. This becomes critical for network servers. Having unused servers, activated either directly as daemon or via super-server program, are considered security risks.

Many programs, such as sshd(8), use PAM based access control. There are many ways to restrict access to some server services.

- configuration files: "`/etc/default/`*`program_name`*"
- Systemd service unit configuration for daemon
- PAM (Pluggable Authentication Modules)
- "`/etc/inetd.conf`" for super-server
- "`/etc/hosts.deny`" and "`/etc/hosts.allow`" for TCP wrapper, tcpd(8)
- "`/etc/rpc.conf`" for Sun RPC
- "`/etc/at.allow`" and "`/etc/at.deny`" for atd(8)
- "`/etc/cron.allow`" and "`/etc/cron.deny`" for crontab(1)
- Network firewall of netfilter infrastructure

See Section 3.6, Section 4.5.1, and Section 5.7.

Tip
Sun RPC services need to be active for NFS and other RPC based programs.

Tip
If you have problems with remote access in a recent Debian system, comment out offending configuration such as "`ALL: PARANOID`" in "`/etc/hosts.deny`" if it exists. (But you must be careful on security risks involved with this kind of action.)

4.7.5 Linux security features

Linux kernel has evolved and supports security features not found in traditional UNIX implementations.

Linux supports extended attributes which extend the traditional UNIX attributes (see xattr(7)).

Linux divides the privileges traditionally associated with superuser into distinct units, known as capabilities(7), which can be independently enabled and disabled. Capabilities are a per-thread attribute since kernel version 2.2.

The Linux Security Module (LSM) framework provides a mechanism for various security checks to be hooked by new kernel extensions. For example:

- AppArmor
- Security-Enhanced Linux (SELinux)
- Smack (Simplified Mandatory Access Control Kernel)
- Tomoyo Linux

Since these extensions may tighten privilege model tighter than the ordinary Unix-like security model policies, even the root power may be restricted. You are advised to read the Linux Security Module (LSM) framework document at kernel.org.

Linux namespaces wrap a global system resource in an abstraction that makes it appear to the processes within the namespace that they have their own isolated instance of the global resource. Changes to the global resource are visible to other processes that are members of the namespace, but are invisible to other processes. Since kernel version 5.6, there are 8 kinds of namespaces (see namespaces(7), unshare(1), nsenter(1)).

As of Debian 11 Bullseye (2021), Debian uses unified cgroup hierarchy (a.k.a. cgroups-v2).

Usage examples of namespaces with cgroups to isolate their processes and to allow resource control are:

- Systemd. See Section 3.3.1.
- Sandbox environment. See Section 7.7.
- Linux containers such as Docker, LXC. See Section 9.11.

These functionalities can't be realized by Section 4.1. These advanced topics are mostly out-of-scope for this introductory document.

Chapter 5

Network setup

Tip
For modern Debian specific guide to the networking, read The Debian Administrator's Handbook — Configuring the Network.

Tip
Under systemd, networkd may be used to manage networks. See systemd-networkd(8).

5.1 The basic network infrastructure

Let's review the basic network infrastructure on the modern Debian system.

5.1.1 The hostname resolution

The hostname resolution is currently supported by the NSS (Name Service Switch) mechanism too. The flow of this resolution is the following.

1. The "`/etc/nsswitch.conf`" file with stanza like "`hosts: files dns`" dictates the hostname resolution order. (This replaces the old functionality of the "`order`" stanza in "`/etc/host.conf`".)
2. The `files` method is invoked first. If the hostname is found in the "`/etc/hosts`" file, it returns all valid addresses for it and exits. (The "`/etc/host.conf`" file contains "`multi on`".)
3. The `dns` method is invoked. If the hostname is found by the query to the Internet Domain Name System (DNS) identified by the "`/etc/resolv.conf`" file, it returns all valid addresses for it and exits.

A typical workstation may be installed with its host name set to, e.g., "*host_name*" and its optional domain name set to an empty string. Then, "`/etc/hosts`" looks like the following.

```
127.0.0.1 localhost
127.0.1.1 host_name

# The following lines are desirable for IPv6 capable hosts
::1     localhost ip6-localhost ip6-loopback
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
```

packages	popcon	size	type	description
network-manager	V:427, I:474	7365	config::NM	NetworkManager (daemon): manage the network automatically
network-manager-gnome	V:36, I:147	19	config::NM	NetworkManager (GNOME frontend)
netplan.io	V:2.1, I:8.4	340	config::NM+networkd	Netplan (generator): Unified, declarative interface to NetworkManager and systemd-networkd backends
ifupdown	V:649, I:973	204	config::ifupdown	standardized tool to bring up and down the network (Debian specific)
pppoeconf	V:0.2, I:3.9	176	config::helper	configuration helper for PPPoE connection
wpasupplicant	V:421, I:521	3896	config::helper	client support for WPA and WPA2 (IEEE 802.11i)
wpagui	V:0.2, I:1.2	779	config::helper	Qt GUI client for wpa_supplicant
wireless-tools	V:193, I:262	232	config::helper	tools for manipulating Linux Wireless Extensions
iw	V:40, I:478	332	config::helper	tool for configuring Linux wireless devices
iproute2	V:781, I:987	4268	config::iproute2	iproute2, IPv6 and other advanced network configuration: ip(8), tc(8), etc
iptables	V:381, I:635	2408	config::Netfilter	administration tools for packet filtering and NAT (Netfilter)
nftables	V:266, I:880	189	config::Netfilter	administration tools for packet filtering and NAT (Netfilter) (successor to {ip,ip6,arp,eb}tables)
iputils-ping	V:206, I:998	188	test	test network reachability of a remote host by hostname or IP address (iproute2)
iputils-arping	V:2, I:17	53	test	test network reachability of a remote host specified by the ARP address
iputils-tracepath	V:2, I:19	50	test	trace the network path to a remote host
ethtool	V:98, I:251	1093	test	display or change Ethernet device settings
mtr-tiny	V:4, I:38	182	test::low-level	trace the network path to a remote host (curses)
mtr	V:4, I:40	231	test::low-level	trace the network path to a remote host (curses and GTK)
gnome-nettool	V:0.5, I:8.4	2480	test::low-level	tools for common network information operations (GNOME)
nmap	V:25, I:182	4934	test::low-level	network mapper / port scanner (Nmap, console)
tcpdump	V:20, I:177	1346	test::low-level	network traffic analyzer (Tcpdump, console)
wireshark	V:4, I:39	17068	test::low-level	network traffic analyzer (Wireshark, GTK)
tshark	V:2, I:23	433	test::low-level	network traffic analyzer (console)
tcptrace	V:0.2, I:1.5	407	test::low-level	produce a summarization of the connections from `tcpdump` output
dnsutils	V:5, I:146	23	test::low-level	network clients provided with BIND: nslookup(8), nsupdate(8), dig(8)
dlint	V:0.1, I:2.0	52	test::low-level	check DNS zone information using nameserver lookups
dnstracer	V:0.1, I:1.1	63	test::low-level	trace a chain of DNS servers to the source

Table 5.1: List of network configuration tools

Each line starts with a IP address and it is followed by the associated hostname.

The IP address `127.0.1.1` in the second line of this example may not be found on some other Unix-like systems. The Debian Installer creates this entry for a system without a permanent IP address as a workaround for some software (e.g., GNOME) as documented in the bug #719621.

The *host_name* matches the hostname defined in the "`/etc/hostname`" (see Section 3.8.1).

For a system with a permanent IP address, that permanent IP address should be used here instead of `127.0.1.1`.

For a system with a permanent IP address and a fully qualified domain name (FQDN) provided by the Domain Name System (DNS), that canonical *host_name*.*domain_name* should be used instead of just *host_name*.

The "`/etc/resolv.conf`" is a static file if the `resolvconf` package is not installed. If installed, it is a symbolic link. Either way, it contains information that initialize the resolver routines. If the DNS is found at IP="`192.168.11.1`", it contains the following.

```
nameserver 192.168.11.1
```

The `resolvconf` package makes this "`/etc/resolv.conf`" into a symbolic link and manages its contents by the hook scripts automatically.

For the PC workstation on the typical adhoc LAN environment, the hostname can be resolved via Multicast DNS (mDNS) in addition to the basic `files` and `dns` methods.

- Avahi provides a framework for Multicast DNS Service Discovery on Debian.
- It is equivalent of Apple Bonjour / Apple Rendezvous.
- The `libnss-mdns` plugin package provides host name resolution via mDNS for the GNU Name Service Switch (NSS) functionality of the GNU C Library (glibc).
- The "`/etc/nsswitch.conf`" file should have stanza like "`hosts: files mdns4_minimal [NOTFOUND=return] dns`" (see `/usr/share/doc/libnss-mdns/README.Debian` for other configurations).
- A host name suffixed with the ".local" pseudo-top-level domain is resolved by sending a mDNS query message in a multicast UDP packet using IPv4 address "`224.0.0.251`" or IPv6 address "`FF02::FB`".

Note
The expansion of generic Top-Level Domains (gTLD) in the Domain Name System is underway. Watch out for the name collision when choosing a domain name used only within LAN.

Note
Use of packages such as `libnss-resolve` together with `systemd-resolved`, or `libnss-myhostname`, or `libnss-mymachine`, with coressponding listings on the "`hosts`" line in the "`/etc/nsswitch.conf`" file may override the traditional network configuration discussed in the above. See nss-resolve(8), systemd-resolved(8), nss-myhostname(8), and nss-mymachines(8) for more.

5.1.2 The network interface name

The systemd uses "Predictable Network Interface Names" such as "`enp0s25`".

Class	network addresses	net mask	net mask /bits	number of subnets
A	10.x.x.x	255.0.0.0	/8	1
B	172.16.x.x — 172.31.x.x	255.255.0.0	/16	16
C	192.168.0.x — 192.168.255.x	255.255.255.0	/24	256

Table 5.2: List of network address ranges

5.1.3 The network address range for the LAN

Let us be reminded of the IPv4 32 bit address ranges in each class reserved for use on the local area networks (LANs) by rfc1918. These addresses are guaranteed not to conflict with any addresses on the Internet proper.

Note
IP address written with colon are IPv6 address, e.g., "`::1`" for `localhost`.

Note
If one of these addresses is assigned to a host, then that host must not access the Internet directly but must access it through a gateway that acts as a proxy for individual services or else does Network Address Translation (NAT). The broadband router usually performs NAT for the consumer LAN environment.

5.1.4 The network device support

Although most hardware devices are supported by the Debian system, there are some network devices which require DFSG non-free firmware to support them. Please see Section 9.10.5.

5.2 The modern network configuration for desktop

Network interfaces are typically initialized in "`networking.service`" for the `lo` interface and "`NetworkManager.service`" for other interfaces on modern Debian desktop system under `systemd`.

Debian can manage the network connection via management daemon software such as NetworkManager (NM) (network-manager and associated packages).

- They come with their own GUI and command-line programs as their user interfaces.
- They come with their own daemon as their backend system.
- They allow easy connection of your system to the Internet.
- They allow easy management of wired and wireless network configuration.
- They allow us to configure network independent of the legacy `ifupdown` package.

Note
Do not use these automatic network configuration tools for servers. These are aimed primarily for mobile desktop users on laptops.

These modern network configuration tools need to be configured properly to avoid conflicting with the legacy `ifupdown` package and its configuration file "`/etc/network/interfaces`".

5.2.1 GUI network configuration tools

Official documentations for NM on Debian are provided in "`/usr/share/doc/network-manager/README.Debian`".

Essentially, the network configuration for desktop is done as follows.

1. Make desktop user, e.g. `foo`, belong to group "`netdev`" by the following (Alternatively, do it automatically via D-bus under modern desktop environments such as GNOME and KDE).

```
$ sudo usermod -a -G netdev foo
```

2. Keep configuration of "`/etc/network/interfaces`" as simple as in the following.

```
auto lo
iface lo inet loopback
```

3. Restart NM by the following.

```
$ sudo systemctl restart NetworkManager
```

4. Configure your network via GUI.

Note
Only interfaces which are **not** listed in "`/etc/network/interfaces`" are managed by NM to avoid conflict with `ifupdown`.

Tip
If you wish to extend network configuration capabilities of NM, please seek appropriate plug-in modules and supplemental packages such as `network-manager-openconnect`, `network-manager-openvpn-gnome`, `network-manager-pptp-gnome`, `mobile-broadband-provider-info`, `gnome-bluetooth`, etc.

5.3 The modern network configuration without GUI

Under systemd, the network may be configured in `/etc/systemd/network/` instead. See systemd-resolved(8), resolved.conf(5), and systemd-networkd(8).

This allows the modern network configuration without GUI.

A DHCP client configuration can be set up by creating "`/etc/systemd/network/dhcp.network`". E.g.:

```
[Match]
Name=en*

[Network]
DHCP=yes
```

A static network configuration can be set up by creating "`/etc/systemd/network/static.network`". E.g.:

```
[Match]
Name=en*

[Network]
Address=192.168.0.15/24
Gateway=192.168.0.1
```

5.4 The modern network configuration for cloud

The modern network configuration for cloud may use `cloud-init` and `netplan.io` packages (see Section 3.8.4).

The `netplan.io` package supports `systemd-networkd` and `NetworkManager` as its network configuration backends, and enables the declarative network configuration using YAML data. When you change YAML:

- Run "`netplan generate`" command to generate all the necessary backend configuration from YAML.
- Run "`netplan apply`" command to apply the generated configuration to the backends.

See "Netplan documentation", netplan(5), netplan-generate(8), and netplan-apply(8).

See also "Cloud-init documentation" (especially around "Configuration sources" and "Netplan Passthrough") for how `cloud-init` can integrate `netplan.io` configuration with alternative data sources.

5.4.1 The modern network configuration for cloud with DHCP

A DHCP client configuration can be set up by creating a data source file "`/etc/netplan/50-dhcp.yaml`":

```
network:
  version: 2
  ethernets:
    all-en:
      match:
        name: "en*"
      dhcp4: true
      dhcp6: true
```

5.4.2 The modern network configuration for cloud with static IP

A static network configuration can be set up by creating a data source file "`/etc/netplan/50-static.yaml`":

```
network:
  version: 2
  ethernets:
    eth0:
      addresses:
        - 192.168.0.15/24
      routes:
        - to: default
          via: 192.168.0.1
```

5.4.3 The modern network configuration for cloud with Network Manager

The network client configuration using Network Manager infrastructure can be set up by creating a data source file "`/etc/netplan/00-network-manager.yaml`":

```
network:
  version: 2
  renderer: NetworkManager
```

5.5 The low level network configuration

For the low level network configuration on Linux, use the iproute2 programs (ip(8), …) .

5.5.1 Iproute2 commands

Iproute2 commands offer complete low-level network configuration capabilities. Here is a translation table from obsolete net-tools commands to new iproute2 etc. commands.

obsolete net-tools	new iproute2 etc.	manipulation
ifconfig(8)	`ip addr`	protocol (IP or IPv6) address on a device
route(8)	`ip route`	routing table entry
arp(8)	`ip neigh`	ARP or NDISC cache entry
`ipmaddr`	`ip maddr`	multicast address
`iptunnel`	`ip tunnel`	tunnel over IP
nameif(8)	ifrename(8)	name network interfaces based on MAC addresses
mii-tool(8)	ethtool(8)	Ethernet device settings

Table 5.3: Translation table from obsolete `net-tools` commands to new `iproute2` commands

See ip(8) and Linux Advanced Routing & Traffic Control.

5.5.2 Safe low level network operations

You may use low level network commands as follows safely since they do not change network configuration.

command	description
`ip addr show`	display the link and address status of active interfaces
`route -n`	display all the routing table in numerical addresses
`ip route show`	display all the routing table in numerical addresses
`arp`	display the current content of the ARP cache tables
`ip neigh`	display the current content of the ARP cache tables
`plog`	display ppp daemon log
`ping yahoo.com`	check the Internet connection to "`yahoo.com`"
`whois yahoo.com`	check who registered "`yahoo.com`" in the domains database
`traceroute yahoo.com`	trace the Internet connection to "`yahoo.com`"
`tracepath yahoo.com`	trace the Internet connection to "`yahoo.com`"
`mtr yahoo.com`	trace the Internet connection to "`yahoo.com`" (repeatedly)
`dig [@dns-server.com] example.com [{a|mx|any}]`	check DNS records of "`example.com`" by "`dns-server.com`" for a "a", "mx", or "any" record
`iptables -L -n`	check packet filter
`ss -a`	find all open ports
`ss -l`	find listening ports
`ss -l -t -n`	find listening TCP ports (numeric)
`dlint example.com`	check DNS zone information of "`example.com`"

Table 5.4: List of low level network commands

Tip
Some of these low level network configuration tools reside in "`/usr/sbin/`". You may need to issue full command path such as "`/usr/sbin/ifconfig`" or add "`/usr/sbin`" to the "`$PATH`" list in your "`~/.bashrc`".

5.6 Network optimization

Generic network optimization is beyond the scope of this documentation. I touch only subjects pertinent to the consumer grade connection.

packages	popcon	size	description
iftop	V:6, I:85	93	display bandwidth usage information on an network interface
iperf	V:2, I:32	431	Internet Protocol bandwidth measuring tool
ifstat	V:0.5, I:5.3	52	InterFace STATistics Monitoring
bmon	V:2, I:19	154	portable bandwidth monitor and rate estimator
ethstatus	V:0.2, I:2.3	41	script that quickly measures network device throughput
bing	V:0.08, I:0.47	80	empirical stochastic bandwidth tester
bwm-ng	V:1.0, I:9.1	95	small and simple console-based bandwidth monitor
ethstats	V:0.05, I:0.36	21	console-based Ethernet statistics monitor
ipfm	V:0.04, I:0.09	78	bandwidth analysis tool

Table 5.5: List of network optimization tools

5.6.1 Finding optimal MTU

NM normally sets optimal Maximum Transmission Unit (MTU) automatically.

In some occasion, you may wish to set MTU manually after experiments with ping(8) with "`-M do`" option to send a ICMP packet with various data packet size. MTU is the maximum succeeding data packet size without IP fragmentation plus 28 bytes for the IPv4 and plus 48 bytes for the IPv6. For example the following finds MTU for IPv4 connection to be 1460 and MTU for IPv6 connection to be 1500.

```
$ ping -4 -c 1 -s $((1500-28)) -M do www.debian.org
PING  (149.20.4.15) 1472(1500) bytes of data.
ping: local error: message too long, mtu=1460

---  ping statistics ---
1 packets transmitted, 0 received, +1 errors, 100% packet loss, time 0ms

$ ping -4 -c 1 -s $((1460-28)) -M do www.debian.org
PING  (130.89.148.77) 1432(1460) bytes of data.
1440 bytes from klecker-misc.debian.org (130.89.148.77): icmp_seq=1 ttl=50 time=325 ms

---  ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 325.318/325.318/325.318/0.000 ms
$ ping -6 -c 1 -s $((1500-48)) -M do www.debian.org
PING www.debian.org(mirror-csail.debian.org (2603:400a:ffff:bb8::801f:3e)) 1452 data bytes
1460 bytes from mirror-csail.debian.org (2603:400a:ffff:bb8::801f:3e): icmp_seq=1 ttl=47  ↩
    time=191 ms
```

```
--- www.debian.org ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 191.332/191.332/191.332/0.000 ms
```

This process is Path MTU (PMTU) discovery (RFC1191) and the tracepath(8) command can automate this.

network environment	MTU	rationale
Dial-up link (IP: PPP)	576	standard
Ethernet link (IP: DHCP or fixed)	1500	standard and default

Table 5.6: Basic guide lines of the optimal MTU value

In addition to these basic guide lines, you should know the following.

- Any use of tunneling methods (VPN etc.) may reduce optimal MTU further by their overheads.
- The MTU value should not exceed the experimentally determined PMTU value.
- The bigger MTU value is generally better when other limitations are met.

The maximum segment size (MSS) is used as an alternative measure of packet size. The relationship between MSS and MTU are the following.

- MSS = MTU - 40 for IPv4
- MSS = MTU - 60 for IPv6

Note
The iptables(8) (see Section 5.7) based optimization can clamp packet size by the MSS and is useful for the router. See "TCPMSS" in iptables(8).

5.6.2 WAN TCP optimization

The TCP throughput can be maximized by adjusting TCP buffer size parameters as in "TCP tuning" for the modern high-bandwidth and high-latency WAN. So far, the current Debian default settings serve well even for my LAN connected by the fast 1G bps FTTP service.

5.7 Netfilter infrastructure

Netfilter provides infrastructure for stateful firewall and network address translation (NAT) with Linux kernel modules (see Section 3.10).

See The netfilter.org "nftables" project for the newer user space nft(8) command in the `nftables` package. (Linux kernel >= 3.13)

See The netfilter.org "iptables" project for the older user space iptables(8) and similar commands in the `iptables` and similar packages. (Linux kernel >= 2.4.x)

You can manually configure netfilter interactively from shell, save its state with "`nft list ruleset > file`" or iptables-save(8), and restore it via init script with "`nft -f file`" or iptables-restore(8) upon system reboot.

Configuration helper scripts such as shorewall ease this process.

See documentations at Netfilter Documentation (or in "`/usr/share/doc/iptables/html/`").

packages	popcon	size	description
nftables	V:266, I:880	189	administration tools for packet filtering and NAT (Netfilter) (successor to {ip,ip6,arp,eb}tables)
iptables	V:381, I:635	2408	administration tools for netfilter (iptables(8) for IPv4, ip6tables(8) for IPv6)
arptables	V:0.1, I:1.6	102	administration tools for netfilter (arptables(8) for ARP)
ebtables	V:14, I:23	276	administration tools for netfilter (ebtables(8) for Ethernet bridging)
iptstate	V:0.1, I:1.6	121	continuously monitor netfilter state (similar to top(1))
ufw	V:83, I:109	862	Uncomplicated Firewall (UFW) is a program for managing a netfilter firewall
gufw	V:6, I:11	3682	graphical user interface for Uncomplicated Firewall (UFW)
firewalld	V:15, I:22	2581	firewalld is a dynamically managed firewall program with support for network zones
firewall-config	V:0.8, I:3.2	1096	graphical user interface for firewalld
shorewall-init	V:0.17, I:0.33	88	Shoreline Firewall initialization
shorewall	V:2.1, I:4.7	3090	Shoreline Firewall, netfilter configuration file generator
shorewall-lite	V:0.03, I:0.06	71	Shoreline Firewall, netfilter configuration file generator (light version)
shorewall6	V:0.6, I:1.2	1334	Shoreline Firewall, netfilter configuration file generator (IPv6 version)
shorewall6-lite	V:0.01, I:0.02	71	Shoreline Firewall, netfilter configuration file generator (IPv6, light version)

Table 5.7: List of firewall tools

- Linux Networking-concepts HOWTO
- Linux 2.4 Packet Filtering HOWTO
- Linux 2.4 NAT HOWTO
- nftables HOWTO

Chapter 6

Network applications

After establishing network connectivity (see Chapter 5), you can run various network applications.

Tip
For modern Debian specific guide to the network infrastructure, read The Debian Administrator's Handbook — Network Infrastructure.

Tip
If you enabled "2-Step Verification" with some ISP, you need to obtain an application password to access POP and SMTP services from your program. You may need to approve your host IP in advance.

6.1 Web browsers

There are many web browser packages to access remote contents with Hypertext Transfer Protocol (HTTP).

package	popcon	size	type	description of web browser
chromium	V:31, I:100	320285	X	Chromium, (open-source browser from Google)
firefox	V:15, I:21	308351	, ,	Firefox, (open-source browser from Mozilla, only available in Debian Unstable)
firefox-esr	V:186, I:425	299827	, ,	Firefox ESR, (Firefox Extended Support Release)
epiphany-browser	V:2, I:10	6735	, ,	GNOME, HIG compliant, Epiphany
konqueror	V:26, I:111	8179	, ,	KDE, Konqueror
dillo	V:0.4, I:3.9	2203	, ,	Dillo, (light weight browser, FLTK based)
w3m	V:10, I:129	2853	text	w3m
lynx	V:29, I:455	2031	, ,	Lynx
elinks	V:2, I:15	1791	, ,	ELinks
links	V:2, I:20	2321	, ,	Links (text only)
links2	V:0.8, I:9.4	5466	graphics	Links (console graphics without X)

Table 6.1: List of web browsers

6.1.1 Spoofing the User-Agent string

In order to access some overly restrictive web sites, you may need to spoof the User-Agent string returned by the web browser program. See:

- MDN Web Docs: userAgent
- Chrome Developers: Override the user agent string
- How to change your user agent
- How to Change User-Agent in Chrome, Firefox, Safari, and more
- How to Change Your Browser's User Agent Without Installing Any Extensions
- How to change the User Agent in Gnome Web (epiphany)

6.1.2 Browser extension

All modern GUI browsers support source code based browser extension and it is becoming standardized as web extensions.

6.2 The mail system

This section focuses on typical mobile workstations on consumer grade Internet connections.



Caution
If you are to set up the mail server to exchange mail directly with the Internet, you should be better than reading this elementary document.

6.2.1 Email basics

An email message consists of three components, the message envelope, the message header, and the message body.

- The "To" and "From" information in the message envelope is used by the SMTP to deliver the email. (The "From" information in the message envelope is also called bounce address, From_, etc.).
- The "To" and "From" information in the message header is displayed by the email client. (While it is most common for these to be the same as ones in the message envelope, such is not always the case.)
- The email message format covering header and body data is extended by Multipurpose Internet Mail Extensions (MIME) from the plain ASCII text to other character encodings, as well as attachments of audio, video, images, and application programs.

Full featured GUI based email clients offer all the following functions using the GUI based intuitive configuration.

- It creates and interprets the message header and body data using Multipurpose Internet Mail Extensions (MIME) to deal the content data type and encoding.
- It authenticates itself to the ISP's SMTP and IMAP servers using the legacy basic access authentication or modern OAuth 2.0. (For OAuth 2.0, set it via Desktop environment settings. E.g., "Settings" -> "Online Accounts".)

- It sends the message to the ISP's smarthost SMTP server listening to the message submission port (587).
- It receives the stored message on the ISP's server from the TLS/IMAP4 port (993).
- It can filter mails by their attributes.
- It may offer additional functionalities: Contacts, Calendar, Tasks, Memos.

package	popcon	size	type
`evolution`	V:27, I:229	489	X GUI program (GNOME, groupware suite)
`thunderbird`	V:39, I:102	303329	X GUI program (GTK, Mozilla Thunderbird)
`kmail`	V:43, I:103	25265	X GUI program (KDE)
`mutt`	V:10, I:82	7334	character terminal program probably used with `vim`
`mew`	V:0.01, I:0.14	2319	character terminal program under `(x)emacs`

Table 6.2: List of mail user agent (MUA)

6.2.2 Modern mail service limitation

Modern mail service are under some limitations in order to minimize exposure to the spam (unwanted and unsolicited email) problems.

- It is not realistic to run SMTP server on the consumer grade network to send mail directly to the remote host reliably.
- A mail may be rejected by any host en route to the destination quietly unless it appears as authentic as possible.
- It is not realistic to expect a single smarthost to send mails of unrelated source mail addresses to the remote host reliably.

This is because:

- The SMTP port (25) connections from hosts serviced by the consumer grade network to the Internet are blocked.
- The SMTP port (25) connections to hosts serviced by the consumer grade network from the Internet are blocked.
- The outgoing messages from hosts serviced by the consumer grade network to the Internet can only be sent via the message submission port (587).
- Anti-spam techniques such as DomainKeys Identified Mail (DKIM), Sender_Policy_Framework (SPF), and Domain-based Message Authentication, Reporting and Conformance (DMARC) are widely used for the email filtering.
- The DomainKeys Identified Mail service may be provided for your mail sent through the smarthost.
- The smarthost may rewrite the source mail address in the message header to your mail account on the smarthost to prevent email address spoofing.

6.2.3 Historic mail service expectation

Some programs on Debian expect to access the `/usr/sbin/sendmail` command to send emails as their default or customized setting since the mail service on a UNIX system functioned historically as:

- An email is created as a text file.
- The email is handed to the `/usr/sbin/sendmail` command.

- For the destination address on the same host, the `/usr/sbin/sendmail` command makes local delivery of the email by appending it to the `/var/mail/$username` file.
  - Commands expecting this feature: `apt-listchanges`, `cron`, `at`, ...
- For the destination address on the remote host, the `/usr/sbin/sendmail` command makes remote transfer of the email to the destination host found by the DNS MX record using SMTP.
  - Commands expecting this feature: `popcon`, `reportbug`, `bts`, ...

6.2.4 Mail transport agent (MTA)

Debian mobile workstations can be configured just with full featured GUI based email clients without mail transfer agent (MTA) program after Debian 12 Bookworm.

Debian traditionally installed some MTA program to support programs expecting the `/usr/sbin/sendmail` command. Such MTA on mobile workstations must cope with Section 6.2.2 and Section 6.2.3.

For mobile workstations, the typical choice of MTA is either `exim4-daemon-light` or `postfix` with its installation option such as "Mail sent by smarthost; received via SMTP or fetchmail" selected. These are light weight MTAs that respect "`/etc/aliases`".

Tip
Configuring `exim4` to send the Internet mail via multiple corresponding smarthosts for multiple source email addresses is non-trivial. If you need such capability for some programs, set them up to use `msmtp` which is easy to set up for multiple source email addresses. Then leave main MTA only for a single email address.

package	popcon	size	description
`exim4-daemon-light`	V:200, I:209	1776	Exim4 mail transport agent (MTA: Debian default)
`exim4-daemon-heavy`	V:4.8, I:4.9	1954	Exim4 mail transport agent (MTA: flexible alternative)
`exim4-base`	V:205, I:214	1685	Exim4 documentation (text) and common files
`exim4-doc-html`	I:0.98	3809	Exim4 documentation (html)
`exim4-doc-info`	I:0.55	649	Exim4 documentation (info)
`postfix`	V:102, I:107	4274	Postfix mail transport agent (MTA: secure alternative)
`postfix-doc`	I:4.3	5028	Postfix documentation (html+text)
`sasl2-bin`	V:4.5, I:9.9	381	Cyrus SASL API implementation (supplement postfix for SMTP AUTH)
`cyrus-sasl2-doc`	I:0.68	2140	Cyrus SASL - documentation
`msmtp`	V:9, I:15	822	Light weight MTA
`msmtp-mta`	V:7.2, I:9.1	139	Light weight MTA (sendmail compatibility extension to `msmtp`)
`nullmailer`	V:7.4, I:7.7	483	Strip down MTA, no local mail
`ssmtp`	V:3.9, I:5.9	134	Strip down MTA, no local mail
`sendmail-bin`	V:10, I:10	1954	Full featured MTA (only if you are already familiar)
`git-email`	V:0.3, I:9.3	1229	git-send-email(1) program for sending series of patch emails

Table 6.3: List of basic mail transport agent related packages

6.2.4.1 The configuration of exim4

For the Internet mail via smarthost, you (re)configure `exim4-*` packages as the following.

```
$ sudo systemctl stop exim4
$ sudo dpkg-reconfigure exim4-config
```

Select "mail sent by smarthost; received via SMTP or fetchmail" for "General type of mail configuration".

Set "System mail name:" to its default as the FQDN (see Section 5.1.1).

Set "IP-addresses to listen on for incoming SMTP connections:" to its default as "127.0.0.1 ; ::1".

Unset contents of "Other destinations for which mail is accepted:".

Unset contents of "Machines to relay mail for:".

Set "IP address or host name of the outgoing smarthost:" to "smtp.hostname.dom:587".

Select "No" for "Hide local mail name in outgoing mail?". (Use "`/etc/email-addresses`" as in Section 6.2.4.3, instead.)

Reply to "Keep number of DNS-queries minimal (Dial-on-Demand)?" as one of the following.

- "No" if the system is connected to the Internet while booting.
- "Yes" if the system is **not** connected to the Internet while booting.

Set "Delivery method for local mail:" to "mbox format in /var/mail/".

Select "Yes" for "Split configuration into small files?:".

Create password entries for the smarthost by editing "`/etc/exim4/passwd.client`".

```
$ sudo vim /etc/exim4/passwd.client
 ...
$ cat /etc/exim4/passwd.client
^smtp.*\.hostname\.dom:username@hostname.dom:password
```

Configure exim4(8) with "`QUEUERUNNER='queueonly'`", "`QUEUERUNNER='nodaemon'`", etc. in "`/etc/default/exim4`" to minimize system resource usages. (optional)

Start `exim4` by the following.

```
$ sudo systemctl start exim4
```

The host name in "`/etc/exim4/passwd.client`" should not be the alias. You check the real host name with the following.

```
$ host smtp.hostname.dom
smtp.hostname.dom is an alias for smtp99.hostname.dom.
smtp99.hostname.dom has address 123.234.123.89
```

I use regex in "`/etc/exim4/passwd.client`" to work around the alias issue. SMTP AUTH probably works even if the ISP moves host pointed by the alias.

You can manually update `exim4` configuration by the following:

- Update `exim4` configuration files in "`/etc/exim4/`".
  - creating "`/etc/exim4/exim4.conf.localmacros`" to set MACROs and editing "`/etc/exim4/exim4.conf.temp` (non-split configuration)
  - creating new files or editing existing files in the "`/etc/exim4/exim4.conf.d`" subdirectories. (split configuration)
- Run "`systemctl reload exim4`".



Caution
Starting `exim4` takes long time if "No" (default value) was chosen for the debconf query of "Keep number of DNS-queries minimal (Dial-on-Demand)?" and the system is **not** connected to the Internet while booting.

Please read the official guide at: "`/usr/share/doc/exim4-base/README.Debian.gz`" and update-exim4.conf(8).



Warning
For all practical consideration, use SMTP with STARTTLS on port 587 or SMTPS (SMTP over SSL) on port 465, instead of plain SMTP on port 25.

6.2.4.2 The configuration of postfix with SASL

For the Internet mail via smarthost, you should first read postfix documentation and key manual pages.

command	function
postfix(1)	Postfix control program
postconf(1)	Postfix configuration utility
postconf(5)	Postfix configuration parameters
postmap(1)	Postfix lookup table maintenance
postalias(1)	Postfix alias database maintenance

Table 6.4: List of important postfix manual pages

You (re)configure `postfix` and `sasl2-bin` packages as follows.

```
$ sudo systemctl stop postfix
$ sudo dpkg-reconfigure postfix
```

Chose "Internet with smarthost".

Set "SMTP relay host (blank for none):" to "`[smtp.hostname.dom]:587`" and configure it by the following.

```
$ sudo postconf -e 'smtp_sender_dependent_authentication = yes'
$ sudo postconf -e 'smtp_sasl_auth_enable = yes'
$ sudo postconf -e 'smtp_sasl_password_maps = hash:/etc/postfix/sasl_passwd'
$ sudo postconf -e 'smtp_sasl_type = cyrus'
$ sudo vim /etc/postfix/sasl_passwd
```

Create password entries for the smarthost.

```
$ cat /etc/postfix/sasl_passwd
[smtp.hostname.dom]:587     username:password
$ sudo postmap hush:/etc/postfix/sasl_passwd
```

Start the `postfix` by the following.

```
$ sudo systemctl start postfix
```

Here the use of "`[`" and "`]`" in the `dpkg-reconfigure` dialog and "`/etc/postfix/sasl_passwd`" ensures not to check MX record but directly use exact hostname specified. See "Enabling SASL authentication in the Postfix SMTP client" in "`/usr/share/doc/postfix/html/SASL_README.html`".

file	function	application
/etc/mailname	default host name for (outgoing) mail	Debian specific, mailname(5)
/etc/email-addresses	host name spoofing for outgoing mail	exim(8) specific, exim4-config_files(5)
/etc/postfix/generic	host name spoofing for outgoing mail	postfix(1) specific, activated after postmap(1) command execution.
/etc/aliases	account name alias for incoming mail	general, activated after newaliases(1) command execution.

Table 6.5: List of mail address related configuration files

6.2.4.3 The mail address configuration

There are a few mail address configuration files for mail transport, delivery and user agents.

The **mailname** in the "/etc/mailname" file is usually a fully qualified domain name (FQDN) that resolves to one of the host's IP addresses. For the mobile workstation which does not have a hostname with resolvable IP address, set this **mailname** to the value of "hostname -f". (This is safe choice and works for both exim4-* and postfix.)

Tip
The contents of "/etc/mailname" is used by many non-MTA programs for their default behavior. For mutt, set "hostname" and "from" variables in ~/muttrc file to override the **mailname** value. For programs in the devscripts package, such as bts(1) and dch(1), export environment variables "$DEBFULLNAME" and "$DEBEMAIL" to override it.

Tip
The popularity-contest package normally send mail from root account with FQDN. You need to set MAILFROM in /etc/popularity-contest.conf as described in the /usr/share/popularity-contest/default.conf file. Otherwise, your mail will be rejected by the smarthost SMTP server. Although this is tedious, this approach is safer than rewriting the source address for all mails from root by MTA and should be used for other daemons and cron scripts.

When setting the **mailname** to "hostname -f", the spoofing of the source mail address via MTA can be realized by the following.

- "/etc/email-addresses" file for exim4(8) as explained in the exim4-config_files(5)
- "/etc/postfix/generic" file for postfix(1) as explained in the generic(5)

For postfix, the following extra steps are needed.

```
# postmap hash:/etc/postfix/generic
# postconf -e 'smtp_generic_maps = hash:/etc/postfix/generic'
# postfix reload
```

You can test mail address configuration using the following.

- exim(8) with -brw, -bf, -bF, -bV, … options
- postmap(1) with -q option.

Tip
Exim comes with several utility programs such as exiqgrep(8) and exipick(8). See "dpkg -L exim4-base|grep man8/" for available commands.

6.2.4.4 Basic MTA operations

There are several basic MTA operations. Some may be performed via sendmail(1) compatibility interface.

exim command	postfix command	description
`sendmail`	`sendmail`	read mails from standard input and arrange for delivery (`-bm`)
`mailq`	`mailq`	list the mail queue with status and queue ID (`-bp`)
`newaliases`	`newaliases`	initialize alias database (`-I`)
`exim4 -q`	`postqueue -f`	flush waiting mails (`-q`)
`exim4 -qf`	`postsuper -r ALL deferred; postqueue -f`	flush all mails
`exim4 -qff`	`postsuper -r ALL; postqueue -f`	flush even frozen mails
`exim4 -Mg queue_id`	`postsuper -h queue_id`	freeze one message by its queue ID
`exim4 -Mrm queue_id`	`postsuper -d queue_id`	remove one message by its queue ID
N/A	`postsuper -d ALL`	remove all messages

Table 6.6: List of basic MTA operation

Tip
It may be a good idea to flush all mails by a script in "`/etc/ppp/ip-up.d/*`".

6.3 The remote access server and utilities (SSH)

The Secure SHell (SSH) is the **secure** way to connect over the Internet. A free version of SSH called OpenSSH is available as `openssh-client` and `openssh-server` packages in Debian.

For the user, ssh(1) functions as a smarter and more secure telnet(1). Unlike `telnet` command, `ssh` command does not stop on the `telnet` escape character (initial default CTRL-]).

package	popcon	size	tool	description
`openssh-client`	V:682, I:997	3611	ssh(1)	Secure shell client
`openssh-server`	V:774, I:820	3615	sshd(8)	Secure shell server
`ssh-askpass`	I:15	103	ssh-askpass(1)	asks user for a pass phrase for ssh-add (plain X)
`ssh-askpass-gnome`	V:0.5, I:2.9	46	ssh-askpass-gnome(1)	asks user for a pass phrase for ssh-add (GNOME)
`ssh-askpass-fullscreen`	V:0.09, I:0.40	41	ssh-askpass-fullscreen(1)	asks user for a pass phrase for ssh-add (GNOME) with extra eye candy
`shellinabox`	V:0.61, I:0.94	528	shellinaboxd(1)	web server for browser accessible VT100 terminal emulator

Table 6.7: List of remote access server and utilities

Although `shellinabox` is not a SSH program, it is listed here as an interesting alternative for the remote terminal access.

See also Section 7.9 for connecting to remote X client programs.



Caution
See Section 4.6.3 if your SSH is accessible from the Internet.

Tip
Please use the screen(1) program to enable remote shell process to survive the interrupted connection (see Section 9.1.2).

6.3.1 Basics of SSH

The OpenSSH SSH daemon supports SSH protocol 2 only.

Please read "`/usr/share/doc/openssh-client/README.Debian.gz`", ssh(1), sshd(8), ssh-keygen(1), ssh-add(1) and ssh-agent(1).



Warning
"`/etc/ssh/sshd_not_to_be_run`" must not be present if one wishes to run the OpenSSH server. Don't enable rhost based authentication (`HostbasedAuthentication` in `/etc/ssh/sshd_config`).

configuration file	description of configuration file
`/etc/ssh/ssh_config`	SSH client defaults, see ssh_config(5)
`/etc/ssh/sshd_config`	SSH server defaults, see sshd_config(5)
`~/.ssh/authorized_keys`	default public SSH keys that clients use to connect to this account on this SSH server
`~/.ssh/id_rsa`	secret SSH-2 RSA key of the user
`~/.ssh/id_`*`key-type-name`*	secret SSH-2 *key-type-name* key such as `ecdsa`, `ed25519`, ... of the user

Table 6.8: List of SSH configuration files

The following starts an ssh(1) connection from a client.

command	description
`ssh username@hostname.domain.ext`	connect with default mode
`ssh -v username@hostname.domain.ext`	connect with default mode with debugging messages
`ssh -o PreferredAuthentications=password username@hostname.domain.ext`	force to use password with SSH version 2
`ssh -t username@hostname.domain.ext passwd`	run `passwd` program to update password on a remote host

Table 6.9: List of SSH client startup examples

6.3.2 User name on the remote host

If you use the same user name on the local and the remote host, you can eliminate typing "`username@`".

Even if you use different user name on the local and the remote host, you can eliminate it using "`~/.ssh/config`". For Debian Salsa service with account name "`foo-guest`", you set "`~/.ssh/config`" to contain the following.

```
Host salsa.debian.org people.debian.org
User foo-guest
```

6.3.3 Connecting without remote passwords

One can avoid having to remember passwords for remote systems by using "`PubkeyAuthentication`" (SSH-2 protocol).

On the remote system, set the respective entries, "`PubkeyAuthentication yes`", in "`/etc/ssh/sshd_config`".

Generate authentication keys locally and install the public key on the remote system by the following.

```
$ ssh-keygen -t rsa
$ cat .ssh/id_rsa.pub | ssh user1@remote "cat - >>.ssh/authorized_keys"
```

You can add options to the entries in "`~/.ssh/authorized_keys`" to limit hosts and to run specific commands. See sshd(8) "AUTHORIZED_KEYS FILE FORMAT".

6.3.4 Dealing with alien SSH clients

There are some free SSH clients available for other platforms.

environment	free SSH program
Windows	puTTY (PuTTY: a free SSH and Telnet client) (GPL)
Windows (cygwin)	SSH in cygwin (Cygwin: Get that Linux feeling - on Windows) (GPL)
Mac OS X	OpenSSH; use `ssh` in the Terminal application (GPL)

Table 6.10: List of free SSH clients for other platforms

6.3.5 Setting up ssh-agent

It is safer to protect your SSH authentication secret keys with a pass phrase. If a pass phrase was not set, use "`ssh-keygen -p`" to set it.

Place your public SSH key (e.g. "`~/.ssh/id_rsa.pub`") into "`~/.ssh/authorized_keys`" on a remote host using a password-based connection to the remote host as described above.

```
$ ssh-agent bash
$ ssh-add ~/.ssh/id_rsa
Enter passphrase for /home/username/.ssh/id_rsa:
Identity added: /home/username/.ssh/id_rsa (/home/username/.ssh/id_rsa)
```

No remote password needed from here on for the next command.

```
$ scp foo username@remote.host:foo
```

Press ^D to terminating ssh-agent session.

For the X server, the normal Debian startup script executes `ssh-agent` as the parent process. So you only need to execute `ssh-add` once. For more, read ssh-agent(1) and ssh-add(1).

6.3.6 Sending a mail from a remote host

If you have an SSH shell account on a server with proper DNS settings, you can send a mail generated on your workstation as an email genuinely sent from the remote server.

```
$ ssh username@example.org /usr/sbin/sendmail -bm -ti -f "username@example.org" < mail_data ↩
    .txt
```

6.3.7 Port forwarding for SMTP/POP3 tunneling

To establish a pipe to connect to port 25 of `remote-server` from port 4025 of `localhost`, and to port 110 of `remote-server` from port 4110 of `localhost` through `ssh`, execute on the local host as the following.

```
# ssh -q -L 4025:remote-server:25 4110:remote-server:110 username@remote-server
```

This is a secure way to make connections to SMTP/POP3 servers over the Internet. Set the "`AllowTcpForwarding`" entry to "`yes`" in "`/etc/ssh/sshd_config`" of the remote host.

6.3.8 How to shutdown the remote system on SSH

You need to protect the process doing "`shutdown -h now`" (see Section 1.1.8) from the termination of SSH using the at(1) command (see Section 9.4.13) by the following.

```
# echo "shutdown -h now" | at now
```

Running "`shutdown -h now`" in screen(1) (see Section 9.1.2) session is another way to do the same.

6.3.9 Troubleshooting SSH

If you have problems, check the permissions of configuration files and run `ssh` with the "`-v`" option.

Use the "`-p`" option if you are root and have trouble with a firewall; this avoids the use of server ports 1 — 1023.

If `ssh` connections to a remote site suddenly stop working, it may be the result of tinkering by the sysadmin, most likely a change in "`host_key`" during system maintenance. After making sure this is the case and nobody is trying to fake the remote host by some clever hack, one can regain a connection by removing the "`host_key`" entry from "`~/.ssh/known_hosts`" on the local host.

6.4 The print server and utilities

In the old Unix-like system, the BSD Line printer daemon (lpd) was the standard and the standard print out format of the classic free software was PostScript (PS). Some filter system was used along with Ghostscript to enable printing to the non-PostScript printer. See Section 11.4.1.

In the modern Debian system, the Common UNIX Printing System (CUPS) is the de facto standard and the standard print out format of the modern free software is Portable Document Format (PDF).

The CUPS uses Internet Printing Protocol (IPP). The IPP is the cross-platform de facto standard for remote printing with bi-directional communication capability.

Thanks to the file format dependent auto-conversion feature of the CUPS system, simply feeding any data to the `lpr` command should generate the expected print output. (In CUPS, `lpr` can be enabled by installing the `cups-bsd` package.)

The Debian system has some notable packages for the print servers and utilities.

package	popcon	size	port	description
lpr	V:1.9, I:2.2	378	printer (515)	BSD lpr/lpd (Line printer daemon)
cups	V:144, I:447	1088	IPP (631)	Internet Printing CUPS server
cups-client	V:152, I:459	429	, ,	System V printer commands for CUPS: lp(1), lpstat(1), lpoptions(1), cancel(1), lpmove(8), lpinfo(8), lpadmin(8), …
cups-bsd	V:34, I:182	131	, ,	BSD printer commands for CUPS: lpr(1), lpq(1), lprm(1), lpc(8)
printer-driver-gutenprint	V:11, I:52	1117	Not applicable	printer drivers for CUPS

Table 6.11: List of print servers and utilities

Tip
You can configure CUPS system by pointing your web browser to "http://localhost:631/" .

6.5 Other network application servers

Here are other network application servers.

package	popcon	size	protocol	description
telnetd	V:0.3, I:1.6	56	TELNET	TELNET server
nfs-kernel-server	V:47, I:54	874	NFS	Unix file sharing
samba	V:110, I:123	5182	SMB	Windows file and printer sharing
netatalk	V:0.62, I:0.85	933	ATP	Apple/Mac file and printer sharing (AppleTalk)
proftpd-basic	V:3.3, I:8.4	452	FTP	General file download
apache2	V:175, I:211	587	HTTP	General web server
squid	V:9.0, I:9.4	9583	, ,	General web proxy server
bind9	V:33, I:35	904	DNS	IP address for other hosts
kea	I:0.51	245	DHCP	IP address of client itself

Table 6.12: List of other network application servers

Common Internet File System Protocol (CIFS) is the same protocol as Server Message Block (SMB) and is used widely by Microsoft Windows.

Tip
See Section 4.5.2 for integration of server systems.

Tip
The hostname resolution is usually provided by the DNS server. For the host IP address dynamically assigned by DHCP, Dynamic DNS can be set up for the hostname resolution using `bind9` and `kea` as described in the DDNS page on the Debian wiki.

Tip
Use of proxy server such as `squid` is much more efficient for saving bandwidth than use of local mirror server with the full Debian archive contents.

6.6 Other network application clients

Here are other network application clients.

package	popcon	size	protocol	description
`netcat-traditional`	V:55, I:906	139	TCP/IP	TCP/IP swiss army knife
`netcat-openbsd`	V:22, I:119	105	TCP/IP	TCP/IP swiss army knife with support for IPv6, proxies, and Unix sockets
`openssl`	V:846, I:996	2564	SSL	Secure Socket Layer (SSL) binary and related cryptographic tools
`stunnel4`	V:6.6, I:9.9	24	, ,	universal SSL Wrapper
`telnet`	V:10, I:190	56	TELNET	TELNET client
`nfs-common`	V:147, I:191	1158	NFS	Unix file sharing
`smbclient`	V:25, I:206	2117	SMB	MS Windows file and printer sharing client
`cifs-utils`	V:32, I:118	368	, ,	mount and umount commands for remote MS Windows file
`wget`	V:181, I:982	3784	HTTP and FTP	web downloader
`curl`	V:260, I:727	520	, ,	, ,
`transmission-gtk`	V:9, I:143	6259	BitTorrent	BitTorrent client (GTK)
`transmission-qt`	V:0.6, I:2.6	6213	, ,	BitTorrent client (Qt)
`ktorrent`	V:1.3, I:5.0	5031	, ,	BitTorrent client (Qt)
`qbittorrent`	V:8, I:23	15378	, ,	BitTorrent client (Qt)
`bind9-host`	V:124, I:940	136	DNS	host(1) from bind9, "`Priority: standard`"
`dnsutils`	V:5, I:146	23	, ,	dig(1) from bind, "`Priority: standard`"
`ldap-utils`	V:11, I:57	790	LDAP	obtain data from LDAP server

Table 6.13: List of network application clients

6.7 The diagnosis of the system daemons

The `telnet` program enables manual connection to the system daemons and its diagnosis.

For testing plain POP3 service, try the following

```
$ telnet mail.ispname.net pop3
```

For testing the TLS/SSL enabled POP3 service by some ISPs, you need TLS/SSL enabled `telnet` client by the `telnet-ssl` or `openssl` packages.

```
$ telnet -z ssl pop.gmail.com 995
```

```
$ openssl s_client -connect pop.gmail.com:995
```

The following RFCs provide required knowledge to each system daemon.

RFC	description
rfc1939 and rfc2449	POP3 service
rfc3501	IMAP4 service
rfc2821 (rfc821)	SMTP service
rfc2822 (rfc822)	Mail file format
rfc2045	Multipurpose Internet Mail Extensions (MIME)
rfc819	DNS service
rfc2616	HTTP service
rfc2396	URI definition

Table 6.14: List of popular RFCs

The port usage is described in "`/etc/services`".

Chapter 7

GUI System

7.1 GUI desktop environment

There are several choices for the full featured GUI desktop environment on the Debian system.

task package	popcon	size	description
task-gnome-desktop	I:192	9	GNOME desktop environment
task-xfce-desktop	I:88	9	Xfce desktop environment
task-kde-desktop	I:94	6	KDE Plasma desktop environment
task-mate-desktop	I:30	9	MATE desktop environment
task-cinnamon-desktop	I:35	9	Cinnamon desktop environment
task-lxde-desktop	I:19	9	LXDE desktop environment
task-lxqt-desktop	I:16	9	LXQt desktop environment
task-gnome-flashback-desktop	I:9.6	6	GNOME Flashback desktop environment

Table 7.1: List of desktop environment

Tip
Dependency packages selected by a task metapackage may be out of sync with the latest package transition state under the Debian `unstable`/`testing` environment. For `task-gnome-desktop`, you may need to adjust package selections as follows:

- Start aptitude(8) as `sudo aptitude -u`.
- Move cursor to ”Tasks” and press ”Enter”.
- Move cursor to ”End-user” press ”Enter”.
- Move cursor to ”GNOME” press ”Enter”.
- Move cursor to `task-gnome-desktop` and press ”Enter”.
- Move cursor to ”Depends” and press ”m” (manually selected).
- Move cursor to ”Recommends” and press ”m” (manually selected).
- Move cursor to ”`task-gnome-desktop` and press ”-”. (drop)
- Adjust selected packages while dropping problematic ones causing package conflicts.
- Press ”g” to start install.

This chapter will focus mostly on the default desktop environment of Debian: `task-gnome-desktop` offering GNOME on wayland.

7.2 GUI communication protocol

GUI communication protocol used on the GNOME desktop can be:

- Wayland (display server protocol) (native)
- X Window System core protocol (via `xwayland`)

Please check freedesktop.org site for how Wayland architecture is different from X Window architecture.

From user’s perspective, differences can be colloquially summarized as:

- Wayland is a same-host GUI communication protocol: new, simpler, faster, no setuid root binary
- X Window is a network-capable GUI communication protocol: traditional, complex, slower, setuid root binary

For applications using Wayland protocol, the access to their display contents from a remote host is supported by the VNC or RDP. See Section 7.8

Modern X servers have the MIT Shared Memory Extension and communicate with their local X clients using the local shared memory. This bypasses the network transparent Xlib interprocess communication channel and gains performance. This situation was the background of creating Wayland as a local-only GUI communication protocol.

Using the `xeyes` program started from the GNOME terminal, you can check GUI communication protocol used by each GUI application.

```
$ xeyes
```

- If the mouse cursor is on an application such as ”GNOME terminal” which uses Wayland display server protocol, eyes don’t move with the mouse cursor.

- If the mouse cursor is on an application such as ”xterm” which uses X Window System core protocol, eyes move with the mouse cursor exposing not-so-isolated nature of X Window architecture.

As of April 2021, many popular GUI applications such as GNOME and LibreOffice (LO) applications have been migrated to the Wayland display server protocol. I see `xterm`, `gitk`, `chromium`, `firefox`, `gimp`, `dia`, and KDE applications still use X Window System core protocol.

Note
For both the xwayland on Wayland or the native X Window System, the old X server configuration file ”`/etc/X11/xorg.conf`” shouldn’t exist on the system. The graphics and input devices are now configured by the kernel with DRM, KMS, and udev. The native X server has been rewritten to use them. See ”modedb default video mode support” in the Linux kernel documentation.

7.3 GUI infrastructure

Here are notable GUI infrastructure packages for the GNOME on Wayland environment.

package	popcon	package size	description
`mutter`	V:1, I:22	216	GNOME’s mutter window manager **[auto]**
`xwayland`	V:256, I:335	2652	An X server running on top of wayland **[auto]**
`gnome-remote-desktop`	V:147, I:241	2743	Remote desktop daemon for GNOME using PipeWire **[auto]**
`gnome-tweaks`	V:17, I:232	1145	Advanced configuration settings for GNOME
`gnome-shell-extension-prefs`	V:8, I:117	73	Tool to enable / disable GNOME Shell extensions

Table 7.2: List of notable GUI infrastructure packages

Here, ”**[auto]**” means that these packages are automatically installed when `task-gnome-desktop` is installed.

Tip
`gnome-tweaks` is the indispensable configuration utility. For example:

- You can force ”Over-Amplification” of sound volume from ”General”.
- You can force ”Caps” to become ”Esc” from ”Keyboard & Mouse” -> ”Keyboard” -> ”Additional Layout Option”.

Tip
Detail features of GNOME desktop environment can be configured with utilities started by typing ”`settings`”, ”`tweaks`”, or ”`extensions`” after pressing `Super`-key.

7.4 GUI applications

Many useful GUI applications are available on Debian now. Installing software packages such as `scribus` (KDE) on GNOME desktop environment are quite acceptable since corresponding functionality is not available under GNOME desktop environment. But installing too many packages with duplicated functionalities may clutter your system.

Here is a list of GUI applications which caught my eyes.

package	popcon	package size	type	description
evolution	V:27, I:229	489	GNOME	Personal information Management (groupware and email)
thunderbird	V:39, I:102	303329	GTK	Email client (Mozilla Thunderbird)
kontact	V:1, I:10	2360	KDE	Personal information Management (groupware and email)
libreoffice-writer	V:105, I:422	34232	LO	word processor
abiword	V:0.8, I:4.7	3596	GNOME	word processor
calligrawords	V:0.2, I:2.7	7144	KDE	word processor
scribus	V:1, I:12	32942	KDE	desktop publishing editor to edit PDF files
glabels	V:0.3, I:2.5	1283	GNOME	label editor
libreoffice-calc	V:101, I:419	29620	LO	spreadsheet
gnumeric	V:3, I:10	9945	GNOME	spreadsheet
calligrasheets	V:0.1, I:1.7	14014	KDE	spreadsheet
libreoffice-impress	V:88, I:417	2499	LO	presentation
calligrastage	V:0.1, I:1.6	6237	KDE	presentation
libreoffice-base	V:21, I:67	5130	LO	database management
kexi	V:0.05, I:0.80	7565	KDE	database management
libreoffice-draw	V:89, I:418	11418	LO	vector graphics editor (draw)
inkscape	V:12, I:76	113353	GNOME	vector graphics editor (draw)
karbon	V:0.1, I:2.1	4029	KDE	vector graphics editor (draw)
dia	V:1, I:16	3846	GTK	flowchart and diagram editor
gimp	V:34, I:211	33315	GTK	bitmap graphics editor (paint)
shotwell	V:14, I:246	6334	GTK	digital photo organizer
digikam	V:1.5, I:8.3	333	KDE	digital photo organizer
darktable	V:4, I:11	36092	GTK	lighttable and darkroom for photographers
gnome-video-trimmer	V:0.1, I:1.0	1665	GNOME	lossless cutting of video files
kdenlive	V:3, I:21	22134	KDE	non-linear video editor
shotcut	V:1.0, I:7.5	8270	Qt-app	non-linear video editor
openshot-qt	V:0.9, I:8.8	196007	Qt-app	non-linear video editor
flowblade	V:0.2, I:2.2	42213	GTK	non-linear video editor
planner	V:0.2, I:4.2	1400	GNOME	project management
calligraplan	V:0.2, I:2.9	23545	KDE	project management
gnucash	V:2.0, I:6.7	29434	GNOME	personal accounting
homebank	V:0.3, I:1.5	3194	GTK	personal accounting
kmymoney	V:0.4, I:1.9	19297	KDE	personal accounting
frescobaldi	V:0.3, I:1.5	11102	Qt-app	LilyPond sheet music text editor
librecad	V:1, I:13	9164	Qt-app	computer-aided design (CAD) system (2D)
freecad	V:1, I:20	112	Qt-app	computer-aided design (CAD) system (3D)
kicad	V:3, I:14	224155	GTK	electronic schematic and PCB design software
xsane	V:9, I:128	1512	GTK	scanner frontend
libreoffice-math	V:84, I:420	1982	LO	mathematical equation/formula editor
calibre	V:7, I:24	71216	KDE	e-book converter and library management
fbreader	V:0.7, I:5.4	3827	GTK	e-book reader
evince	V:76, I:284	953	GNOME	document(pdf) viewer
okular	V:42, I:130	4544	KDE	document(pdf) viewer
x11-apps	V:31, I:443	2461	pure X-app	xeyes(1), etc.
x11-utils	V:228, I:555	651	pure X-app	xev(1), xwininfo(1), etc.

Table 7.3: List of notable GUI applications

7.5 User directories

Default names for user directories such as "`~/Desktop`", "`~/Documents`", ..., used by the Desktop environment depend on the locale used for the system installation. You can reset them to the English ones by:

```
$ LANGUAGE=C xdg-user-dirs-update --force
```

Then you manually move all the data to the newer directories. See xdg-user-dirs-update(1).

You can also set them to any names by editing "`~/.config/user-dirs.dirs`". See user-dirs.dirs(5).

7.6 Fonts

Many useful scalable fonts are available for users on Debian. User's concern is how to avoid redundancy and how to configure parts of installed fonts to be disabled. Otherwise, useless font choices may clutter your GUI application menus.

Debian system uses FreeType 2.0 library to rasterise many scalable font formats for screen and print:

- Type 1 (PostScript) fonts which use cubic Bézier curves (almost obsolete format)
- TrueType fonts which use quadratic Bézier curves (good choice format)
- OpenType fonts which use cubic Bézier curves (best choice format)

7.6.1 Basic fonts

The following table is compiled in the hope to help users to choose appropriate scalable fonts with clear understanding of the metric compatibility and the glyph coverage. Most fonts cover all Latin, Greek, and Cyril characters. The final choice of activated fonts can also be affected by your aesthetics. These fonts can be used for the screen display or for the paper printing.

Here:

- "MCM" stands for "metric compatible with fonts provided by Microsoft"
- "MCMATC" stands for "metric compatible with fonts provided by Microsoft: Arial, Times New Roman, Courier New"
- "MCAHTC" stands for "metric compatible with fonts provided by Adobe: Helvetica, Times, Courier"
- Numbers in font type columns stands for the rough relative "M" width for the same point size font.
- "P" in mono font type columns stands for its usability for programming having clearly distinguishable "0"/"O" and "1"/"l"/"|".
- The `ttf-mscorefonts-installer` package downloads Microsoft's "Core fonts for the Web" and installs Arial, Times New Roman, Courier New, Verdana, ... These installed font data are non-free data.

Many free Latin fonts have their lineage traced to URW Nimbus family or Bitstream Vera.

Tip
If your locale needs fonts not covered well by the above fonts, please use aptitude to check under task packages listed under "Tasks" -> "Localization". The font packages listed as "Depends:" or "Recommends:" in the localization task packages are the primary candidates.

package	popcon	size	sans	serif	mono	note on font
fonts-cantarell	V:174, I:288	224	59	-	-	Cantarell (GNOME 3, display)
fonts-noto	I:150	30	61	63	40	Noto fonts (Google, multi-lingual with CJK)
fonts-dejavu	I:385	34	58	68	40	DejaVu (GNOME 2, MCM:Verdana, extended Bitstream Vera)
fonts-liberation2	V:40, I:156	15	56	60	40	Liberation fonts for LibreOffice (Red Hat, MCMATC)
fonts-croscore	V:20, I:36	5260	56	60	40	Chrome OS: Arimo, Tinos and Cousine (Google, MCMATC)
fonts-crosextra-carlito	V:20, I:90	2696	57	-	-	Chrome OS: Carlito (Google, MCM:Calibri)
fonts-crosextra-caladea	V:12, I:83	347	-	55	-	Chrome OS: Caladea (Google, MCM:Cambria) (Latin only)
fonts-freefont-ttf	V:78, I:203	14460	57	59	40	GNU FreeFont (extended URW Nimbus)
fonts-quicksand	V:210, I:452	392	56	-	-	Debian task-desktop, Quicksand (display, Latin only)
fonts-hack	V:34, I:136	2509	-	-	40 P	A typeface designed for source code Hack (Facebook)
fonts-sil-gentiumplus	V:0, I:28	14345	-	54	-	Gentium SIL
fonts-sil-charis	V:1, I:28	6704	-	59	-	Charis SIL
fonts-urw-base35	V:187, I:534	15558	56	60	40	URW Nimbus (Nimbus Sans, Roman No. 9 L, Mono L, MCAHTC)
fonts-ubuntu	V:2.2, I:4.7	4339	58	-	33 P	Ubuntu fonts (display)
fonts-terminus	V:0.3, I:3.9	451	-	-	33	Cool retro terminal fonts
ttf-mscorefonts-installer	V:1, I:40	85	56?	60	40	Downloader of Microsoft non-free fonts (see below)

Table 7.4: List of notable TrueType and OpenType fonts

package	popcon	size	description
`libfreetype6`	V:589, I:997	1050	FreeType font rasterization library
`libfontconfig1`	V:577, I:808	344	Fontconfig font configuration library
`fontconfig`	V:471, I:693	415	`fc-*`: CLI commands for Fontconfig
`font-manager`	V:2.3, I:6.6	1116	Font Manager: GUI command for Fontconfig
`nautilus-font-manager`	V:0.7, I:0.38	38	Nautilus extension for Font Manager

Table 7.5: List of notable font environment and related packages

7.6.2 Font rasterization

Debian uses FreeType to rasterize fonts. Its font choice infrastructure is provided by the Fontconfig font configuration library.

Tip
Some font packages such as `fonts-noto*` install too many fonts. You may also want to keep some font packages installed but disabled under the normal use situation. The multiple glyphs are expected for some Unicode code points due to Han unification and unwanted gliphs may be chosen by the unconfigured Fontconfig library. One of the most annoying case is "U+3001 IDEOGRAPHIC COMMA" and "U+3002 IDEOGRAPHIC FULL STOP" among CJK countries. You can avoid this problematic situation easily by configuring font availability using Font Manager GUI (font-manager).

You can list font configuration state from the command line, too.

- "`fc-match(1)`" for fontconfig font default
- "`fc-list(1)`" for available fontconfig fonts

You can configure font configuration state from the text editor but this is non-trivial. See fonts.conf(5).

7.7 Sandbox

Many mostly GUI applications on Linux are available in binary formats from non-Debian sources.

- AppImage -- Linux apps that run anywhere
- FLATHUB -- Apps for Linux, right here
- snapcraft -- The app store for Linux



Warning
Binaries from these sites may include proprietary non-free software packages.

There is some raison d'être for these binary format distributions for Free Software aficionados using Debian since these can accommodate clean set of libraries used for each application by the respective upstream developer independent of the ones provided by Debian.

The inherent risk of running external binaries can be reduced by using the sandbox environment which leverages modern Linux security features (see Section 4.7.5).

- For binaries from AppImage and some upstream sites, run them in firejail with manual configuration.
- For binaries from FLATHUB, run them in Flatpak . (No manual configuration required.)
- For binaries from snapcraft, run them in Snap . (No manual configuration required. Compatible with daemon programs.)

The `xdg-desktop-portal` package provides a standardized API to common desktop features. See xdg-desktop-portal (flatpak) and xdg-desktop-portal (snap) .

This sandbox environment technology is very much like apps on smart phone OS where apps are executed under controlled resource accesses.

Some large GUI applications such as web browsers on Debian also use sandbox environment technology internally to make them more secure.

package	popcon	size	description
flatpak	V:104, I:115	9293	Flatpak application deployment framework for desktop apps
gnome-software-plugin-flatpak	V:29, I:41	291	Flatpak support for GNOME Software
snapd	V:58, I:62	62738	Daemon and tooling that enable snap packages
gnome-software-plugin-snap	V:1.4, I:2.2	144	Snap support for GNOME Software
xdg-desktop-portal	V:373, I:439	2291	desktop integration portal for Flatpak and Snap
xdg-desktop-portal-gtk	V:337, I:438	715	xdg-desktop-portal backend for gtk (GNOME)
xdg-desktop-portal-kde	V:81, I:112	3333	xdg-desktop-portal backend for Qt (KDE)
xdg-desktop-portal-wlr	V:2.4, I:6.6	159	xdg-desktop-portal backend for wlroots (Wayland)
firejail	V:1.2, I:4.4	1827	a SUID security sandbox program firejail for use with AppImage

Table 7.6: List of notable sandbox environment and related packages

7.8 Remote desktop

7.9 X server connection

There are several ways to connect from an application on a remote host to the X server including `xwayland` on the local host.

7.9.1 X server local connection

Access to the local X server by the local applications which use X core protocol can be locally connected through a local UNIX domain socket. This can be authorized by the authority file holding access cookie. The authority file location is identified by the "`$XAUTHORITY`" environment variable and X display is identified by the "`$DISPLAY`" environment variable. Since these are normally set automatically, no special action is needed, e.g. "`gitk`" as the following.

```
username $ gitk
```

Note
For `xwayland`, `XAUTHORITY` holds value like "`/run/user/1000/.mutter-Xwaylandauth.YVSU30`".

7.9.2 X server remote connection

Access to the local X server display from the remote applications which use X core protocol is supported by using the X11 forwarding feature.

- Open an `gnome-terminal` on the local host.
- Run ssh(1) with `-X` option to establish a connection with the remote site as the following.

```
localname @ localhost $ ssh -q -X loginname@remotehost.domain
Password:
```

package	popcon	size	protocols	description
gnome-remote-desktop	V:17, I:241	2743	RDP	GNOME Remote Desktop server
xrdp	V:28, I:31	4757	RDP	xrdp, Remote Desktop Protocol (RDP) server
x11vnc	V:9, I:46	1863	RFB (VNC)	x11vnc, Remote Framebuffer Protocol (VNC) server
tigervnc-standalone-server	V:5, I:15	3124	RFB (VNC)	TigerVNC, Remote Framebuffer Protocol (VNC) server
gnome-connections	V:6, I:149	1726	RDP, RFB (VNC)	GNOME remote desktop client
vinagre	V:1, I:21	4249	RDP, RFB (VNC), SPICE, SSH	Vinagre: GNOME remote desktop client
remmina	V:16, I:59	983	RDP, RFB (VNC), SPICE, SSH, ...	Remmina: GTK remote desktop client
krdc	V:2, I:15	4144	RDP, RFB (VNC)	KRDC: KDE remote desktop client
virt-viewer	V:4, I:37	1278	RFB (VNC), SPICE	Virtual Machine Manager's GUI display client of guest OS

Table 7.7: List of notable remote access server

package	popcon	size	command	description
openssh-server	V:774, I:820	3615	`sshd` with option `X11-forwarding`	SSH server (secure)
openssh-client	V:682, I:997	3611	`ssh -X`	SSH client (secure)
xauth	V:192, I:974	81	`xauth`	X authority file utility
x11-xserver-utils	V:373, I:533	559	`xhost`	server access control for X

Table 7.8: List of connection methods to the X server

- Run an X application command, e.g. "`gitk`", on the remote site as the following.

```
loginname @ remotehost $ gitk
```

This method can display the output from a remote X client as if it were locally connected through a local UNIX domain socket.

See Section 6.3 for SSH/SSHD.



Warning
A remote TCP/IP connection to the X server is disabled by default on the Debian system for security reasons. Don't enable them by simply setting "`xhost +`" nor by enabling XDMCP connection, if you can avoid it.

7.9.3 X server chroot connection

Access to the X server by the applications which use X core protocol and run on the same host but in an environment such as chroot where the authority file is not accessible, can be authorized securely with `xhost` by using the User-based access, e.g. "`gitk`" as the following.

```
username $ xhost + si:localuser:root ; sudo chroot /path/to
# cd /src
# gitk
# exit
username $ xhost -
```

7.10 Clipboard

For clipping text to clipboard, see Section 1.4.4.

For clipping graphics to clipboard, see Section 11.6.

Some CLI commands can manipulate character clipboard (PRIMARY and CLIPBOARD), too.

package	popcon	package size	target	description
`xsel`	V:8, I:42	55	X	command line interface to X selections (clipboard)
`xclip`	V:18, I:78	62	X	command line interface to X selections (clipboard)
`wl-clipboard`	V:10, I:27	174	Wayland	`wl-copy` `wl-paste`: command line interface to Wayland clipboard
`gpm`	V:8.3, I:9.0	524	Linux console	a daemon that captures mouse events on Linux console

Table 7.9: List of programs related to manipulating character clipboard

Chapter 8

I18N and L10N

Multilingualization (M17N) or Native Language Support for an application software is done in 2 steps.

- Internationalization (I18N): To make a software potentially handle multiple locales.
- Localization (L10N): To make a software handle an specific locale.

Tip
There are 17, 18, or 10 letters between ”m” and ”n”, ”i” and ”n”, or ”l” and ”n” in multilingualization, internationalization, and localization which correspond to M17N, I18N, and L10N. See Internationalization and localization for details.

8.1 The locale

The behavior of programs supporting internationalization are configured by the environment variable ”`$LANG`” to support localization. Actual support of locale dependent features by the `libc` library requires to install `locales` or `locales-all` packages. The `locales` package requires to be initialized properly.

If neither `locales` or `locales-all` package are installed, support of locale features are lost and system uses US English messages and handles data as **ASCII**. This behavior is the same way as ”`$LANG`” is set by ”`LANG=`”, ”`LANG=C`”, or ”`LANG=POSIX`”.

The modern software such as GNOME and KDE are multilingualized. They are internationalized by making them handle UTF-8 data and localized by providing their translated messages through the gettext(1) infrastructure. Translated messages may be provided as separate localization packages.

The current Debian desktop GUI system normally sets the locale under GUI environment as ”`LANG=xx_YY.UTF-8`”. Here, ”`xx`” is ISO 639 language codes and ”YY” is ISO 3166 country codes. These values are set by the desktop configuration GUI dialogue and change the program behavior. See Section 1.5.2

8.1.1 Rationale for UTF-8 locale

The simplest representation of the text data is **ASCII** which is sufficient for English and uses less than 127 characters (representable with 7 bits).

Even plain English text may contain non-ASCII characters, e.g. slightly curly left and right quotation marks are not available in ASCII.

```
b''“b''double quoted textb''”b'' is not "double quoted ASCII"
b''‘b''single quoted textb''’b'' is not 'single quoted ASCII'
```

In order to support more characters, many character sets and encoding systems have been used to support many languages (see Table 11.2).

Unicode character set can represent practically all characters known to human with 21 bit code point range (i.e., 0 to 10FFFF in hexadecimal notation).

Text encoding system UTF-8 fits Unicode code points into a sensible 8 bit data stream mostly compatible with the ASCII data processing system. This makes **UTF-8** the modern preferred choice. **UTF** stands for Unicode Transformation Format. When ASCII plain text data is converted to UTF-8 one, it has exactly the same content and size as the original ASCII one. So you loose nothing by deploying UTF-8 locale.

Under UTF-8 locale with the compatible application program, you can display and edit any foreign language text data as long as required fonts and input methods are installed and enabled. For example under "`LANG=fr_FR.UTF-8`" locale, gedit(1) (text editor for the GNOME desktop) can display and edit Chinese character text data while presenting menus in French.

Tip
Both the new standard "`en_US.UTF-8`" locale and the old standard "`C`"/"`POSIX`" locale use the standard US English message, they have subtle differences in sorting order etc. If you want to handle not only ASCII characters but also handle all UTF-8 encoded characters gracefully while maintaining the old "`C`" local behavior, use the non-standard "`C.UTF-8`" locale on Debian.

Note
Some programs consume more memory after supporting I18N. This is because they are coded to use UTF-32(UCS4) internally to support Unicode for speed optimization and consume 4 bytes per each ASCII character data independent of locale selected. Again, you loose nothing by deploying UTF-8 locale.

8.1.2 The reconfiguration of the locale

In order for the system to access a particular locale, the locale data must be compiled from the locale database.

The `locales` package does **not** come with pre-compiled locale data. You need to configure it as:

```
# dpkg-reconfigure locales
```

This process involves 2 steps.

1. Select all required locale data to be compiled into the binary form. (Please make sure to include at least one UTF-8 locale)
2. Set the system wide default locale value by creating "`/etc/default/locale`" for use by PAM (see Section 4.5).

The system wide default locale value set in "`/etc/default/locale`" may be overridden by the GUI configuration for GUI applications.

Note
Actual traditional encoding system can be identified by "`/usr/share/i18n/SUPPORTED`". Thus, the "`LANG=en_US`" is "`LANG=en_US.ISO-8859-1`".

The `locales-all` package comes with pre-compiled locale data for all locale data. Since it doesn't create "`/etc/default/`" you may still need to install the `locales` package, too.

Tip
The `locales` package of some Debian derivative distributions come with pre-compiled locale data for all locale data. You need to install both `locales` and `locales-all` packages on Debian to emulate such system environment.

8.1.3 Filename encoding

For cross platform data exchanges (see Section 10.1.7), you may need to mount some filesystem with particular encodings. For example, mount(8) for vfat filesystem assumes CP437 if used without option. You need to provide explicit mount option to use UTF-8 or CP932 for filenames.

Note
When auto-mounting a hot-pluggable USB flash drive under modern desktop environment such as GNOME, you may provide such mount option by right clicking the icon on the desktop, click "Drive" tab, click to expand "Setting", and entering "utf8" to "Mount options:". The next time this USB flash drive is mounted, mount with UTF-8 is enabled.

Note
If you are upgrading system or moving disk drives from older non-UTF-8 system, file names with non-ASCII characters may be encoded in the historic and deprecated encodings such as ISO-8859-1 or eucJP. Please seek help of text conversion tools to convert them to UTF-8. See Section 11.1.

Samba uses Unicode for newer clients (Windows NT, 200x, XP) but uses CP850 for older clients (DOS and Windows 9x/Me) as default. This default for older clients can be changed using "`dos charset`" in the "`/etc/samba/smb.conf`" file, e.g., to CP932 for Japanese.

8.1.4 Localized messages and translated documentation

Translations exist for many of the text messages and documents that are displayed in the Debian system, such as error messages, standard program output, menus, and manual pages. GNU gettext(1) command tool chain is used as the backend tool for most translation activities.

Under "Tasks" → "Localization" aptitude(8) provides an extensive list of useful binary packages which add localized messages to applications and provide translated documentation.

For example, you can obtain the localized message for manpage by installing the `manpages-`*`LANG`* package. To read the Italian-language manpage for *programname* from "`/usr/share/man/it/`", execute as the following.

```
LANG=it_IT.UTF-8 man programname
```

GNU gettext can accommodate priority list of translation languages with `$LANGUAGE` environment variable. For example:

```
$ export LANGUAGE="pt:pt_BR:es:it:fr"
```

For more, see `info gettext` and read the section "The LANGUAGE variable".

8.1.5 Effects of the locale

The sort order of characters with sort(1) and ls(1) are affected by the locale. Exporting `LANG=en_US.UTF-8` sorts in the dictionary `A->a->B->b...->Z->z` order, while exporting `LANG=C.UTF-8` sorts in ASCII binary `A->B->...->Z->a->b` order.

The date format of ls(1) is affected by the locale (see Section 9.3.4).

The date format of date(1) is affected by the locale. For example:

```
 $ unset LC_ALL
 $ LANG=en_US.UTF-8 date
Thu Dec 24 08:30:00 PM JST 2023
 $ LANG=en_GB.UTF-8 date
Thu 24 Dec 20:30:10 JST 2023
 $ LANG=es_ES.UTF-8 date
jue 24 dic 2023 20:30:20 JST
 $ LC_TIME=en_DK.UTF-8 date
2023-12-24T20:30:30 JST
```

Number punctuation are different for locales. For example, in English locale, one thousand point one is displayed as "`1,000.1`" while in German locale, it is displayed as "`1.000,1`". You may see this difference in spreadsheet program.

Each detail feature of "`$LANG`" environment variable may be overridden by setting "`$LC_*`" variables. These environment variables can be overridden again by setting "`$LC_ALL`" variable. See locale(7) manpage for the details. Unless you have strong reason to create complicated configuration, please stay away from them and use only "`$LANG`" variable set to one of the UTF-8 locales.

8.2 The keyboard input

The keyboard system can be configured at different layers of the system.

- Linux kernel: keyboard(5)
- X server: setxkbmap(1), xkeyboard-config(5), environment variable `XMODIFIERS`
- GUI desktop environment: Input Method framework: `ibus`, `fcitx5`
- Application: environment variables to set its input source: `GTK_IM_MODULE`, `QT_IM_MODULE`, `QT_IM_MODULES`, ...

Input method framework (IM) consists of:

- Input method engine (IME): Actual input method
- Configuration: Handles the configuration for IBus and other services such as IME plugins
- Panel: User interface such as language bar and candidate selection table

Multilingual input to the application is processed roughly as:

```
Keyboard        UI panel    Configuration          Application
|                  ^ |           |                       ^  ^
v                  | v           v                       |  |
Linux kernel -> Input method engine (IME) -+-> Gtk, Qt ---+  |
                   | ^                     |                 |
                   | |                     +-> X11, Wayland -+
                   v |
                IME plugin (ibus-mozc, ...)
```

8.2.1 The keyboard input for Linux console and X Window

The Debian system can be configured to work with many international keyboard arrangements using the `keyboard-config` package.

```
# dpkg-reconfigure keyboard-configuration
```

For the Linux console and the X Window system, this updates configuration parameters in "`/etc/default/keyboard`". Many non-ASCII characters including accented characters used by many European languages can be made available with dead key, AltGr key, and compose key.

Note
If ibus is active, your classic X keyboard configuration by the `setxkbmap` may be overridden by `ibus` even under classic X-based desktop environment. You can disable installed `ibus` using `im-config` to set input method to "None". For more, see Debian Wiki on keyboard.

8.2.2 The keyboard input for Wayland

Unlike the X Window protocol, the Wayland core protocol doesn't even support the input of accented characters. Popular Wayland Compositors, such as Mutter for GNOME or KWin for KDE, implement extension protocols such as the `text-input-unstable-v3` for the text input (see "current Wayland protocols and their support status").

The `text-input-unstable-v3` protocol works well with Input methods for Wayland (see "Wayland input method project post-mortem").

- Most GUI applications are built with GUI libraries such as GTK or Qt which support this `text-input-unstable-v3`.
- Popular Input Method Engines (IME), such as IBus or Fcitx (version 5), can work with this `text-input-unstable-v3`.
- IMEs support text input for many languages with plugins.
- Recent IMEs integrate "X Keyboard Extension (XKB)" functionalities such as `setxkbmap` previously provided by the X Window to support accented character text input for European languages for Wayland.

8.2.3 The input method support with IBus

For GNOME, "`ibus`" is the default IME which is automatically installed via its package dependency.

- Most multilingualized keyboard input features can be configured from "GNOME Settings" or "GNOME Tweaks".
- Some multilingualized keyboard input features may need to be configured from the ibus-setup(1) command.
- The emoji keyboard input is available by typing "`SUPER-.`" (Simultaneously type Windows and period keys) followed by a keyword for each emoji and SPACE-keys.

The list of IBus and its plugin packages are the following.

8.2.4 The input method support with Fcitx

The Fcitx (version 5) input method framework is popular with Chinese users and compatible with "`ibus`".

The list of "`fcitx5`" and its plugin packages are the following.

8.2.5 An example for Japanese

I find the Japanese input method started under English environment ("`en_US.UTF-8`") very useful. Here is how I did this with IBus:

1. Install the Japanese input tool package `ibus-mozc` (or `ibus-anthy`).

package	popcon	size	supported locale
ibus	V:212, I:253	1884	input method framework using dbus
ibus-mozc	V:2.1, I:3.8	1054	Japanese
ibus-anthy	V:0.5, I:1.1	8740	Japanese
ibus-skk	V:0.03, I:0.14	242	Japanese
ibus-kkc	V:0.04, I:0.19	206	Japanese
ibus-libpinyin	V:1.1, I:5.1	123	Chinese (for zh_CN)
ibus-chewing	V:0.16, I:0.84	298	Chinese (for zh_TW)
ibus-libzhuyin	I:0.10	41021	Chinese (for zh_TW)
ibus-rime	V:0.29, I:0.51	76	Chinese (for zh_CN/zh_TW)
ibus-cangjie	V:0.02, I:0.10	235	Chinese (for zh_HK)
ibus-hangul	V:0.3, I:1.9	261	Korean
ibus-libthai	V:0.00, I:0.05	84	Thai
ibus-table-thai	I:0.04	59	Thai
ibus-unikey	V:0.22, I:0.40	286	Vietnamese
keyman	V:0.01, I:0.10	507	Multilingual: Keyman plugin for over 2000 languages
ibus-table	V:0.04, I:0.92	2364	table plugin for IBus
ibus-m17n	V:0.1, I:1.8	373	Multilingual: Indic, Arabic and others

Table 8.1: List of IBus and its plugin packages

package	popcon	size	supported locale
fcitx5	V:8, I:13	712	input method framework compatible with "ibus"
fcitx5-mozc	V:1.1, I:1.8	1328	Japanese
fcitx5-anthy	V:0.07, I:0.25	853	Japanese
fcitx5-skk	V:0.05, I:0.15	370	Japanese
fcitx5-kkc	V:0.01, I:0.06	418	Japanese
fcitx5-chinese-addons	I:9.2	17	Chinese (metapackage for zh_*)
fcitx5-pinyin	V:4.0, I:9.7	1000	Chinese (for zh_CN)
fcitx5-chewing	V:0.29, I:0.98	228	Chinese (for zh_TW)
fcitx5-zhuyin	I:0.08	108535	Chinese (for zh_TW)
fcitx5-rime	V:0.46, I:0.93	393	Chinese (for zh_CN/zh_TW)
fcitx5-table-cangjie-large	I:0.13	1292	Chinese (for zh_HK)
fcitx5-hangul	V:0.12, I:0.27	239	Korean
fcitx5-libthai	V:0.00, I:0.04	119	Thai
fcitx5-table-thai	I:0.09	34	Thai
fcitx5-unikey	V:0.13, I:0.23	624	Vietnamese
fcitx5-m17n	V:0.16, I:0.61	171	Multilingual: Indic, Arabic and others
fcitx5-table	V:0.3, I:9.6	428	table plugin for fcitx5
fcitx5-keyman	V:0.04, I:0.05	241	Multilingual: Keyman plugin for over 2000 languages

Table 8.2: List of Fcitx5 and its plugin packages

2. • Generic: Execute ibus-setup(1) → select ”Input Method” → click ”Add” → ”Japanese” → ”Mozc (or Anthy)” → click ”Add”
   • GNOME: Select ”Settings” → ”Keyboard” → ”Input Sources” → click ”+” in ”Input Sources” → ”Japanese” → ”Mozc (or Anthy)” → click ”Add”
3. You may choose as many input sources.
4. Relogin to user’s account.
5. Setup each input source by right clicking the GUI toolbar icon.
6. Switch among installed input sources by SUPER-SPACE. (SUPER is normally the Windows key.)

Tip
If you wish to have access to alphabet only keyboard environment with the physical Japanese keyboard on which shift-2 has " (double quotation mark) engraved, you select ”Japanese” in the above procedure. You can enter Japanese using ”Japanese mozc (or anthy)” with physical ”US” keyboard on which shift-2 has @ (at mark) engraved.

- For Wayland:
  - The `im-config` package does nothing and can be removed safely.
  - You probably don’t need to set environment variables except for the backward compatibility etc.
  - If you need to set environment variables, create a file such as ”`~/.config/environment.d/50-input-method.con`” to set them.
- For X Window:
  - Install the `im-config` package.
  - The GUI menu entry for im-config(8) is ”Input method”.
  - Alternatively, execute ”`im-config`” from user’s shell.
  - im-config(8) behaves differently if command is executed from root or not.
  - im-config(8) enables the best input method on the system as default without any user actions.

8.3 The display output

Linux console can only display limited characters. (You need to use special terminal program such as jfbterm(1) to display non-European languages on the non-GUI console.)

GUI environment (Chapter 7) can display any characters in the UTF-8 as long as required fonts are installed and enabled. (The encoding of the original font data is taken care and transparent to the user.)

8.3.1 The terminal configuration

The Debian system can be configured to work with many international console arrangements using the `console-setup` package.

```
# dpkg-reconfigure console-setup
```

For the Linux console and the X Window system, this updates configuration parameters in ”`/etc/default/console-setu`” to display many non-ASCII characters including accented characters used by many European languages can be made available.

There are several components to configure character console and ncurses(3) system features.

- The "`/etc/terminfo/*/*`" file (terminfo(5))
- The "`$TERM`" environment variable (term(7))
- setterm(1), stty(1), tic(1), and toe(1)

If the `terminfo` entry for `xterm` doesn't work with a non-Debian `xterm`, change your terminal type, "`$TERM`", from "`xterm`" to one of the feature-limited versions such as "`xterm-r6`" when you log in to a Debian system remotely. See "`/usr/share/doc/libncurses5/FAQ`" for more. "`dumb`" is the lowest common denominator for "`$TERM`".

8.3.2 East Asian Ambiguous Character Width Characters

Under the East Asian locale, the box drawing, Greek, and Cyrillic characters may be displayed wider than your desired width to cause the unaligned terminal output (see Unicode Standard Annex #11, 4.2 Ambiguous Characters).

You can work around this problem:

- `gnome-terminal`: Preferences → Profiles → *Profile name* → Compatibility → Ambiguous-wide characters → Narrow
- `ncurses`: Set environment `export NCURSES_NO_UTF8_ACS=0`.

Chapter 9

System tips

Here, I describe basic tips to configure and manage systems, mostly from the console.

9.1 The console tips

There are some utility programs to help your console activities.

package	popcon	size	description
mc	V:40, I:176	1590	See Section 1.3
bsdutils	V:421, I:999	338	script(1) command to make a record of terminal session
screen	V:50, I:191	1007	terminal multiplexer with VT100/ANSI terminal emulation
tmux	V:106, I:167	1423	terminal multiplexer alternative (Use ”Control-B” instead)
ripgrep	V:17, I:48	5284	fast recursive string search in the source code tree with automatic filtering
fzf	V:10, I:36	5677	fuzzy text finder
fzy	V:0.07, I:0.34	59	fuzzy text finder
rlwrap	V:1, I:11	328	readline feature command line wrapper
ledit	V:0.5, I:7.3	375	readline feature command line wrapper
rlfe	V:0.02, I:0.48	45	readline feature command line wrapper

Table 9.1: List of programs to support console activities

9.1.1 Recording the shell activities cleanly

The simple use of script(1) (see Section 1.4.9) to record shell activity produces a file with control characters. This can be avoided by using col(1) as the following.

```
$ script
Script started, file is typescript
```

Do whatever … and press `Ctrl-D` to exit `script`.

```
$ col -bx < typescript > cleanedfile
$ vim cleanedfile
```

There are alternative methods to record the shell activities:

- Use `tee` (usable during the boot process in the initramfs):

```
$ sh -i 2>&1 | tee typescript
```

- Use `gnome-terminal` with the extend line buffer for scrollback.
- Use `screen` with "`^A H`" (see Section 9.1.2) to perform recording of console.
- Use `vim` with "`:terminal`" to enter the terminal mode. Use "`Ctrl-W N`" to exit from terminal mode to normal mode. Use "`:w typescript`" to write the buffer to a file.
- Use `emacs` with "`M-x shell`", "`M-x eshell`", or "`M-x term`" to enter recording console. Use "`C-x C-w`" to write the buffer to a file.

9.1.2 The screen program

screen(1) not only allows one terminal window to work with multiple processes, but also allows **remote shell process to survive interrupted connections**. Here is a typical use scenario of screen(1).

1. You login to a remote machine.
2. You start `screen` on a single console.
3. You execute multiple programs in `screen` windows created with `^A c` ("Control-A" followed by "c").
4. You switch among the multiple `screen` windows by `^A n` ("Control-A" followed by "n").
5. Suddenly you need to leave your terminal, but you don't want to lose your active work by keeping the connection.
6. You may **detach** the `screen` session by any methods.
   - Brutally unplug your network connection
   - Type `^A d` ("Control-A" followed by "d") and manually logging out from the remote connection
   - Type `^A DD` ("Control-A" followed by "DD") to have `screen` detach and log you out
7. You log in again to the same remote machine (even from a different terminal).
8. You start `screen` as "`screen -r`".
9. `screen` magically **reattaches** all previous `screen` windows with all actively running programs.

Tip
You can save connection fees with `screen` for metered network connections such as dial-up and packet ones, because you can leave a process active while disconnected, and then re-attach it later when you connect again.

In a `screen` session, all keyboard inputs are sent to your current window except for the command keystroke. All `screen` command keystrokes are entered by typing `^A` ("Control-A") plus a single key [plus any parameters]. Here are important ones to remember.

See screen(1) for details.

See tmux(1) for functionalities of the alternative command.

key binding	meaning
^A ?	show a help screen (display key bindings)
^A c	create a new window and switch to it
^A n	go to next window
^A p	go to previous window
^A 0	go to window number 0
^A 1	go to window number 1
^A w	show a list of windows
^A a	send a Ctrl-A to current window as keyboard input
^A h	write a hardcopy of current window to file
^A H	begin/end logging current window to file
^A ^X	lock the terminal (password protected)
^A d	detach screen session from the terminal
^A DD	detach screen session and log out

Table 9.2: List of key bindings for screen

9.1.3 Navigating around directories

In Section 1.4.2, 2 tips to allow quick navigation around directories are described: `$CDPATH` and `mc`.

If you use fuzzy text filter program, you can do without typing the exact path. For `fzf`, include following in `~/.bashrc`.

```
FZF_KEYBINDINGS_PATH=/usr/share/doc/fzf/examples/key-bindings.bash
if [ -f $FZF_KEYBINDINGS_PATH ]; then
  . $FZF_KEYBINDINGS_PATH
fi
```

For example:

- You can jump to a very deep subdirectory with minimal efforts. You first type "`cd **`" and press `Tab`. Then you will be prompted with candidate paths. Typing in partial path strings, e.g., `s/d/b foo`, will narrow down candidate paths. You select the path to be used by `cd` with cursor and return keys.
- You can select a command from the command history more efficiently with minimal efforts. You press `Ctrl-R` at the command prompt. Then you will be prompted with candidate commands. Typing in partial command strings, e.g., `vim d`, will narrow down candidates. You select the one to be used with cursor and return keys.

9.1.4 Readline wrapper

Some commands such as `/usr/bin/dash` which lacks command line history editing capability can add such functionality transparently by running under `rlwrap` or its equivalents.

```
$ rlwrap dash -i
```

This provides convenient platform to test subtle points for `dash` with friendly `bash`-like environment.

9.1.5 Scanning the source code tree

The rg(1) command in the `ripgrep` package offers a faster alternative to the grep(1) command for scanning the source code tree for typical situation. It takes advantage of modern multi-core CPUs and automatically applies reasonable filters to skip some files.

9.2 Customizing vim

After you learn basics of vim(1) through Section 1.4.8, please read Bram Moolenaar's "Seven habits of effective text editing (2000)" to understand how `vim` should be used.

9.2.1 Customizing vim with internal features

The behavior of `vim` can be changed significantly by enabling its internal features through the Ex-mode commands such as "`set ...`" to set vim options.

These Ex-mode commands can be included in user's vimrc file, traditional "`~/.vimrc`" or git-friendly "`~/.vim/vimrc`". Here is a very simple example 1:

```
""" Generic baseline Vim and Neovim configuration (~/.vimrc)
"""   - For NeoVim, use "nvim -u ~/.vimrc [filename]"
""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""
let mapleader = ' '             " :h mapleader
""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""
set nocompatible                " :h 'cp -- sensible (n)vim mode
syntax on                       " :h :syn-on
filetype plugin indent on       " :h :filetype-overview
set encoding=utf-8              " :h 'enc (default: latin1) -- sensible encoding
""" current vim option value can be verified by :set encoding?
set backspace=indent,eol,start  " :h 'bs (default: nobs) -- sensible BS
set statusline=%<%f%m%r%h%w%=%y[U+%04B]%2l/%2L=%P,%2c%V
set listchars=eol:¶,tab:b''▫b''\ ,extends:b''▫b'',precedes:b''▫b'',nbsp:b''▫b''
set viminfo=!,'100,<5000,s100,h " :h 'vi -- bigger copy buffer etc.
""" Pick "colorscheme" from blue darkblue default delek desert elflord evening
""" habamax industry koehler lunaperche morning murphy pablo peachpuff quiet ron
""" shine slate torte zellner
colorscheme industry
""" don't pick "colorscheme" as "default" which may kill SpellUnderline settings
set scrolloff=5                 " :h 'scr -- show 5 lines around cursor
set laststatus=2                " :h 'ls (default 1)  k
""" boolean options can be unset by prefixing "no"
set ignorecase                  " :h 'ic
set smartcase                   " :h 'scs
set autoindent                  " :h 'ai
set smartindent                 " :h 'si
set nowrap                      " :h 'wrap
"set list                        " :h 'list (default nolist)
set noerrorbells                " :h 'eb
set novisualbell                " :h 'vb
set t_vb=                       " :h 't_vb -- termcap visual bell
set spell                       " :h 'spell
set spelllang=en_us,cjk         " :h 'spl -- english spell, ignore CJK
set clipboard=unnamedplus       " :h 'cb -- cut/copy/paste with other app
set hidden                      " :h 'hid
set autowrite                   " :h 'aw
set timeoutlen=300              " :h 'tm
```

The keymap of `vim` can be changed in user's vimrc file. E.g.:



Caution
Don't try to change the default key bindings without very good reasons.

1More elaborate customization examples: "Vim Galore", "sensible.vim", ...

```
""" Popular mappings (imitating LazyVim etc.)
""" Window moves without using CTRL-W which is dangerous in INSERT mode
nnoremap <C-H> <C-W>h
nnoremap <C-J> <C-W>j
nnoremap <C-K> <C-W>k
silent! nnoremap <C-L> <C-W>l
""" Window resize
nnoremap <C-LEFT> <CMD>vertical resize -2<CR>
nnoremap <C-DOWN> <CMD>resize -2<CR>
nnoremap <C-UP> <CMD>resize +2<CR>
nnoremap <C-RIGHT> <CMD>vertical resize +2<CR>
""" Clear hlsearch with <ESC> (<C-L> is mapped as above)
nnoremap <ESC> <CMD>noh<CR><ESC>
inoremap <ESC> <CMD>noh<CR><ESC>
""" center after jump next
nnoremap n nzz
nnoremap N Nzz
""" fast "jk" to get out of INSERT mode (<ESC>)
inoremap  jk <CMD>noh<CR><ESC>
""" fast "<ESC><ESC>" to get out of TERM mode (CTRL-\ CTRL-N)
tnoremap <ESC><ESC> <C-\><C-N>
""" fast "jk" to get out of TERM mode (CTRL-\ CTRL-N)
tnoremap jk <C-\><C-N>
""" previous/next trouble/quickfix item
nnoremap [q <CMD>cprevious<CR>
nnoremap ]q <CMD>cnext<CR>
""" buffers
nnoremap <S-H> <CMD>bprevious<CR>
nnoremap <S-L> <CMD>bnext<CR>
nnoremap [b <CMD>bprevious<CR>
nnoremap ]b <CMD>bnext<CR>
""" Add undo break-points
inoremap  , ,<C-G>u
inoremap  . .<C-G>u
inoremap  ; ;<C-G>u
""" save file
inoremap <C-S> <CMD>w<CR><ESC>
xnoremap <C-S> <CMD>w<CR><ESC>
nnoremap <C-S> <CMD>w<CR><ESC>
snoremap <C-S> <CMD>w<CR><ESC>
""" better indenting
vnoremap < <gv
vnoremap > >gv
""" terminal (Somehow under Linux, <C-/> becomes <C-_> in Vim)
nnoremap <C-_> <CMD>terminal<CR>
"nnoremap <C-/> <CMD>terminal<CR>
"""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""
if ! has('nvim')
""" Toggle paste mode with <SPACE>p for Vim (no need for Nvim)
set pastetoggle=<leader>p
""" nvim default mappings for Vim.  See :h default-mappings in nvim
""" copy to EOL (no delete) like D for d
noremap Y y$
""" sets a new undo point before deleting
inoremap <C-U> <C-G>u<C-U>
inoremap <C-W> <C-G>u<C-W>
""" <C-L> is re-purposed as above
""" execute the previous macro recorded with Q
nnoremap Q @@
""" repeat last substitute and *KEEP* flags
nnoremap & :&&<CR>
```

```
""" search visual selected string for visual mode
xnoremap * y/\V<C-R>"<CR>
xnoremap # y?\V<C-R>"<CR>
endif
```

In order for the above keybindings to function properly, the terminal program needs to be configured to generate "ASCII DEL" for `Backspace`-key and "Escape sequence" for `Delete`-key.

Other miscellaneous configuration can be changed in user's vimrc file. E.g.:

```
""" Use faster 'rg' (ripgrep package) for :grep
if executable("rg")
  set grepprg=rg\ --vimgrep\ --smart-case
  set grepformat=%f:%l:%c:%m
endif
"""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""
""" Retain last cursor position :h '"
augroup RetainLastCursorPosition
  autocmd!
  autocmd BufReadPost *
    \ if line("'\"") > 0 && line ("'\"") <= line("$") |
    \   exe "normal! g'\"" |
    \ endif
augroup END
"""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""
""" Force to use underline for spell check results
augroup SpellUnderline
  autocmd!
  autocmd ColorScheme * highlight SpellBad term=Underline gui=Undercurl
  autocmd ColorScheme * highlight SpellCap term=Underline gui=Undercurl
  autocmd ColorScheme * highlight SpellLocal term=Underline gui=Undercurl
  autocmd ColorScheme * highlight SpellRare term=Underline gui=Undercurl
augroup END
"""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""
""" highlight tailing spaces except when typing as red (set after colorscheme)
highlight TailingWhitespaces ctermbg=red guibg=red
""" \s\+     1 or more whitespace character: <Space> and <Tab>
""" \%#\@<!  Matches with zero width if the cursor position does NOT match.
match TailingWhitespaces /\s\+\%#\@<!$/
```

9.2.2 Customizing vim with external packages

Interesting external plugin packages can be found:

- Vim - the ubiquitous text editor -- The official upstream site of Vim and vim scripts
- VimAwsome -- The listing of Vim plugins
- vim-scripts -- Debian package: a collection of vim scripts

Plugin packages in the vim-scripts package can be enabled using user's vimrc file. E.g.:

```
packadd! secure-modelines
packadd! winmanager
" IDE-like UI for files and buffers with <space>w
nnoremap <leader>w         :WMToggle<CR>
```

The new native Vim package system works nicely with "`git`" and "`git submodule`". One such example configuration can be found at my git repository: dot-vim. This does essentially:

- By using "`git`" and "`git submodule`", latest external packages, such as "*name*", are placed into `~/.vim/pack/*/opt/r` and similar.
- By adding `:packadd! name` line to user's vimrc file, these packages are placed on `runtimepath`.
- Vim loads these packages on `runtimepath` during its initialization.
- At the end of its initialization, tags for the installed documents are updated with "`helptags ALL`".

For more, please start `vim` with "`vim --startuptime vimstart.log`" to check actual execution sequence and time spent for each step.

It is quite confusing to see too many ways[2] to manage and load these external packages to `vim`. Checking the original information is the best cure.

key strokes	information
`:help package`	explanation on the vim package mechanism
`:help runtimepath`	explanation on the `runtimepath` mechanism
`:version`	internal states including candidates for the vimrc file
`:echo $VIM`	the environment variable "`$VIM`" used to locate the vimrc file
`:set runtimepath?`	list of directories which will be searched for all runtime support files
`:echo $VIMRUNTIME`	the environment variable "`$VIMRUNTIME`" used to locate various system provided runtime support files

Table 9.3: Information on the initialization of `vim`

9.3 Data recording and presentation

9.3.1 The log daemon

Many traditional programs record their activities in the text file format under the "`/var/log/`" directory.

logrotate(8) is used to simplify the administration of log files on a system which generates a lot of log files.

Many new programs record their activities in the binary file format using systemd-journald(8) Journal service under the "`/var/log/journal`" directory.

You can log data to the systemd-journald(8) Journal from a shell script by using the systemd-cat(1) command.

See Section 3.5 and Section 3.4.

9.3.2 Log analyzer

Here are notable log analyzers ("`~Gsecurity::log-analyzer`" in aptitude(8)).

Note
CRM114 provides language infrastructure to write **fuzzy** filters with the TRE regex library. Its popular use is spam mail filter but it can be used as log analyzer.

2vim-pathogen was popular.

package	popcon	size	description
fail2ban	V:101, I:111	2275	ban IPs that cause multiple authentication errors
logwatch	V:9, I:10	2440	log analyzer with nice output written in Perl
awstats	V:5.1, I:8.0	6934	powerful and featureful web server log analyzer
analog	V:3, I:83	3739	web server log analyzer
sarg	V:0.65, I:0.71	863	squid analysis report generator
pflogsumm	V:1.5, I:3.5	173	Postfix log entry summarizer
fwlogwatch	V:0.10, I:0.16	487	firewall log analyzer
squidview	V:0.04, I:0.41	224	monitor and analyze squid access.log files
swatch	V:0.07, I:0.30	99	log file viewer with regexp matching, highlighting, and hooks
crm114	V:0.06, I:0.32	1371	Controllable Regex Mutilator and Spam Filter (CRM114)
icmpinfo	V:0.01, I:0.35	42	interpret ICMP messages

Table 9.4: List of system log analyzers

9.3.3 Customized display of text data

Although pager tools such as more(1) and less(1) (see Section 1.4.5) and custom tools for highlighting and formatting (see Section 11.1.8) can display text data nicely, general purpose editors (see Section 1.4.6) are most versatile and customizable.

Tip
For vim(1) and its pager mode alias view(1), "`:set hls`" enables highlighted search.

9.3.4 Customized display of time and date

The default display format of time and date by the "`ls -l`" command depends on the **locale** (see Section 1.2.6 for value). The "`$LANG`" variable is referred first and it can be overridden by the "`$LC_TIME`" or "`$LC_ALL`" exported environment variables.

The actual default display format for each locale depends on the version of the standard C library (the `libc6` package) used. I.e., different releases of Debian had different defaults. For iso-formats, see ISO 8601.

If you really wish to customize this display format of time and date beyond the **locale**, you should set the **time style value** by the "`--time-style`" argument or by the "`$TIME_STYLE`" value (see ls(1), date(1), "`info coreutils 'ls invocation'`").

Tip
You can eliminate typing long option on commandline using command alias (see Section 1.5.9):

```
alias ls='ls --time-style=+%d.%m.%y %H:%M'
```

9.3.5 Colorized shell echo

Shell echo to most modern terminals can be colorized using ANSI escape code (see "`/usr/share/doc/xterm/ctlseqs.`

For example, try the following

time style value	locale	display of time and date
iso	any	01-19 00:15
long-iso	any	2009-01-19 00:15
full-iso	any	2009-01-19 00:15:16.000000000 +0900
locale	C	Jan 19 00:15
locale	en_US.UTF-8	Jan 19 00:15
locale	es_ES.UTF-8	ene 19 00:15
+%d.%m.%y %H:%M	any	19.01.09 00:15
+%d.%b.%y %H:%M	C or en_US.UTF-8	19.Jan.09 00:15
+%d.%b.%y %H:%M	es_ES.UTF-8	19.ene.09 00:15

Table 9.5: Display examples of time and date for the "`ls -l`" command with the **time style value**

```
$ RED=$(printf "\x1b[31m")
$ NORMAL=$(printf "\x1b[0m")
$ REVERSE=$(printf "\x1b[7m")
$ echo "${RED}RED-TEXT${NORMAL} ${REVERSE}REVERSE-TEXT${NORMAL}"
```

9.3.6 Colorized commands

Colorized commands are handy for inspecting their output in the interactive environment. I include the following in my "`~/.bashrc`".

```
if [ "$TERM" != "dumb" ]; then
    eval "`dircolors -b`"
    alias ls='ls --color=always'
    alias ll='ls --color=always -l'
    alias la='ls --color=always -A'
    alias less='less -R'
    alias ls='ls --color=always'
    alias grep='grep --color=always'
    alias egrep='egrep --color=always'
    alias fgrep='fgrep --color=always'
    alias zgrep='zgrep --color=always'
else
    alias ll='ls -l'
    alias la='ls -A'
fi
```

The use of alias limits color effects to the interactive command usage. It has advantage over exporting environment variable "`export GREP_OPTIONS='--color=auto'`" since color can be seen under pager programs such as less(1). If you wish to suppress color when piping to other programs, use "`--color=auto`" instead in the above example for "`~/.bashrc`".

Tip
You can turn off these colorizing aliases in the interactive environment by invoking shell with "`TERM=dumb bash`".

9.3.7 Recording the editor activities for complex repeats

You can record the editor activities for complex repeats.

For Vim, as follows.

- ”qa”: start recording typed characters into named register ”a”.
- … editor activities
- ”q”: end recording typed characters.
- ”@a”: execute the contents of register ”a”.

For Emacs, as follows.

- ”`C-x (`”: start defining a keyboard macro.
- … editor activities
- ”`C-x )`”: end defining a keyboard macro.
- ”`C-x e`”: execute a keyboard macro.

9.3.8 Recording the graphics image of an X application

There are few ways to record the graphics image of an X application, including an `xterm` display.

package	popcon	size	screen
`gnome-screenshot`	V:12, I:94	1115	Wayland
`flameshot`	V:7, I:17	3542	Wayland
`gimp`	V:34, I:211	33315	Wayland + X
`x11-apps`	V:31, I:443	2461	X
`imagemagick`	V:7, I:272	80	X
`scrot`	V:4, I:48	144	X

Table 9.6: List of graphics image manipulation tools

9.3.9 Recording changes in configuration files

There are specialized tools to record changes in configuration files with help of DVCS and to make system snapshots on Btrfs.

package	popcon	size	description
`etckeeper`	V:24, I:27	157	store configuration files and their metadata with Git (default), Mercurial, or GNU Bazaar
`timeshift`	V:7, I:13	4481	system restore utility using rsync or BTRFS snapshots
`snapper`	V:6.1, I:8.3	2396	Linux filesystem snapshot management tool

Table 9.7: List of packages which can record configuration history

You may also think about local script Section 10.2.3 approach.

package	popcon	size	description
coreutils	V:912, I:1000	17994	nice(1): run a program with modified scheduling priority
bsdutils	V:421, I:999	338	renice(1): modify the scheduling priority of a running process
procps	V:831, I:997	2691	"/proc" filesystem utilities: ps(1), top(1), kill(1), watch(1), …
psmisc	V:400, I:715	950	"/proc" filesystem utilities: killall(1), fuser(1), peekfd(1), pstree(1)
time	V:6, I:76	148	time(1): run a program to report system resource usages with respect to time
sysstat	V:115, I:164	1904	sar(1), iostat(1), mpstat(1), …: system performance tools for Linux
isag	V:0.1, I:3.1	109	Interactive System Activity Grapher for sysstat
lsof	V:456, I:950	492	lsof(8): list files opened by a running process using "-p" option
strace	V:10, I:100	3209	strace(1): trace system calls and signals
ltrace	V:1, I:11	420	ltrace(1): trace library calls
xtrace	V:0.03, I:0.69	342	xtrace(1): trace communication between X11 client and server
powertop	V:36, I:230	696	powertop(1): information about system power use
cron	V:923, I:997	248	run processes according to a schedule in background from cron(8) daemon
anacron	V:423, I:481	110	cron-like command scheduler for systems that don't run 24 hours a day
at	V:69, I:91	158	at(1) or batch(1): run a job at a specified time or below certain load level

Table 9.8: List of tools for monitoring and controlling program activities

9.4 Monitoring, controlling, and starting program activities

Program activities can be monitored and controlled using specialized tools.

Tip
The procps packages provide very basics of monitoring, controlling, and starting program activities. You should learn all of them.

9.4.1 Timing a process

Display time used by the process invoked by the command.

```
# time some_command >/dev/null
real    0m0.035s       # time on wall clock (elapsed real time)
user    0m0.000s       # time in user mode
sys     0m0.020s       # time in kernel mode
```

9.4.2 The scheduling priority

A nice value is used to control the scheduling priority for the process.

```
# nice  -19 top                                      # very nice
# nice --20 wodim -v -eject speed=2 dev=0,0 disk.img # very fast
```

Sometimes an extreme nice value does more harm than good to the system. Use this command carefully.

nice value	scheduling priority
19	lowest priority process (nice)
0	very high priority process for user
-20	very high priority process for root (not-nice)

Table 9.9: List of nice values for the scheduling priority

9.4.3 The ps command

The ps(1) command on a Debian system support both BSD and SystemV features and helps to identify the process activity statically.

style	typical command	feature
BSD	`ps aux`	display %CPU %MEM
System V	`ps -efH`	display PPID

Table 9.10: List of ps command styles

For the zombie (defunct) children process, you can kill them by the parent process ID identified in the "`PPID`" field.

The pstree(1) command display a tree of processes.

9.4.4 The top command

top(1) on the Debian system has rich features and helps to identify what process is acting funny dynamically.

It is an interactive full screen program. You can get its usage help press by pressing the "h"-key and terminate it by pressing the "q"-key.

9.4.5 Listing files opened by a process

You can list all files opened by a process with a process ID (PID), e.g. 1, by the following.

```
$ sudo lsof -p 1
```

PID=1 is usually the `init` program.

9.4.6 Tracing program activities

You can trace program activity with strace(1), ltrace(1), or xtrace(1) for system calls and signals, library calls, or communication between X11 client and server.

You can trace system calls of the `ls` command as the following.

```
$ sudo strace ls
```

Tip
Use **strace-graph** script found in **/usr/share/doc/strace/examples/** to make a nice tree view

9.4.7 Identification of processes using files or sockets

You can also identify processes using files by fuser(1), e.g. for "`/var/log/mail.log`" by the following.

```
$ sudo fuser -v /var/log/mail.log
                     USER        PID ACCESS COMMAND
/var/log/mail.log:   root       2946 F.... rsyslogd
```

You see that file "`/var/log/mail.log`" is open for writing by the rsyslogd(8) command.

You can also identify processes using sockets by fuser(1), e.g. for "`smtp/tcp`" by the following.

```
$ sudo fuser -v smtp/tcp
                     USER        PID ACCESS COMMAND
smtp/tcp:            Debian-exim   3379 F.... exim4
```

Now you know your system runs exim4(8) to handle TCP connections to SMTP port (25).

9.4.8 Repeating a command with a constant interval

watch(1) executes a program repeatedly with a constant interval while showing its output in fullscreen.

```
$ watch w
```

This displays who is logged on to the system updated every 2 seconds.

9.4.9 Repeating a command looping over files

There are several ways to repeat a command looping over files matching some condition, e.g. matching glob pattern "`*.ext`".

- Shell for-loop method (see Section 12.1.4):

```
for x in *.ext; do if [ -f "$x"]; then command "$x" ; fi; done
```

- find(1) and xargs(1) combination:

```
find . -type f -maxdepth 1 -name '*.ext' -print0 | xargs -0 -n 1 command
```

- find(1) with "`-exec`" option with a command:

```
find . -type f -maxdepth 1 -name '*.ext' -exec command '{}' \;
```

- find(1) with "`-exec`" option with a short shell script:

```
find . -type f -maxdepth 1 -name '*.ext' -exec sh -c "command '{}' && echo 'successful'" \;
```

The above examples are written to ensure proper handling of funny file names such as ones containing spaces. See Section 10.1.5 for more advance uses of find(1).

9.4.10 Starting a program from GUI

For the command-line interface (CLI), the first program with the matching name found in the directories specified in the `$PATH` environment variable is executed. See Section 1.5.3.

For the graphical user interface (GUI) compliant to the freedesktop.org standards, the `*.desktop` files in the `/usr/share/a` directory provide necessary attributes for the GUI menu display of each program. Each package which is compliant to Freedesktop.org's xdg menu system installs its menu data provided by "*.desktop" under "/usr/share/applications/". Modern desktop environments which are compliant to Freedesktop.org standard use these data to generate their menu using the xdg-utils package. See "/usr/share/doc/xdg-utils/README".

For example, the `chromium.desktop` file defines attributes for the "Chromium Web Browser" such as "Name" for the program name, "Exec" for the program execution path and arguments, "Icon" for the icon used, etc. (see the Desktop Entry Specification) as follows:

```
[Desktop Entry]
Version=1.0
Name=Chromium Web Browser
GenericName=Web Browser
Comment=Access the Internet
Comment[fr]=Explorer le Web
Exec=/usr/bin/chromium %U
Terminal=false
X-MultipleArgs=false
Type=Application
Icon=chromium
Categories=Network;WebBrowser;
MimeType=text/html;text/xml;application/xhtml_xml;x-scheme-handler/http;x-scheme-handler/ ↩
    https;
StartupWMClass=Chromium
StartupNotify=true
```

This is an oversimplified description. The `*.desktop` files are scanned as follows.

The desktop environment sets `$XDG_DATA_HOME` and `$XDG_DATA_DIR` environment variables. For example, under the GNOME:

- `$XDG_DATA_HOME` is unset. (The default value of `$HOME/.local/share` is used.)
- `$XDG_DATA_DIRS` is set to `/usr/share/gnome:/usr/local/share/:/usr/share/`.

So the base directories (see XDG Base Directory Specification) and the `applications` directories are as follows.

- `$HOME/.local/share/` → `$HOME/.local/share/applications/`
- `/usr/share/gnome/` → `/usr/share/gnome/applications/`
- `/usr/local/share/` → `/usr/local/share/applications/`
- `/usr/share/` → `/usr/share/applications/`

The `*.desktop` files are scanned in these `applications` directories in this order.

Tip
A user custom GUI menu entry can be created by adding a `*.desktop` file in the `$HOME/.local/share/applications/` directory.

Tip
The "`Exec=...`" line isn't parsed by the shell. Use the env(1) command if environment variables need to be set.

Tip
Similarly, if a `*.desktop` file is created in the `autostart` directory under these base directories, the specified program in the `*.desktop` file is executed automatically when the desktop environment is started. See Desktop Application Autostart Specification.

Tip
Similarly, if a `*.desktop` file is created in the `$HOME/Desktop` directory and the Desktop environment is configured to support the desktop icon launcher feature, the specified program in it is executed upon clicking the icon. Please note that the actual name of the `$HOME/Desktop` directory is locale dependent. See xdg-user-dirs-update(1).

9.4.11 Customizing program to be started

Some programs start another program automatically. Here are check points for customizing this process.

- Application configuration menu:
  - GNOME desktop: "Settings" → "Apps" → "Default Apps"
  - KDE desktop: "Application Launcher" → "System Settings" → "Default Applications"
  - Iceweasel browser: "Edit" → "Preferences" → "Applications"
  - mc(1): "`/etc/mc/mc.ext`"
- Environment variables such as "`$BROWSER`", "`$EDITOR`", "`$VISUAL`", and "`$PAGER`" (see environ(7))
- The update-alternatives(1) system for programs such as "`editor`", "`view`", "`x-www-browser`", "`gnome-www-browser`", and "`www-browser`" (see Section 1.4.7)
- the "`~/.mailcap`" and "`/etc/mailcap`" file contents which associate MIME type with program (see mailcap(5))
- The "`~/.mime.types`" and "`/etc/mime.types`" file contents which associate file name extension with MIME type (see run-mailcap(1))

Tip
update-mime(8) updates the "`/etc/mailcap`" file using "`/etc/mailcap.order`" file (see mailcap.order(5)).

Tip
The `debianutils` package provides sensible-browser(1), sensible-editor(1), and sensible-pager(1) which make sensible decisions on which editor, pager, and web browser to call, respectively. I recommend you to read these shell scripts.

Tip
In order to run a console application such as `mutt` under GUI as your preferred application, you should create an GUI application as following and set "`/usr/local/bin/mutt-term`" as your preferred application to be started as described.

```
# cat /usr/local/bin/mutt-term <<EOF
#!/bin/sh
gnome-terminal -e "mutt \$@"
EOF
# chmod 755 /usr/local/bin/mutt-term
```

9.4.12 Killing a process

Use kill(1) to kill (or send a signal to) a process by the process ID.

Use killall(1) or pkill(1) to do the same by the process command name and other attributes.

signal value	signal name	action	note
0	---	no signal is sent (see kill(2))	check if process is running
1	SIGHUP	terminate the process	disconnected terminal (signal hang up)
2	SIGINT	terminate the process	interrupt from keyboard (CTRL-C)
3	SIGQUIT	terminate the process and dump core	quit from keyboard (CTRL-\)
9	SIGKILL	terminate the process	unblockable kill signal
15	SIGTERM	terminate the process	blockable termination signal

Table 9.11: List of frequently used signals for kill command

9.4.13 Scheduling tasks once

Run the at(1) command to schedule a one-time job by the following.

```
$ echo 'command -args'| at 3:40 monday
```

9.4.14 Scheduling tasks regularly

Use cron(8) to schedule tasks regularly. See crontab(1) and crontab(5).

You can schedule to run processes as a normal user, e.g. `foo` by creating a crontab(5) file as "`/var/spool/cron/crontab` with "`crontab -e`" command.

Here is an example of a crontab(5) file.

```
# use /usr/bin/sh to run commands, no matter what /etc/passwd says
SHELL=/bin/sh
# mail any output to paul, no matter whose crontab this is
MAILTO=paul
# Min Hour DayOfMonth Month DayOfWeek command (Day... are OR'ed)
# run at 00:05, every day
5  0  *  * *   $HOME/bin/daily.job >> $HOME/tmp/out 2>&1
# run at 14:15 on the first of every month -- output mailed to paul
15 14 1  * *   $HOME/bin/monthly
# run at 22:00 on weekdays(1-5), annoy Joe. % for newline, last % for cc:
0 22 *   * 1-5 mail -s "It's 10pm" joe%Joe,%%Where are your kids?%.%%
23 */2 1 2 *   echo "run 23 minutes after 0am, 2am, 4am ..., on Feb 1"
5  4 *   * sun echo "run at 04:05 every Sunday"
# run at 03:40 on the first Monday of each month
40 3 1-7 * *   [ "$(date +%a)" == "Mon" ] && command -args
```

Tip

For the system not running continuously, install the `anacron` package to schedule periodic commands at the specified intervals as closely as machine-uptime permits. See anacron(8) and anacrontab(5).

Tip

For scheduled system maintenance scripts, you can run them periodically from root account by placing such scripts in "`/etc/cron.hourly/`", "`/etc/cron.daily/`", "`/etc/cron.weekly/`", or "`/etc/cron.monthly/`". Execution timings of these scripts can be customized by "`/etc/crontab`" and "`/etc/anacrontab`".

Systemd has low level capability to schedule programs to run without `cron` daemon. For example, `/lib/systemd/system/`
and `/lib/systemd/system/apt-daily.service` set up daily apt download activities. See systemd.timer(5) .

9.4.15 Scheduling tasks on event

Systemd can schedule program not only on the timer event but also on the mount event. See Section 10.2.3.3 and Section 10.2.3.2 for examples.

9.4.16 Alt-SysRq key

Pressing Alt-SysRq (PrtScr) followed by one keys does the magic of rescuing control of the system.

key following Alt-SysRq	description of action
k	**k**ill all processes on the **current virtual console** (SAK)
s	**s**ync all mounted filesystems to avoid data corruption
u	remount all mounted filesystems read-only (**u**mount)
r	restore the keyboard from **r**aw mode after X crashes

Table 9.12: List of notable SAK command keys

See more on Linux kernel user's and administrator's guide » Linux Magic System Request Key Hacks

Tip

From SSH terminal etc., you can use the Alt-SysRq feature by writing to the "`/proc/sysrq-trigger`". For example, "`echo s > /proc/sysrq-trigger; echo u > /proc/sysrq-trigger`" from the root shell prompt **s**yncs and **u**mounts all mounted filesystems.

The current (2021) Debian amd64 Linux kernel has `/proc/sys/kernel/sysrq=438=0b110110110`:

- 2 = 0x2 - enable control of console logging level (ON)
- 4 = 0x4 - enable control of keyboard (SAK, unraw) (ON)
- 8 = 0x8 - enable debugging dumps of processes etc. (OFF)
- 16 = 0x10 - enable sync command (ON)
- 32 = 0x20 - enable remount read-only (ON)
- 64 = 0x40 - enable signaling of processes (term, kill, oom-kill) (OFF)
- 128 = 0x80 - allow reboot/poweroff (ON)
- 256 = 0x100 - allow nicing of all RT tasks (ON)

9.5 System maintenance tips

9.5.1 Who is on the system?

You can check who is on the system by the following.

- who(1) shows who is logged on.
- w(1) shows who is logged on and what they are doing.
- last(1) shows listing of last logged in user.
- lastb(1) shows listing of last bad logged in users.

Tip
"`/var/run/utmp`", and "`/var/log/wtmp`" hold such user information. See login(1) and utmp(5).

9.5.2 Warning everyone

You can send message to everyone who is logged on to the system with wall(1) by the following.

```
$ echo "We are shutting down in 1 hour" | wall
```

9.5.3 Hardware identification

For the PCI-like devices (AGP, PCI-Express, CardBus, ExpressCard, etc.), lspci(8) (probably with "`-nn`" option) is a good start for the hardware identification.

Alternatively, you can identify the hardware by reading contents of "`/proc/bus/pci/devices`" or browsing directory tree under "`/sys/bus/pci`" (see Section 1.2.12).

package	popcon	size	description
pciutils	V:264, I:994	289	Linux PCI Utilities: lspci(8)
usbutils	V:90, I:892	322	Linux USB utilities: lsusb(8)
nvme-cli	V:27, I:36	2222	NVMe utilities for Linux: nvme(1)
pcmciautils	V:4.1, I:6.1	92	PCMCIA utilities for Linux: pccardctl(8)
scsitools	V:0.2, I:2.1	261	collection of tools for SCSI hardware management: lsscsi(8)
procinfo	V:0.4, I:5.8	149	system information obtained from "/proc": lsdev(8)
lshw	V:12, I:88	971	information about hardware configuration: lshw(1)
discover	V:24, I:561	81	hardware identification system: discover(8)

Table 9.13: List of hardware identification tools

package	popcon	size	description
console-setup	V:81, I:976	422	Linux console font and keytable utilities
keyd	V:1.5, I:1.7	1437	keyboard key remapping daemon using kernel level input primitives (evdev, uinput)
keyd-application-mapper	V:0.04, I:0.08	34	keyboard key remapping daemon - application-specific remapper
x11-xserver-utils	V:373, I:533	559	X server utilities: xset(1), xmodmap(1)
acpid	V:52, I:78	158	daemon to manage events delivered by the Advanced Configuration and Power Interface (ACPI)
acpi	V:6, I:73	49	utility to display information on ACPI devices
sleepd	V:0.07, I:0.10	84	daemon to put a laptop to sleep during inactivity
hdparm	V:110, I:200	246	hard disk access optimization (see Section 9.6.9)
smartmontools	V:242, I:270	2455	control and monitor storage systems using S.M.A.R.T.
setserial	V:3.0, I:4.9	104	collection of tools for serial port management
memtest86+	V:1, I:19	12483	collection of tools for memory hardware management
scsitools	V:0.2, I:2.1	261	collection of tools for SCSI hardware management
setcd	V:0.05, I:0.61	33	compact disc drive access optimization
big-cursor	I:0.67	26	larger mouse cursors for X

Table 9.14: List of hardware configuration tools

9.5.4 Hardware configuration

Although most of the hardware configuration on modern GUI desktop systems such as GNOME and KDE can be managed through accompanying GUI configuration tools, it is a good idea to know some basics methods to configure them.

For the keyboard and terminal configuration, see Section 8.2 and Section 8.3.

Here, ACPI is a newer framework for the power management system than APM.

Tip
CPU frequency scaling on modern system is governed by kernel modules such as `acpi_cpufreq`.

9.5.5 System and hardware time

The following sets system and hardware time to MM/DD hh:mm, CCYY.

```
# date MMDDhhmmCCYY
# hwclock --utc --systohc
# hwclock --show
```

Times are normally displayed in the local time on the Debian system but the hardware and system time usually use UTC(GMT).

If the hardware time is set to UTC, change the setting to "`UTC=yes`" in the "`/etc/default/rcS`".

The following reconfigure the timezone used by the Debian system.

```
# dpkg-reconfigure tzdata
```

If you wish to update system time via network, consider to use the NTP service with the packages such as `ntp`, `ntpdate`, and `chrony`.

Tip
Under systemd, use `systemd-timesyncd` for the network time synchronization instead. See systemd-timesyncd(8).

See the following.

- Managing Accurate Date and Time HOWTO
- NTP Public Services Project
- The `ntp-doc` package

Tip
ntptrace(8) in the `ntp` package can trace a chain of NTP servers back to the primary source.

9.5.6 The sound infrastructure

Device drivers for sound cards for current Linux are provided by Advanced Linux Sound Architecture (ALSA). ALSA provides emulation mode for previous Open Sound System (OSS) for compatibility.

Application software may be configured not only to access sound devices directly but also to access them via some standardized sound server system. Currently, PulseAudio, JACK, and PipeWire are used as sound server system. See Debian wiki page on Sound for the latest situation.

There is usually a common sound engine for each popular desktop environment. Each sound engine used by the application can choose to connect to different sound servers.

Tip
Use "`cat /dev/urandom > /dev/audio`" or speaker-test(1) to test speaker (^C to stop).

Tip
If you can not get sound, your speaker may be connected to a muted output. Modern sound system has many outputs. alsamixer(1) in the `alsa-utils` package is useful to configure volume and mute settings.

package	popcon	size	description
`alsa-utils`	V:342, I:460	2700	utilities for configuring and using ALSA
`oss-compat`	V:0.6, I:9.2	21	OSS compatibility under ALSA preventing "`/dev/dsp not found`" errors
`pipewire`	V:326, I:370	136	audio and video processing engine multimedia server - metapackage
`pipewire-bin`	V:333, I:370	2251	audio and video processing engine multimedia server - audio server and CLI programs
`pipewire-alsa`	V:172, I:241	194	audio and video processing engine multimedia server - audio server to replace ALSA
`pipewire-pulse`	V:298, I:347	53	audio and video processing engine multimedia server - audio server to replace PulseAudio
`pulseaudio`	V:147, I:171	6616	PulseAudio server
`libpulse0`	V:443, I:579	984	PulseAudio client library
`jackd`	V:2, I:14	8	JACK Audio Connection Kit. (JACK) server (low latency)
`libjack0`	V:1.8, I:8.6	328	JACK Audio Connection Kit. (JACK) library (low latency)
`libphonon4qt5-4`	V:18, I:42	572	Phonon: KDE sound engine

Table 9.15: List of sound packages

9.5.7 Disabling the screen saver

For disabling the screen saver, use following commands.

9.5.8 Disabling beep sounds

One can always unplug the PC speaker to disable beep sounds. Removing `pcspkr` kernel module does this for you.

The following prevents the readline(3) program used by bash(1) to beep when encountering an alert character (ASCII=7).

```
$ echo "set bell-style none">> ~/.inputrc
```

environment	command
The Linux console	`setterm -powersave off`
The X Window (turning off screensaver)	`xset s off`
The X Window (disabling dpms)	`xset -dpms`
The X Window (GUI configuration of screen saver)	`xscreensaver-command -prefs`

Table 9.16: List of commands for disabling the screen saver

9.5.9 Memory usage

There are 2 resources available for you to get the memory usage situation.

- The kernel boot message in the "`/var/log/dmesg`" contains the total exact size of available memory.
- free(1) and top(1) display information on memory resources on the running system.

Here is an example.

```
# grep '\] Memory' /var/log/dmesg
[    0.004000] Memory: 990528k/1016784k available (1975k kernel code, 25868k reserved, 931k ↩
    data, 296k init)
$ free -k
             total       used       free     shared    buffers     cached
Mem:        997184     976928      20256          0     129592     171932
-/+ buffers/cache:     675404     321780
Swap:      4545576          4    4545572
```

You may be wondering "dmesg tells me a free of 990 MB, and free -k says 320 MB is free. More than 600 MB missing …".

Do not worry about the large size of "`used`" and the small size of "`free`" in the "`Mem:`" line, but read the one under them (675404 and 321780 in the example above) and relax.

For my MacBook with 1GB=1048576k DRAM (video system steals some of this), I see the following.

report	size
Total size in dmesg	1016784k = 1GB - 31792k
Free in dmesg	990528k
Total under shell	997184k
Free under shell	20256k (but effectively 321780k)

Table 9.17: List of memory sizes reported

9.5.10 System security and integrity check

Poor system maintenance may expose your system to external exploitation.

For system security and integrity check, you should start with the following.

- The `debsums` package, see debsums(1) and Section 2.5.2.
- The `chkrootkit` package, see chkrootkit(1).
- The `clamav` package family, see clamscan(1) and freshclam(1).

package	popcon	size	description
logcheck	V:4.4, I:5.4	120	daemon to mail anomalies in the system logfiles to the administrator
debsums	V:4, I:29	108	utility to verify installed package files against MD5 checksums
chkrootkit	V:10, I:14	987	rootkit detector
clamav	V:8, I:38	19877	anti-virus utility for Unix - command-line interface
tiger	V:1.0, I:1.3	7800	report system security vulnerabilities
tripwire	V:1.1, I:1.4	5060	file and directory integrity checker
john	V:1.1, I:7.3	469	active password cracking tool
aide	V:2.2, I:2.6	328	Advanced Intrusion Detection Environment - static binary
integrit	V:0.09, I:0.17	2759	file integrity verification program
crack	V:0.06, I:0.55	151	password guessing program

Table 9.18: List of tools for system security and integrity check

- Debian security FAQ.
- Securing Debian Manual.

Here is a simple script to check for typical world writable incorrect file permissions.

```
# find / -perm 777 -a \! -type s -a \! -type l -a \! \( -type d -a -perm 1777 \)
```



Caution
Since the debsums package uses MD5 checksums stored locally, it can not be fully trusted as the system security audit tool against malicious attacks.

9.6 Data storage tips

Booting your system as the Linux live system (see Section 3.2.2 and Section 3.2.3) makes it easy for you to reconfigure data storage on the installed system.

Note
Statements on hard disk (HDD) are applicable to other storage devices such as SSD / USB flash drive / Memory card / ... . Replace device names in examples such as /dev/sda with applicable device names /dev/nvme0, /dev/mmcblk0, ...

You may need to umount(8) some devices manually from the command line before operating on them if they are automatically mounted by the GUI desktop system.

9.6.1 Disk space usage

The disk space usage can be evaluated by programs provided by the mount, coreutils, and xdu packages:

- mount(8) reports all mounted filesystems (= disks).
- df(1) reports the disk space usage for the file system.

- du(1) reports the disk space usage for the directory tree.

Tip
You can feed the output of du(8) to xdu(1) to produce its graphical and interactive presentation with "`du -k . |xdu`", "`sudo du -k -x / |xdu`", etc.

9.6.2 Disk partition configuration

For disk partition configuration, although fdisk(8) has been considered standard, parted(8) deserves some attention. "Disk partitioning data", "partition table", "partition map", and "disk label" are all synonyms.

Older PCs use the classic Master Boot Record (MBR) scheme to hold disk partitioning data in the first sector, i.e., LBA sector 0 (512 bytes).

Recent PCs with Unified Extensible Firmware Interface (UEFI), including Intel-based Macs, use GUID Partition Table (GPT) scheme to hold disk partitioning data not in the first sector.

Although fdisk(8) has been standard for the disk partitioning tool, parted(8) is replacing it.

package	popcon	size	description
`util-linux`	V:913, I:1000	4826	miscellaneous system utilities including fdisk(8) and cfdisk(8)
`parted`	V:449, I:569	126	GNU Parted disk partition resizing program
`gparted`	V:12, I:83	2313	GNOME partition editor based on `libparted`
`gdisk`	V:19, I:253	967	partition editor for the GPT/MBR hybrid disk
`kpartx`	V:15, I:24	78	program to create device mappings for partitions

Table 9.19: List of disk partition management packages



Caution
Although parted(8) claims to create and to resize filesystem too, it is safer to do such things using best maintained specialized tools such as mkfs(8) (mkfs.msdos(8), mkfs.ext2(8), mkfs.ext3(8), mkfs.ext4(8), …) and resize2fs(8).

Note
In order to switch between GPT and MBR, you need to erase first few blocks of disk contents directly (see Section 9.8.6) and use "`parted /dev/sdx mklabel gpt`" or "`parted /dev/sdx mklabel msdos`" to set it. Please note "`msdos`" is use here for MBR.

9.6.3 Accessing partition using UUID

Although reconfiguration of your partition or activation order of removable storage media may yield different names for partitions, you can access them consistently. This is also helpful if you have multiple disks and your BIOS/UEFI doesn't give them consistent device names.

- mount(8) with "`-U`" option can mount a block device using UUID, instead of using its file name such as "`/dev/sda3`".
- "`/etc/fstab`" (see fstab(5)) can use UUID.
- Boot loaders (Section 3.1.2) may use UUID too.

Tip
You can probe UUID of a block special device with blkid(8).
You can also probe UUID and other information with "`lsblk -f`".

9.6.4 LVM2

LVM2 is a logical volume manager for the Linux kernel. With LVM2, disk partitions can be created on logical volumes instead of the physical harddisks.

LVM requires the following.

- device-mapper support in the Linux kernel (default for Debian kernels)
- the userspace device-mapper support library (`libdevmapper*` package)
- the userspace LVM2 tools (`lvm2` package)

Please start learning LVM2 from the following manpages.

- lvm(8): Basics of LVM2 mechanism (list of all LVM2 commands)
- lvm.conf(5): Configuration file for LVM2
- lvs(8): Report information about logical volumes
- vgs(8): Report information about volume groups
- pvs(8): Report information about physical volumes

9.6.5 Filesystem configuration

For ext4 filesystem, the `e2fsprogs` package provides the following.

- mkfs.ext4(8) to create new ext4 filesystem
- fsck.ext4(8) to check and to repair existing ext4 filesystem
- tune2fs(8) to configure superblock of ext4 filesystem
- debugfs(8) to debug ext4 filesystem interactively. (It has `undel` command to recover deleted files.)

The mkfs(8) and fsck(8) commands are provided by the `e2fsprogs` package as front-ends to various filesystem dependent programs (`mkfs.fstype` and `fsck.fstype`). For ext4 filesystem, they are mkfs.ext4(8) and fsck.ext4(8) (they are symlinked to mke2fs(8) and e2fsck(8)).

Similar commands are available for each filesystem supported by Linux.

Tip
Ext4 filesystem is the default filesystem for the Linux system and strongly recommended to use it unless you have some specific reasons not to.
Btrfs status can be found at Debian wiki on btrfs and kernel.org wiki on btrfs. It is expected to be the next default filesystem after the ext4 filesystem.
Some tools allow access to filesystem without Linux kernel support (see Section 9.8.2).

package	popcon	size	description
e2fsprogs	V:821, I:997	1560	utilities for the ext2/ext3/ext4 filesystems
btrfs-progs	V:52, I:79	5255	utilities for the Btrfs filesystem
reiserfsprogs	V:11, I:21	473	utilities for the Reiserfs filesystem
zfsutils-linux	V:32, I:33	1905	utilities for the OpenZFS filesystem
dosfstools	V:265, I:568	310	utilities for the FAT filesystem. (Microsoft: MS-DOS, Windows)
exfatprogs	V:36, I:466	580	utilities for the exFAT filesystem maintained by Samsung.
exfat-fuse	V:2, I:40	73	read/write exFAT filesystem (Microsoft) driver for FUSE.
xfsprogs	V:42, I:84	4651	utilities for the XFS filesystem. (SGI: IRIX)
ntfs-3g	V:208, I:514	1507	read/write NTFS filesystem (Microsoft: Windows NT, …) driver for FUSE.
jfsutils	V:0.4, I:6.7	1104	utilities for the JFS filesystem. (IBM: AIX, OS/2)
xorriso	V:15, I:65	378	utilities for the ISO-9660 filesystem and CD/DVD writing from libburnia
wodim	V:7, I:83	898	command line CD/DVD writing tool from cdrkit
genisoimage	V:17, I:153	1567	command line ISO-9660 filesystem tool from cdrkit
reiser4progs	V:0.1, I:1.1	1367	utilities for the Reiser4 filesystem
hfsprogs	V:0.3, I:3.1	394	utilities for HFS and HFS Plus filesystem. (Apple: Mac OS)
zerofree	V:7, I:116	30	program to zero free blocks from ext2/3/4 filesystems

Table 9.20: List of filesystem management packages

9.6.6 Filesystem creation and integrity check

The mkfs(8) command creates the filesystem on a Linux system. The fsck(8) command provides the filesystem integrity check and repair on a Linux system.

Debian now defaults to no periodic `fsck` after filesystem creation.



Caution
It is generally not safe to run `fsck` on **mounted filesystems**.

Tip
You can run the fsck(8) command safely on all filesystems including root filesystem on reboot by setting "`enable_periodic_fsck`" in "`/etc/mke2fs.conf`" and the max mount count to 0 using "`tune2fs -c0 /dev/`*`partition_name`*". See mke2fs.conf(5) and tune2fs(8).
Check files in "`/var/log/fsck/`" for the result of the fsck(8) command run from the boot script.

9.6.7 Optimization of filesystem by mount options

The basic static filesystem configuration is given by "`/etc/fstab`". For example,

```
«file system»                    «mount point» «type» «options»    «dump» «pass»
proc                                      /proc proc   defaults          0 0
UUID=709cbe4c-80c1-56db-8ab1-dbce3146d2f7 /     ext4   errors=remount-ro 0 1
UUID=817bae6b-45d2-5aca-4d2a-1267ab46ac23 none  swap   sw                0 0
/dev/scd0                          /media/cdrom0  udf,iso9660 user,noauto  0 0
```

Tip
UUID (see Section 9.6.3) may be used to identify a block device instead of normal block device names such as "`/dev/sda1`", "`/dev/sda2`", …

Since Linux 2.6.30, the kernel defaults to the behavior provided by "`relatime`" option.

See fstab(5) and mount(8).

9.6.8 Optimization of filesystem via superblock

Characteristics of a filesystem can be optimized via its superblock using the tune2fs(8) command.

- Execution of "`sudo tune2fs -l /dev/sda1`" displays the contents of the filesystem superblock on "`/dev/sda1`".
- Execution of "`sudo tune2fs -c 50 /dev/sda1`" changes frequency of filesystem checks (`fsck` execution during boot-up) to every 50 boots on "`/dev/sda1`".
- Execution of "`sudo tune2fs -j /dev/sda1`" adds journaling capability to the filesystem, i.e. filesystem conversion from ext2 to ext3 on "`/dev/sda1`". (Do this on the unmounted filesystem.)
- Execution of "`sudo tune2fs -O extents,uninit_bg,dir_index /dev/sda1 && fsck -pf /dev/sda1`" converts it from ext3 to ext4 on "`/dev/sda1`". (Do this on the unmounted filesystem.)

Tip
Despite its name, tune2fs(8) works not only on the ext2 filesystem but also on the ext3 and ext4 filesystems.

9.6.9 Optimization of hard disk



Warning
Please check your hardware and read manpage of hdparm(8) before playing with hard disk configuration because this may be quite dangerous for the data integrity.

You can test disk access speed of a hard disk, e.g. "`/dev/sda`", by "`hdparm -tT /dev/sda`". For some hard disk connected with (E)IDE, you can speed it up with "`hdparm -q -c3 -d1 -u1 -m16 /dev/sda`" by enabling the "(E)IDE 32-bit I/O support", enabling the "using_dma flag", setting "interrupt-unmask flag", and setting the "multiple 16 sector I/O" (dangerous!).

You can test write cache feature of a hard disk, e.g. "`/dev/sda`", by "`hdparm -W /dev/sda`". You can disable its write cache feature with "`hdparm -W 0 /dev/sda`".

You may be able to read badly pressed CDROMs on modern high speed CD-ROM drive by slowing it down with "`setcd -x 2`".

9.6.10 Optimization of solid state drive

Solid state drive (SSD) is auto detected now.

Reduce unnecessary disk accesses to prevent disk wear out by mounting "`tmpfs`" on volatile data path in `/etc/fstab`.

9.6.11 Using SMART to predict hard disk failure

You can monitor and log your hard disk which is compliant to SMART with the smartd(8) daemon.

1. Enable SMART feature in BIOS.

2. Install the `smartmontools` package.
3. Identify your hard disk drives by listing them with df(1).
   - Let's assume a hard disk drive to be monitored as "`/dev/sda`".
4. Check the output of "`smartctl -a /dev/sda`" to see if SMART feature is actually enabled.
   - If not, enable it by "`smartctl -s on -a /dev/sda`".
5. Enable smartd(8) daemon to run by the following.
   - uncomment "`start_smartd=yes`" in the "`/etc/default/smartmontools`" file.
   - restart the smartd(8) daemon by "`sudo systemctl restart smartmontools`".

Tip
The smartd(8) daemon can be customized with the `/etc/smartd.conf` file including how to be notified of warnings.

9.6.12 Specify temporary storage directory via $TMPDIR

Applications create temporary files normally under the temporary storage directory "`/tmp`". If "`/tmp`" does not provide enough space, you can specify such temporary storage directory via the `$TMPDIR` variable for well-behaving programs.

9.6.13 Expansion of usable storage space via LVM

For partitions created on Logical Volume Manager (LVM) (Linux feature) at install time, they can be resized easily by concatenating extents onto them or truncating extents from them over multiple storage devices without major system reconfiguration.

9.6.14 Expansion of usable storage space by mounting another partition

If you have an empty partition (e.g., "`/dev/sdx`"), you can format it with mkfs.ext4(1) and mount(8) it to a directory where you need more space. (You need to copy original data contents.)

```
$ sudo mv work-dir old-dir
$ sudo mkfs.ext4 /dev/sdx
$ sudo mount -t ext4 /dev/sdx work-dir
$ sudo cp -a old-dir/* work-dir
$ sudo rm -rf old-dir
```

Tip
You may alternatively mount an empty disk image file (see Section 9.7.5) as a loop device (see Section 9.7.3). The actual disk usage grows with the actual data stored.

9.6.15 Expansion of usable storage space by bind-mounting another directory

If you have an empty directory (e.g., "`/path/to/emp-dir`") on another partition with usable space, you can mount(8) it with "`--bind`" option to a directory (e.g., "`work-dir`") where you need more space.

```
$ sudo mount --bind /path/to/emp-dir work-dir
```

9.6.16 Expansion of usable storage space by overlay-mounting another directory

If you have usable space in another partition (e.g., "`/path/to/empty`" and "`/path/to/work`"), you can create a directory in it and stack that on to an old directory (e.g., "`/path/to/old`") where you need space using the OverlayFS for Linux kernel 3.18 or newer (Debian Stretch 9.0 or newer).

```
$ sudo mount -t overlay overlay \
  -olowerdir=/path/to/old-dir,upperdir=/path/to/empty,workdir=/path/to/work
```

Here, "`/path/to/empty`" and "`/path/to/work`" should be on the RW-enabled partition to write on "`/path/to/old`".

9.6.17 Expansion of usable storage space using symlink



Caution
This is a deprecated method. Some software may not function well with "symlink to a directory". Instead, use the "mounting" approaches described in the above.

If you have an empty directory (e.g., "`/path/to/emp-dir`") in another partition with usable space, you can create a symlink to the directory with ln(8).

```
$ sudo mv work-dir old-dir
$ sudo mkdir -p /path/to/emp-dir
$ sudo ln -sf /path/to/emp-dir work-dir
$ sudo cp -a old-dir/* work-dir
$ sudo rm -rf old-dir
```



Warning
Do not use "symlink to a directory" for directories managed by the system such as "`/opt`". Such a symlink may be overwritten when the system is upgraded.

9.7 The disk image

Here, we discuss manipulations of the disk image.

9.7.1 Making the disk image file

The disk image file, "`disk.img`", of an unmounted device, e.g., the second SCSI or serial ATA drive "`/dev/sdb`", can be made by one of the following.

```
# dd if=/dev/sdb of=disk.img; sync
```

```
# cp /dev/sdb disk.img ; sync
```

```
# cat /dev/sdb > disk.img ; sync
```

The disk image of the traditional PC's master boot record (MBR) (see Section 9.6.2) which reside on the first sector on the primary IDE disk can be made by using dd(1) by the following.

```
# dd if=/dev/sda of=mbr.img bs=512 count=1
# dd if=/dev/sda of=mbr-nopart.img bs=446 count=1
# dd if=/dev/sda of=mbr-part.img skip=446 bs=1 count=66
```

- ”`mbr.img`”: The MBR with the partition table
- ”`mbr-nopart.img`”: The MBR without the partition table
- ”`mbr-part.img`”: The partition table of the MBR only

If you are making an image of a disk partition of the original disk, substitute ”`/dev/sda`” with ”`/dev/sda1`” etc.

9.7.2 Writing directly to the disk

The disk image file, ”`disk.img`” can be written to an unmounted device, e.g., the second SCSI drive ”`/dev/sdb`” with matching size, by one of the following.

```
# dd if=disk.img of=/dev/sdb ; sync
```

```
# cp disk.img /dev/sdb ; sync
```

```
# cat disk.img >/dev/sdb ; sync
```

Similarly, the disk partition image file, ”`partition.img`” can be written to an unmounted partition, e.g., the first partition of the second SCSI drive ”`/dev/sdb1`” with matching size, by the following.

```
# dd if=partition.img of=/dev/sdb1 ; sync
```

9.7.3 Mounting the disk image file

The disk image ”`partition.img`” containing a single partition image can be mounted and unmounted by using the loop device as follows.

```
# losetup --show -f partition.img
/dev/loop0
# mkdir -p /mnt/loop0
# mount -t auto /dev/loop0 /mnt/loop0
...hack...hack...hack
# umount /dev/loop0
# losetup -d /dev/loop0
```

This can be simplified as follows.

```
# mkdir -p /mnt/loop0
# mount -t auto -o loop partition.img /mnt/loop0
...hack...hack...hack
# umount partition.img
```

Each partition of the disk image ”`disk.img`” containing multiple partitions can be mounted by using the loop device.

```
# losetup --show -f -P disk.img
/dev/loop0
# ls -l /dev/loop0*
brw-rw---- 1 root disk   7,  0 Apr  2 22:51 /dev/loop0
brw-rw---- 1 root disk 259, 12 Apr  2 22:51 /dev/loop0p1
```

```
brw-rw---- 1 root disk 259, 13 Apr  2 22:51 /dev/loop0p14
brw-rw---- 1 root disk 259, 14 Apr  2 22:51 /dev/loop0p15
# fdisk -l /dev/loop0
Disk /dev/loop0: 2 GiB, 2147483648 bytes, 4194304 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: gpt
Disk identifier: 6A1D9E28-C48C-2144-91F7-968B3CBC9BD1

Device         Start     End Sectors  Size Type
/dev/loop0p1  262144 4192255 3930112  1.9G Linux root (x86-64)
/dev/loop0p14   2048    8191    6144    3M BIOS boot
/dev/loop0p15   8192  262143  253952  124M EFI System

Partition table entries are not in disk order.
# mkdir -p /mnt/loop0p1
# mkdir -p /mnt/loop0p15
# mount -t auto /dev/loop0p1 /mnt/loop0p1
# mount -t auto /dev/loop0p15 /mnt/loop0p15
# mount |grep loop
/dev/loop0p1 on /mnt/loop0p1 type ext4 (rw,relatime)
/dev/loop0p15 on /mnt/loop0p15 type vfat (rw,relatime,fmask=0002,dmask=0002,allow_utime ↩
    =0020,codepage=437,iocharset=ascii,shortname=mixed,utf8,errors=remount-ro)
...hack...hack...hack
# umount /dev/loop0p1
# umount /dev/loop0p15
# losetup -d /dev/loop0
```

Alternatively, similar effects can be done by using the device mapper devices created by kpartx(8) from the `kpartx` package as follows.

```
# kpartx -a -v disk.img
add map loop0p1 (253:0): 0 3930112 linear 7:0 262144
add map loop0p14 (253:1): 0 6144 linear 7:0 2048
add map loop0p15 (253:2): 0 253952 linear 7:0 8192
# fdisk -l /dev/loop0
Disk /dev/loop0: 2 GiB, 2147483648 bytes, 4194304 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: gpt
Disk identifier: 6A1D9E28-C48C-2144-91F7-968B3CBC9BD1

Device         Start     End Sectors  Size Type
/dev/loop0p1  262144 4192255 3930112  1.9G Linux root (x86-64)
/dev/loop0p14   2048    8191    6144    3M BIOS boot
/dev/loop0p15   8192  262143  253952  124M EFI System

Partition table entries are not in disk order.
# ls -l /dev/mapper/
total 0
crw------- 1 root root 10, 236 Apr  2 22:45 control
lrwxrwxrwx 1 root root       7 Apr  2 23:19 loop0p1 -> ../dm-0
lrwxrwxrwx 1 root root       7 Apr  2 23:19 loop0p14 -> ../dm-1
lrwxrwxrwx 1 root root       7 Apr  2 23:19 loop0p15 -> ../dm-2
# mkdir -p /mnt/loop0p1
# mkdir -p /mnt/loop0p15
# mount -t auto /dev/mapper/loop0p1 /mnt/loop0p1
# mount -t auto /dev/mapper/loop0p15 /mnt/loop0p15
# mount |grep loop
/dev/loop0p1 on /mnt/loop0p1 type ext4 (rw,relatime)
```

```
/dev/loop0p15 on /mnt/loop0p15 type vfat (rw,relatime,fmask=0002,dmask=0002,allow_utime ↩
    =0020,codepage=437,iocharset=ascii,shortname=mixed,utf8,errors=remount-ro)
...hack...hack...hack
# umount /dev/mapper/loop0p1
# umount /dev/mapper/loop0p15
# kpartx -d disk.img
```

9.7.4 Cleaning a disk image file

A disk image file, "`disk.img`" can be cleaned of all removed files into clean sparse image "`new.img`" by the following.

```
# mkdir old; mkdir new
# mount -t auto -o loop disk.img old
# dd bs=1 count=0 if=/dev/zero of=new.img seek=5G
# mount -t auto -o loop new.img new
# cd old
# cp -a --sparse=always ./ ../new/
# cd ..
# umount new.img
# umount disk.img
```

If "`disk.img`" is in ext2, ext3 or ext4, you can also use zerofree(8) from the `zerofree` package as follows.

```
# losetup --show -f disk.img
/dev/loop0
# zerofree /dev/loop0
# cp --sparse=always disk.img new.img
# losetup -d /dev/loop0
```

9.7.5 Making the empty disk image file

The empty disk image "`disk.img`" which can grow up to 5GiB can be made using dd(1) as follows.

```
$ dd bs=1 count=0 if=/dev/zero of=disk.img seek=5G
```

Instead of using dd(1), specialized fallocate(8) may be used here.

You can create an ext4 filesystem on this disk image "`disk.img`" using the loop device as follows.

```
# losetup --show -f disk.img
/dev/loop0
# mkfs.ext4 /dev/loop0
...hack...hack...hack
# losetup -d /dev/loop0
$ du  --apparent-size -h disk.img
5.0G  disk.img
$ du -h disk.img
83M disk.img
```

For "`disk.img`", its file size is 5.0 GiB and its actual disk usage is mere 83MiB. This discrepancy is possible since ext4 can hold sparse file.

Tip
The actual disk usage of sparse file grows with data which are written to it.

Using similar operation on devices created by the loop device or the device mapper devices as Section 9.7.3, you can partition this disk image "`disk.img`" using parted(8) or fdisk(8), and can create filesystem on it using mkfs.ext4(8), mkswap(8), etc.

9.7.6 Making the ISO9660 image file

Tip
Both genisoimage(1) provided by cdrkit and xorrisofs(1) provided by Libburnia share the same command syntax except for the command name.

The ISO9660 image file, "`cd.iso`", from the source directory tree at "`source_directory`" can be made using genisoimage(1) provided by cdrkit by the following.

```
#  genisoimage -r -J -T -V volume_id -o cd.iso source_directory
```

Similarly, the bootable ISO9660 image file, "`cdboot.iso`", can be made from `debian-installer` like directory tree at "`source_directory`" by the following.

```
#  genisoimage -r -o cdboot.iso -V volume_id \
   -b isolinux/isolinux.bin -c isolinux/boot.cat \
   -no-emul-boot -boot-load-size 4 -boot-info-table source_directory
```

Here Isolinux boot loader (see Section 3.1.2) is used for booting.

You can calculate the md5sum value and make the ISO9660 image directly from the CD-ROM device as follows.

```
$ isoinfo -d -i /dev/cdrom
CD-ROM is in ISO 9660 format
...
Logical block size is: 2048
Volume size is: 23150592
...
# dd if=/dev/cdrom bs=2048 count=23150592 conv=notrunc,noerror | md5sum
# dd if=/dev/cdrom bs=2048 count=23150592 conv=notrunc,noerror > cd.iso
```



Warning
You must carefully avoid ISO9660 filesystem read ahead bug of Linux as above to get the right result.

9.7.7 Writing directly to the CD/DVD-R/RW

Tip
DVD is only a large CD to wodim(1) provided by cdrkit and xorrecord(1) provided by Libburnia. These commands share the same command syntax except for the command name.

You can find a usable device by the following.

```
# wodim --devices
```

Then the blank CD-R is inserted to the CD drive, and the ISO9660 image file, "`cd.iso`" is written to this device, e.g., "`/dev/sda`", using wodim(1) by the following.

```
# wodim -v -eject dev=/dev/sda cd.iso
```

If CD-RW is used instead of CD-R, do this instead by the following.

```
# wodim -v -eject blank=fast dev=/dev/sda cd.iso
```

Tip
If your desktop system mounts CDs automatically, unmount it by "`sudo umount /dev/sda`" from console before using wodim(1).

9.7.8 Mounting the ISO9660 image file

If "`cd.iso`" contains an ISO9660 image, then the following manually mounts it to "`/cdrom`".

```
# mount -t iso9660 -o ro,loop cd.iso /cdrom
```

Tip
Modern desktop system may mount removable media such as ISO9660 formatted CD automatically (see Section 10.1.7).

9.8 The binary data

Here, we discuss direct manipulations of the binary data on storage media.

9.8.1 Viewing and editing binary data

The most basic viewing method of binary data is to use "`od -t x1`" command.

package	popcon	size	description
coreutils	V:912, I:1000	17994	basic package which has od(1) to dump files (HEX, ASCII, OCTAL, …)
bsdmainutils	V:4, I:121	17	utility package which has hd(1) to dump files (HEX, ASCII, OCTAL, …)
hexedit	V:0.8, I:7.5	70	binary editor and viewer (HEX, ASCII)
bless	V:0.1, I:1.2	924	full featured hexadecimal editor (GNOME)
okteta	V:1, I:11	1591	full featured hexadecimal editor (KDE4)
ncurses-hexedit	V:0.1, I:1.1	130	binary editor and viewer (HEX, ASCII, EBCDIC)
beav	V:0.04, I:0.26	137	binary editor and viewer (HEX, ASCII, EBCDIC, OCTAL, …)

Table 9.21: List of packages which view and edit binary data

Tip
HEX is used as an acronym for hexadecimal format with radix 16. OCTAL is for octal format with radix 8. ASCII is for American Standard Code for Information Interchange, i.e., normal English text code. EBCDIC is for Extended Binary Coded Decimal Interchange Code used on IBM mainframe operating systems.

9.8.2 Manipulating files without mounting disk

There are tools to read and write files without mounting disk.

package	popcon	size	description
mtools	V:7, I:52	400	utilities for MSDOS files without mounting them
hfsutils	V:0.1, I:2.5	178	utilities for HFS and HFS+ files without mounting them

Table 9.22: List of packages to manipulate files without mounting disk

9.8.3 Data redundancy

Software RAID systems offered by the Linux kernel provide data redundancy in the kernel filesystem level to achieve high levels of storage reliability.

There are tools to add data redundancy to files in application program level to achieve high levels of storage reliability, too.

package	popcon	size	description
par2	V:9, I:116	330	Parity Archive Volume Set, for checking and repair of files
dvdisaster	V:0.1, I:1.1	1422	data loss/scratch/aging protection for CD/DVD media
dvbackup	V:0.01, I:0.03	413	backup tool using MiniDV camcorders (providing rsbep(1))

Table 9.23: List of tools to add data redundancy to files

9.8.4 Data file recovery and forensic analysis

There are tools for data file recovery and forensic analysis.

package	popcon	size	description
testdisk	V:2, I:25	528	utilities for partition scan and disk recovery
magicrescue	V:0.2, I:1.7	257	utility to recover files by looking for magic bytes
scalpel	V:0.2, I:2.2	89	frugal, high performance file carver
myrescue	V:0.2, I:1.9	81	rescue data from damaged harddisks
extundelete	V:0.5, I:7.3	152	utility to undelete files on the ext3/4 filesystem
ext4magic	V:0.2, I:3.5	235	utility to undelete files on the ext3/4 filesystem
ext3grep	V:0.2, I:1.8	299	tool to help recover deleted files on the ext3 filesystem
scrounge-ntfs	V:0.1, I:1.6	49	data recovery program for NTFS filesystems
gzrt	V:0.02, I:0.43	34	gzip recovery toolkit
sleuthkit	V:3, I:25	1119	tools for forensics analysis. (Sleuthkit)
autopsy	V:0.1, I:1.1	1026	graphical interface to SleuthKit
foremost	V:0.3, I:3.8	101	forensics application to recover data
guymager	V:0.08, I:0.71	1049	forensic imaging tool based on Qt
dcfldd	V:0.3, I:2.6	113	enhanced version of dd for forensics and security

Table 9.24: List of packages for data file recovery and forensic analysis

Tip
You can undelete files on the ext2 filesystem using `list_deleted_inodes` and `undel` commands of debugfs(8) in the `e2fsprogs` package.

9.8.5 Splitting a large file into small files

When a data is too big to backup as a single file, you can backup its content after splitting it into, e.g. 2000MiB chunks and merge those chunks back into the original file later.

```
$ split -b 2000m large_file
$ cat x* >large_file
```



Caution
Please make sure you do not have any files starting with "x" to avoid name crashes.

9.8.6 Clearing file contents

In order to clear the contents of a file such as a log file, do not use rm(1) to delete the file and then create a new empty file, because the file may still be accessed in the interval between commands. The following is the safe way to clear the contents of the file.

```
$ :>file_to_be_cleared
```

9.8.7 Dummy files

The following commands create dummy or empty files.

```
$ dd if=/dev/zero    of=5kb.file bs=1k count=5
$ dd if=/dev/urandom of=7mb.file bs=1M count=7
$ touch zero.file
$ : > alwayszero.file
```

You should find following files.

- "`5kb.file`" is 5KB of zeros.
- "`7mb.file`" is 7MB of random data.
- "`zero.file`" may be a 0 byte file. If it existed, its `mtime` is updated while its content and its length are kept.
- "`alwayszero.file`" is always a 0 byte file. If it existed, its `mtime` is updated and its content is reset.

9.8.8 Erasing an entire hard disk

There are several ways to completely erase data from an entire hard disk like device, e.g., USB flash drive at "`/dev/sda`".



Caution
Check your USB flash drive location with mount(8) first before executing commands here. The device pointed by "/dev/sda" may be SCSI hard disk or serial-ATA hard disk where your entire system resides.

Erase all the disk content by resetting data to 0 with the following.

```
# dd if=/dev/zero of=/dev/sda
```

Erase everything by overwriting with random data as follows.

```
# dd if=/dev/urandom of=/dev/sda
```

Erase everything by overwriting with random data very efficiently as follows.

```
# shred -v -n 1 /dev/sda
```

You may alternatively use badblocks(8) with `-t random` option.

Since dd(1) is available from the shell of many bootable Linux CDs such as Debian installer CD, you can erase your installed system completely by running an erase command from such media on the system hard disk, e.g., "/dev/sda", "/dev/sda", etc.

9.8.9 Erasing unused area of an hard disk

Unused area on an hard disk (or USB flash drive), e.g. "/dev/sdb1" may still contain erased data themselves since they are only unlinked from the filesystem. These can be cleaned by overwriting them.

```
# mount -t auto /dev/sdb1 /mnt/foo
# cd /mnt/foo
# dd if=/dev/zero of=junk
dd: writing to `junk': No space left on device
...
# sync
# umount /dev/sdb1
```



Warning
This is usually good enough for your USB flash drive. But this is not perfect. Most parts of erased filenames and their attributes may be hidden and remain in the filesystem.

9.8.10 Undeleting deleted but still open files

Even if you have accidentally deleted a file, as long as that file is still being used by some application (read or write mode), it is possible to recover such a file.

For example, try the following

```
$ echo foo > bar
$ less bar
$ ps aux | grep ' less[ ]'
bozo    4775  0.0  0.0  92200   884 pts/8    S+   00:18   0:00 less bar
$ rm bar
$ ls -l /proc/4775/fd | grep bar
lr-x------ 1 bozo bozo 64 2008-05-09 00:19 4 -> /home/bozo/bar (deleted)
```

```
$ cat /proc/4775/fd/4 >bar
$ ls -l
-rw-r--r-- 1 bozo bozo 4 2008-05-09 00:25 bar
$ cat bar
foo
```

Execute on another terminal (when you have the `lsof` package installed) as follows.

```
$ ls -li bar
2228329 -rw-r--r-- 1 bozo bozo 4 2008-05-11 11:02 bar
$ lsof |grep bar|grep less
less 4775 bozo 4r REG 8,3 4 2228329 /home/bozo/bar
$ rm bar
$ lsof |grep bar|grep less
less 4775 bozo 4r REG 8,3 4 2228329 /home/bozo/bar (deleted)
$ cat /proc/4775/fd/4 >bar
$ ls -li bar
2228302 -rw-r--r-- 1 bozo bozo 4 2008-05-11 11:05 bar
$ cat bar
foo
```

9.8.11 Searching all hardlinks

Files with hardlinks can be identified by "`ls -li`".

```
$ ls -li
total 0
2738405 -rw-r--r-- 1 root root 0 2008-09-15 20:21 bar
2738404 -rw-r--r-- 2 root root 0 2008-09-15 20:21 baz
2738404 -rw-r--r-- 2 root root 0 2008-09-15 20:21 foo
```

Both "`baz`" and "`foo`" have link counts of "2" (>1) showing them to have hardlinks. Their inode numbers are common "2738404". This means they are the same hardlinked file. If you do not happen to find all hardlinked files by chance, you can search it by the inode, e.g., "2738404" as the following.

```
# find /path/to/mount/point -xdev -inum 2738404
```

9.8.12 Invisible disk space consumption

All deleted but open files consume disk space although they are not visible from normal du(1). They can be listed with their size by the following.

```
# lsof -s -X / |grep deleted
```

9.9 Data encryption tips

With physical access to your PC, anyone can easily gain root privilege and access all the files on your PC (see Section 4.6.4). This means that login password system can not secure your private and sensitive data against possible theft of your PC. You must deploy data encryption technology to do it. Although GNU privacy guard (see Section 10.3) can encrypt files, it takes some user efforts.

Dm-crypt facilitates automatic data encryption via native Linux kernel modules with minimal user efforts using device-mapper.

package	popcon	size	description
cryptsetup	V:34, I:77	469	utilities for encrypted block device (dm-crypt / LUKS)
cryptmount	V:1.8, I:2.3	236	utilities for encrypted block device (dm-crypt / LUKS) with focus on mount/unmount by normal users
fscrypt	V:0.4, I:1.1	6844	utilities for Linux filesystem encryption (fscrypt)
libpam-fscrypt	V:0.31, I:0.34	6941	PAM module for Linux filesystem encryption (fscrypt)

Table 9.25: List of data encryption utilities



Caution
Data encryption costs CPU time etc. Encrypted data becomes inaccessible if its password is lost. Please weigh its benefits and costs.

Note
Entire Debian system can be installed on a encrypted disk by the debian-installer (lenny or newer) using dm-crypt/LUKS and initramfs.

Tip
See Section 10.3 for user space encryption utility: GNU Privacy Guard.

9.9.1 Removable disk encryption with dm-crypt/LUKS

You can encrypt contents of removable mass devices, e.g. USB flash drive on "`/dev/sdx`", using dm-crypt/LUKS. You simply format it as the following.

```
# fdisk /dev/sdx
... "n" "p" "1" "return" "return" "w"
# cryptsetup luksFormat /dev/sdx1
...
# cryptsetup open /dev/sdx1 secret
...
# ls -l /dev/mapper/
total 0
crw-rw---- 1 root root  10, 60 2021-10-04 18:44 control
lrwxrwxrwx 1 root root       7 2021-10-04 23:55 secret -> ../dm-0
# mkfs.vfat /dev/mapper/secret
...
# cryptsetup close secret
```

Then, it can be mounted just like normal one on to "`/media/`*`username/disk_label`*", except for asking password (see Section 10.1.7) under modern desktop environment using the `udisks2` package. The difference is that every data written to it is encrypted. The password entry may be automated using keyring (see Section 10.3.6).

You may alternatively format media in different filesystem, e.g., ext4 with "`mkfs.ext4 /dev/mapper/sdx1`". If btrfs is used instead, the `udisks2-btrfs` package needs to be installed. For these filesystems, the file ownership and permissions may need to be configured.

9.9.2 Mounting encrypted disk with dm-crypt/LUKS

For example, an encrypted disk partition created with dm-crypt/LUKS on "`/dev/sdc5`" by Debian Installer can be mounted onto "`/mnt`" as follows:

```
$ sudo cryptsetup open /dev/sdc5 ninja --type luks
Enter passphrase for /dev/sdc5: ****
$ sudo lvm
lvm> lvscan
  inactive          '/dev/ninja-vg/root' [13.52 GiB] inherit
  inactive          '/dev/ninja-vg/swap_1' [640.00 MiB] inherit
  ACTIVE            '/dev/goofy/root' [180.00 GiB] inherit
  ACTIVE            '/dev/goofy/swap' [9.70 GiB] inherit
lvm> lvchange -a y /dev/ninja-vg/root
lvm> exit
  Exiting.
$ sudo mount /dev/ninja-vg/root /mnt
```

9.10 The kernel

Debian distributes modularized Linux kernel as packages for supported architectures.

If you are reading this documentation, you probably don't need to compile Linux kernel by yourself.

9.10.1 Kernel parameters

Many Linux features are configurable via kernel parameters as follows.

- Kernel parameters initialized by the bootloader (see Section 3.1.2)
- Kernel parameters changed by sysctl(8) at runtime for ones accessible via sysfs (see Section 1.2.12)
- Module parameters set by arguments of modprobe(8) when a module is activated (see Section 9.7.3)

See "The Linux kernel user's and administrator's guide » The kernel's command-line parameters" for the detail.

9.10.2 Kernel headers

Most **normal programs** don't need kernel headers and in fact may break if you use them directly for compiling. They should be compiled against the headers in "`/usr/include/linux`" and "`/usr/include/asm`" provided by the `libc6-dev` package (created from the `glibc` source package) on the Debian system.

Note
For compiling some kernel-specific programs such as the kernel modules from the external source and the automounter daemon (`amd`), you must include path to the corresponding kernel headers, e.g. "`-I/usr/src/linux-particular-version/include/`", to your command line.

9.10.3 Compiling the kernel and related modules

Debian has its own method of compiling the kernel and related modules.

If you use `initrd` in Section 3.1.2, make sure to read the related information in initramfs-tools(8), update-initramfs(8), mkinitramfs(8) and initramfs.conf(5).

package	popcon	size	description
build-essential	V:20, I:510	17	essential packages for building Debian packages: make, gcc, …
bzip2	V:163, I:973	113	compress and decompress utilities for bz2 files
libncurses5-dev	I:31	6	developer's libraries and docs for ncurses
git	V:422, I:625	52512	git: distributed revision control system used by the Linux kernel
fakeroot	V:32, I:511	225	provide fakeroot environment for building package as non-root
initramfs-tools	V:495, I:989	49	tool to build an initramfs (Debian specific)
dkms	V:73, I:147	235	dynamic kernel module support (DKMS) (generic)
module-assistant	V:1, I:12	395	helper tool to make module package (Debian specific)
devscripts	V:5, I:31	2783	helper scripts for a Debian Package maintainer (Debian specific)

Table 9.26: List of key packages to be installed for the kernel recompilation on the Debian system



Warning
Do not put symlinks to the directories in the source tree (e.g. "/usr/src/linux*") from "/usr/include/linux" and "/usr/include/asm" when compiling the Linux kernel source. (Some outdated documents suggest this.)

Note
When compiling the latest Linux kernel on the Debian stable system, the use of backported latest tools from the Debian unstable may be needed.
module-assistant(8) (or its short form m-a) helps users to build and install module package(s) easily for one or more custom kernels.
The dynamic kernel module support (DKMS) is a new distribution independent framework designed to allow individual kernel modules to be upgraded without changing the whole kernel. This is used for the maintenance of out-of-tree modules. This also makes it very easy to rebuild modules as you upgrade kernels.

9.10.4 Compiling the kernel source: Debian Kernel Team recommendation

For building custom kernel binary packages from the upstream kernel source, you should use the "deb-pkg" target provided by it.

```
$ sudo apt-get build-dep linux
$ cd /usr/src
$ wget https://mirrors.edge.kernel.org/pub/linux/kernel/v7.x/linux-7.2.8.tar.xz
$ tar --xz -xvf linux-7.2.8.tar.xz
$ cd linux-7.2.8
$ cp /boot/config-7.2.8.config
$ make menuconfig
 ...
$ make deb-pkg
```

Tip
The linux-source-*version* package provides the Linux kernel source with Debian patches as "/usr/src/linux-*version*.tar.bz2".

For building specific binary packages from the Debian kernel source package, you should use the "`binary-arch_`*`architec`* targets in "`debian/rules.gen`".

```
$ sudo apt-get build-dep linux
$ apt-get source linux
$ cd linux-7.*
$ fakeroot make -f debian/rules.gen binary-arch_i386_none_686
```

See further information:

- Debian Wiki: KernelFAQ
- Debian Wiki: DebianKernel
- Debian Linux Kernel Handbook: https://kernel-handbook.debian.net

9.10.5 Hardware drivers and firmware

The hardware driver is the code running on the main CPUs of the target system. Most hardware drivers are available as free software now and are included in the normal Debian kernel packages in the `main` area.

- GPU driver
  - Intel GPU driver (`main`)
  - AMD/ATI GPU driver (`main`)
  - NVIDIA GPU driver (`main` for nouveau driver, and `non-free` for binary-only drivers supported by the vendor.)

The firmware is the code or data loaded on the device attach to the target system (e.g., CPU microcode, rendering code running on GPU, or FPGA / CPLD data, …). Some firmware packages are available as free software but many firmware packages are not available as free software since they contain sourceless binary data. Installing these firmware data is essential for the device to function as expected.

- The firmware data packages containing data loaded to the volatile memory on the target device.
  - `firmware-linux-free` (`main`)
  - `firmware-linux-nonfree` (`non-free-firmware`)
  - `firmware-linux-*` (`non-free-firmware`)
  - `*-firmware` (`non-free-firmware`)
  - `intel-microcode` (`non-free-firmware`)
  - `amd64-microcode` (`non-free-firmware`)
- The firmware update program packages which update data on the non-volatile memory on the target device.
  - fwupd (`main`): Firmware update daemon which downloads firmware data from Linux Vendor Firmware Service.
  - gnome-firmware (`main`): GTK front end for fwupd
  - plasma-discover-backend-fwupd (`main`): Qt front end for fwupd

Please note that access to `non-free-firmware` packages are provided by the official installation media to offer functional installation experience to the user since Debian 12 Bookworm. The `non-free-firmware` area is described in Section 2.1.5.

Please also note that the firmware data downloaded by fwupd from Linux Vendor Firmware Service and loaded to the running Linux kernel may be `non-free`.

9.11 Virtualized system

Use of virtualized system enables us to run multiple instances of system simultaneously on a single hardware.

Tip
See Debian wiki on SystemVirtualization.

9.11.1 Virtualization and emulation tools

There are several virtualization and emulation tool platforms.

- Complete hardware emulation packages such as ones installed by the games-emulator metapackage
- Mostly CPU level emulation with some I/O device emulations such as QEMU
- Mostly CPU level virtualization with some I/O device emulations such as Kernel-based Virtual Machine (KVM)
- OS level container virtualization with the kernel level support such as LXC (Linux Containers), Docker, systemd-nspawn(1), ...
- OS level filesystem access virtualization with the system library call override on the file path such as chroot
- OS level filesystem access virtualization with the system library call override on the file ownership such as fakeroot
- OS API emulation such as Wine
- Interpreter level virtualization with its executable selection and run-time library overrides such as virtualenv and venv for Python

The container virtualization uses Section 4.7.5 and is the backend technology of Section 7.7.

Here are some packages to help you to setup the virtualized system.

See Wikipedia article Comparison of platform virtual machines for detail comparison of different platform virtualization solutions.

9.11.2 Virtualization work flow

Note
Default Debian kernels support KVM since `lenny`.

Typical work flow for virtualization involves several steps.

- Create an empty filesystem (a file tree or a disk image).
  - The file tree can be created by "`mkdir -p /path/to/chroot`".
  - The raw disk image file can be created with dd(1) (see Section 9.7.1 and Section 9.7.5).
  - qemu-img(1) can be used to create and convert disk image files supported by QEMU.
  - The raw and VMDK file format can be used as common format among virtualization tools.
- Mount the disk image with mount(8) to the filesystem (optional).
  - For the raw disk image file, mount it as loop device or device mapper devices (see Section 9.7.3).

package	popcon	size	description
coreutils	V:912, I:1000	17994	GNU core utilities which contain chroot(8)
util-linux	V:913, I:1000	4826	miscellaneous system utilities which contain unshare(1)
systemd-container	V:74, I:77	2906	systemd container/nspawn tools which contain systemd-nspawn(1)
schroot	V:7.2, I:9.1	2222	specialized tool for executing Debian binary packages in chroot
sbuild	V:3.9, I:6.9	155	tool for building Debian binary packages from Debian sources
debootstrap	V:4, I:44	332	bootstrap a basic Debian system (written in sh)
mmdebstrap	V:8, I:12	592	bootstrap a Debian system (written in Perl)
cdebootstrap	V:0.1, I:1.3	114	bootstrap a Debian system (written in C)
cloud-image-utils	V:1, I:14	52	cloud image management utilities
cloud-guest-utils	V:5, I:23	67	cloud guest utilities
virt-manager	V:12, I:48	2309	Virtual Machine Manager: desktop application for managing virtual machines
libvirt-clients	V:45, I:71	1168	programs for the libvirt library
docker.io	V:50, I:53	80705	docker: Linux container runtime
podman	V:34, I:38	85864	podman: engine to run OCI-based containers in Pods
podman-docker	V:2.8, I:3.4	271	engine to run OCI-based containers in Pods - wrapper for docker
incus	V:0.8, I:3.4	24	Incus: system container and virtual machine manager
games-emulator	I:0.19	21	games-emulator: Debian's emulators for games
bochs	V:0.04, I:0.59	8469	Bochs: IA-32 PC emulator
qemu-system	I:20	69	QEMU: full system emulation binaries
qemu-user	V:5.6, I:9.3	472213	QEMU: user mode emulation binaries
qemu-utils	V:14, I:114	12643	QEMU: utilities
qemu-system-x86	V:52, I:98	69257	KVM: full virtualization on x86 hardware with the hardware-assisted virtualization
virtualbox	V:2.9, I:3.7	151079	VirtualBox: x86 virtualization solution on i386 and amd64
gnome-boxes	V:1.2, I:6.1	7074	Boxes: Simple GNOME app to access virtual systems
xen-tools	V:0.1, I:1.4	719	tools to manage debian XEN virtual server
wine	V:13, I:54	204	Wine: Windows API Implementation (standard suite)
dosbox	V:1, I:11	2697	DOSBox: x86 emulator with Tandy/Herc/CGA/EGA/VGA/SVGA graphics, sound and DOS
lxc	V:10, I:13	1681	Linux containers user space tools
python3-venv	V:8, I:159	6	venv for creating virtual python environments (system library)
python3-virtualenv	V:7, I:37	374	virtualenv for creating isolated virtual python environments
pipx	V:8, I:50	4412	pipx for installing python applications in isolated environments

Table 9.27: List of virtualization tools

  - For disk images supported by QEMU, mount them as network block device (see Section 9.11.3).
- Populate the target filesystem with required system data.
  - The use of programs such as `debootstrap` and `cdebootstrap` helps with this process (see Section 9.11.4).
  - Use installers of OSs under the full system emulation.
- Run a program under a virtualized environment.
  - chroot provides basic virtualized environment enough to compile programs, run console applications, and run daemons in it.
  - QEMU provides cross-platform CPU emulation.
  - QEMU with KVM provides full system emulation by the hardware-assisted virtualization.
  - VirtualBox provides full system emulation on i386 and amd64 with or without the hardware-assisted virtualization.

9.11.3 Mounting the virtual disk image file

For the raw disk image file, see Section 9.7.

For other virtual disk image files, you can use qemu-nbd(8) to export them using network block device protocol and mount them using the `nbd` kernel module.

qemu-nbd(8) supports disk formats supported by QEMU: raw, qcow2, qcow, vmdk, vdi, bochs, cow (user-mode Linux copy-on-write), parallels, dmg, cloop, vpc, vvfat (virtual VFAT), and host_device.

The network block device can support partitions in the same way as the loop device (see Section 9.7.3). You can mount the first partition of "`disk.img`" as follows.

```
# modprobe nbd max_part=16
# qemu-nbd -v -c /dev/nbd0 disk.img
...
# mkdir /mnt/part1
# mount /dev/nbd0p1 /mnt/part1
```

Tip
You may export only the first partition of "`disk.img`" using "`-P 1`" option to qemu-nbd(8).

9.11.4 Chroot system

If you wish to try a new Debian environment from a terminal console, I recommend you to use chroot. This enables you to run console applications of Debian `unstable` and `testing` without usual risks associated and without rebooting. chroot(8) is the most basic way.



Caution
Examples below assumes both parent system and chroot system share the same `amd64` CPU architecture.

Although you can manually create a chroot(8) environment using debootstrap(1), this requires non-trivial efforts.

The sbuild package to build Debian packages from source uses the chroot environment managed by the schroot package. It comes with helper script sbuild-createchroot(1). Let's learn how it works by running it as follows.

```
$ sudo mkdir -p /srv/chroot
$ sudo sbuild-createchroot -v --include=eatmydata,ccache unstable /srv/chroot/unstable- ↩
    amd64-sbuild http://deb.debian.org/debian
 ...
```

You see how debootstrap(8) populates system data for `unstable` environment under "`/srv/chroot/unstable-amd64-s` for a minimal build system.

You can login to this environment using schroot(1).

```
$ sudo schroot -v -c chroot:unstable-amd64-sbuild
```

You see how a system shell running under `unstable` environment is created.

Note
The "`/usr/sbin/policy-rc.d`" file which always exits with 101 prevents daemon programs to be started automatically on the Debian system. See "`/usr/share/doc/init-system-helpers/README.policy-rc.d.gz`".

Note
Some programs under chroot may require access to more files from the parent system to function than `sbuild-createchroot` provides as above. For example, "`/sys`", "`/etc/passwd`", "`/etc/group`", "`/var/run/utmp`", "`/var/log/wtmp`", etc. may need to be bind-mounted or copied.

Tip
The `sbuild` package helps to construct a chroot system and builds a package inside the chroot using `schroot` as its backend. It is an ideal system to check build-dependencies. See more on sbuild at Debian wiki and sbuild configuration example in "Guide for Debian Maintainers".

Tip
The systemd-nspawn(1) command helps to run a command or OS in a light-weight container in similar ways to `chroot`. It is more powerful since it uses namespaces to fully virtualize the the process tree, IPC, hostname, domain name and, optionally, networking and user databases. See systemd-nspawn.

9.11.5 Multiple desktop systems

If you wish to try a new GUI Desktop environment of any OS, I recommend you to use QEMU or KVM on a Debian `stable` system to run multiple desktop systems safely using virtualization. These enable you to run any desktop applications including ones of Debian `unstable` and `testing` without usual risks associated with them and without rebooting.

Since pure QEMU is very slow, it is recommended to accelerate it with KVM when the host system supports it.

Virtual Machine Manager also known as `virt-manager` is a convenient GUI tool for managing KVM virtual machines via libvirt.

The virtual disk image "`virtdisk.qcow2`" containing a Debian system for QEMU can be created using debian-installer: Small CDs as follows.

```
$ wget https://cdimage.debian.org/debian-cd/5.0.3/amd64/iso-cd/debian-503-amd64-netinst.iso
$ qemu-img create -f qcow2 virtdisk.qcow2 5G
$ qemu -hda virtdisk.qcow2 -cdrom debian-503-amd64-netinst.iso -boot d -m 256
...
```

Tip
Running other GNU/Linux distributions such as Ubuntu and Fedora under virtualization is a great way to learn configuration tips. Other proprietary OSs may be run nicely under this GNU/Linux virtualization, too.

See more tips at Debian wiki: SystemVirtualization.

Chapter 10

Data management

Tools and tips for managing binary and text data on the Debian system are described.

10.1 Sharing, copying, and archiving



Warning
The uncoordinated write access to actively accessed devices and files from multiple processes must not be done to avoid the race condition. File locking mechanisms using flock(1) may be used to avoid it.

The security of the data and its controlled sharing have several aspects.

- The creation of data archive
- The remote storage access
- The duplication
- The tracking of the modification history
- The facilitation of data sharing
- The prevention of unauthorized file access
- The detection of unauthorized file modification

These can be realized by using some combination of tools.

- Archive and compression tools
- Copy and synchronization tools
- Network filesystems
- Removable storage media
- The secure shell
- The authentication system
- Version control system tools
- Hash and cryptographic encryption tools

10.1.1 Archive and compression tools

Here is a summary of archive and compression tools available on the Debian system.



Warning
Do not set the "$TAPE" variable unless you know what to expect. It changes tar(1) behavior.

- The gzipped tar(1) archive uses the file extension "`.tgz`" or "`.tar.gz`".
- The xz-compressed tar(1) archive uses the file extension "`.txz`" or "`.tar.xz`".
- Popular compression method in FOSS tools such as tar(1) has been moving as follows: `gzip` → `bzip2` → `xz`
- cp(1), scp(1) and tar(1) may have some limitation for special files. cpio(1) is most versatile.
- cpio(1) is designed to be used with find(1) and other commands and suitable for creating backup scripts since the file selection part of the script can be tested independently.
- Internal structure of Libreoffice data files are "`.jar`" file which can be opened also by `unzip`.
- The de-facto cross platform archive tool is `zip`. Use it as "`zip -rX`" to attain the maximum compatibility. Use also the "`-s`" option, if the maximum file size matters.

10.1.2 Copy and synchronization tools

Here is a summary of simple copy and backup tools available on the Debian system.

Copying files with rsync(8) offers richer features than others.

- delta-transfer algorithm that sends only the differences between the source files and the existing files in the destination
- quick check algorithm (by default) that looks for files that have changed in size or in last-modified time
- "`--exclude`" and "`--exclude-from`" options similar to tar(1)
- "a trailing slash on the source directory" syntax that avoids creating an additional directory level at the destination.

Tip
Version control system (VCS) tools in Table 10.14 can function as the multi-way copy and synchronization tools.

10.1.3 Idioms for the archive

Here are several ways to archive and unarchive the entire content of the directory "`./source`" using different tools.

GNU tar(1):

```
$ tar -cvJf archive.tar.xz ./source
$ tar -xvJf archive.tar.xz
```

Alternatively, by the following.

```
$ find ./source -xdev -print0 | tar -cvJf archive.tar.xz --null -T -
```

cpio(1):

```
$ find ./source -xdev -print0 | cpio -ov --null > archive.cpio; xz archive.cpio
$ zcat archive.cpio.xz | cpio -i
```

package	popcon	size	extension	command	comment
tar	V:907, I:1000	3101	.tar	tar(1)	the standard archiver (de facto standard)
cpio	V:352, I:999	1201	.cpio	cpio(1)	Unix System V style archiver, use with find(1)
binutils	V:191, I:631	1131	.ar	ar(1)	archiver for the creation of static libraries
fastjar	V:1.0, I:9.5	182	.jar	fastjar(1)	archiver for Java (zip like)
pax	V:5.1, I:9.0	167	.pax	pax(1)	new POSIX standard archiver, compromise between tar and cpio
gzip	V:895, I:1000	262	.gz	gzip(1), zcat(1), …	GNU LZ77 compression utility (de facto standard)
bzip2	V:163, I:973	113	.bz2	bzip2(1), bzcat(1), …	Burrows-Wheeler block-sorting compression utility with higher compression ratio than gzip(1) (slower than gzip with similar syntax)
lzma	V:0.6, I:9.6	349	.lzma	lzma(1)	LZMA compression utility with higher compression ratio than gzip(1) (deprecated)
xz-utils	V:310, I:981	1551	.xz	xz(1), xzdec(1), …	XZ compression utility with higher compression ratio than bzip2(1) (slower than gzip but faster than bzip2; replacement for LZMA compression utility)
zstd	V:334, I:830	2308	.zstd	zstd(1), zstdcat(1), …	Zstandard fast lossless compression utility
p7zip	V:5, I:172	8	.7z	7zr(1), p7zip(1)	7-Zip file archiver with high compression ratio (LZMA compression)
p7zip-full	V:17, I:199	12	.7z	7z(1), 7za(1)	7-Zip file archiver with high compression ratio (LZMA compression and others)
lzop	V:14, I:129	164	.lzo	lzop(1)	LZO compression utility with higher compression and decompression speed than gzip(1) (lower compression ratio than gzip with similar syntax)
zip	V:48, I:355	635	.zip	zip(1)	InfoZIP: DOS archive and compression tool
unzip	V:123, I:744	399	.zip	unzip(1)	InfoZIP: DOS unarchive and decompression tool

Table 10.1: List of archive and compression tools

package	popcon	size	tool	function
coreutils	V:912, I:1000	17994	GNU cp	locally copy files and directories (”-a” for recursive)
openssh-client	V:682, I:997	3611	scp	remotely copy files and directories (client, ”-r” for recursive)
openssh-server	V:774, I:820	3615	sshd	remotely copy files and directories (remote server)
rsync	V:180, I:539	967		1-way remote synchronization and backup
unison	V:3, I:12	14		2-way remote synchronization and backup

Table 10.2: List of copy and synchronization tools

10.1.4 Idioms for the copy

Here are several ways to copy the entire content of the directory ”./source” using different tools.

- Local copy: ”./source” directory → ”/dest” directory
- Remote copy: ”./source” directory at local host → ”/dest” directory at ”user@host.dom” host

rsync(8):

```
# cd ./source; rsync -aHAXSv . /dest
# cd ./source; rsync -aHAXSv . user@host.dom:/dest
```

You can alternatively use ”a trailing slash on the source directory” syntax.

```
# rsync -aHAXSv ./source/ /dest
# rsync -aHAXSv ./source/ user@host.dom:/dest
```

Alternatively, by the following.

```
# cd ./source; find . -print0 | rsync -aHAXSv0 --files-from=- . /dest
# cd ./source; find . -print0 | rsync -aHAXSv0 --files-from=- . user@host.dom:/dest
```

GNU cp(1) and openSSH scp(1):

```
# cd ./source; cp -a . /dest
# cd ./source; scp -pr . user@host.dom:/dest
```

GNU tar(1):

```
# (cd ./source && tar cf - . ) | (cd /dest && tar xvfp - )
# (cd ./source && tar cf - . ) | ssh user@host.dom '(cd /dest && tar xvfp - )'
```

cpio(1):

```
# cd ./source; find . -print0 | cpio -pvdm --null --sparse /dest
```

You can substitute ”.” with ”foo” for all examples containing ”.” to copy files from ”./source/foo” directory to ”/dest/foo” directory.

You can substitute ”.” with the absolute path ”/path/to/source/foo” for all examples containing ”.” to drop ”cd ./source;”. These copy files to different locations depending on tools used as follows.

- ”/dest/foo”: rsync(8), GNU cp(1), and scp(1)
- ”/dest/path/to/source/foo”: GNU tar(1), and cpio(1)

Tip
rsync(8) and GNU cp(1) have option ”-u” to skip files that are newer on the receiver.

10.1.5 Idioms for the selection of files

find(1) is used to select files for archive and copy commands (see Section 10.1.3 and Section 10.1.4) or for xargs(1) (see Section 9.4.9). This can be enhanced by using its command arguments.

Basic syntax of find(1) can be summarized as the following.

- Its conditional arguments are evaluated from left to right.
- This evaluation stops once its outcome is determined.
- "Logical **OR**" (specified by "`-o`" between conditionals) has lower precedence than "logical **AND**" (specified by "`-a`" or nothing between conditionals).
- "Logical **NOT**" (specified by "`!`" before a conditional) has higher precedence than "logical **AND**".
- "`-prune`" always returns logical **TRUE** and, if it is a directory, searching of file is stopped beyond this point.
- "`-name`" matches the base of the filename with shell glob (see Section 1.5.6) but it also matches its initial "`.`" with metacharacters such as "`*`" and "`?`". (New POSIX feature)
- "`-regex`" matches the full path with emacs style **BRE** (see Section 1.6.2) as default.
- "`-size`" matches the file based on the file size (value precedented with "+" for larger, precedented with "`-`" for smaller)
- "`-newer`" matches the file newer than the one specified in its argument.
- "`-print0`" always returns logical **TRUE** and print the full filename (null terminated) on the standard output.

find(1) is often used with an idiomatic style as the following.

```
# find /path/to \
    -xdev -regextype posix-extended \
    -type f -regex ".*\.cpio|.*~" -prune -o \
    -type d -regex ".*/\.git" -prune -o \
    -type f -size +99M -prune -o \
    -type f -newer /path/to/timestamp -print0
```

This means to do following actions.

1. Search all files starting from "`/path/to`"
2. Globally limit its search within its starting filesystem and uses **ERE** (see Section 1.6.2) instead
3. Exclude files matching regex of "`.*\.cpio`" or "`.*~`" from search by stop processing
4. Exclude directories matching regex of "`.*/\.git`" from search by stop processing
5. Exclude files larger than 99 Megabytes (units of 1048576 bytes) from search by stop processing
6. Print filenames which satisfy above search conditions and are newer than "`/path/to/timestamp`"

Please note the idiomatic use of "`-prune -o`" to exclude files in the above example.

Note
For non-Debian Unix-like system, some options may not be supported by find(1). In such a case, please consider to adjust matching methods and replace "`-print0`" with "`-print`". You may need to adjust related commands too.

10.1.6 Archive media

When choosing computer data storage media for important data archive, you should be careful about their limitations. For small personal data backup, I use CD-R and DVD-R by the brand name company and store in a cool, shaded, dry, clean environment. (Tape archive media seem to be popular for professional use.)

Note
A fire-resistant safe are meant for paper documents. Most of the computer data storage media have less temperature tolerance than paper. I usually rely on multiple secure encrypted copies stored in multiple secure locations.

Optimistic storage life of archive media seen on the net (mostly from vendor info).

- 100+ years : Acid free paper with ink
- 100 years : Optical storage (CD/DVD, CD/DVD-R)
- 30 years : Magnetic storage (tape, floppy)
- 20 years : Phase change optical storage (CD-RW)

These do not count on the mechanical failures due to handling etc.

Optimistic write cycle of archive media seen on the net (mostly from vendor info).

- 250,000+ cycles : Harddisk drive
- 10,000+ cycles : Flash memory
- 1,000 cycles : CD/DVD-RW
- 1 cycles : CD/DVD-R, paper



Caution
Figures of storage life and write cycle here should not be used for decisions on any critical data storage. Please consult the specific product information provided by the manufacture.

Tip
Since CD/DVD-R and paper have only 1 write cycle, they inherently prevent accidental data loss by overwriting. This is advantage!

Tip
If you need fast and frequent backup of large amount of data, a hard disk on a remote host linked by a fast network connection, may be the only realistic option.

Tip
If you use re-writable media for your backups, use of filesystem such as btrfs or zfs which supports read-only snapshots may be a good idea.

10.1.7 Removable storage device

Removable storage devices may be any one of the following.

- USB flash drive
- Hard disk drive
- Optical disc drive
- Digital camera
- Digital music player

They may be connected via any one of the following.

- USB
- IEEE 1394 / FireWire

Modern desktop environments such as GNOME and KDE can mount these removable devices automatically without a matching "`/etc/fstab`" entry.

- `udisks2` package provides a daemon and associated utilities to mount and unmount these devices.
- D-bus creates events to initiate automatic processes.
- PolicyKit provides required privileges.

Tip
Automounted devices may have the "`uhelper=`" mount option which is used by umount(8).

Tip
Automounting under modern desktop environment happens only when those removable media devices are not listed in "`/etc/fstab`".

Mount point under modern desktop environment is chosen as "`/media/`*`username/disk_label`*" which can be customized by the following.

- mlabel(1) for FAT filesystem
- genisoimage(1) with "`-V`" option for ISO9660 filesystem
- tune2fs(1) with "`-L`" option for ext2/ext3/ext4 filesystem

---

Tip
The choice of encoding may need to be provided as mount option (see Section 8.1.3).

Tip
The use of the GUI menu to unmount a filesystem may remove its dynamically generated device node such as "`/dev/sdc`". If you wish to keep its device node, unmount it with the umount(8) command from the shell prompt.

10.1.8 Filesystem choice for sharing data

When sharing data with other system via removable storage device, you should format it with common filesystem supported by both systems. Here is a list of filesystem choices.

Note
Statements on hard disk (HDD) are applicable to other storage devices such as SSD / USB flash drive / Memory card / ... . Replace device names in examples such as `/dev/sda` with applicable device names `/dev/nvme0`, `/dev/mmcblk0`, ...

filesystem name	typical usage scenario
FAT12	cross platform sharing of data on the floppy disk (<32MiB)
FAT16	cross platform sharing of data on the small hard disk like device (<2GiB)
FAT32	cross platform sharing of data on the large hard disk like device (<8TiB, supported by newer than MS Windows95 OSR2)
exFAT	cross platform sharing of data on the large hard disk like device (<512TiB, supported by WindowsXP, Mac OS X Snow Leopard 10.6.5, and Linux kernel since 5.4 release)
NTFS	cross platform sharing of data on the large hard disk like device (supported natively on MS Windows NT and later version, and supported by NTFS-3G via FUSE on Linux)
ISO9660	cross platform sharing of static data on CD-R and DVD+/-R
UDF	incremental data writing on CD-R and DVD+/-R (new)
MINIX	space efficient unix file data storage on the floppy disk
ext2	sharing of data on the hard disk like device with older Linux systems
ext3	sharing of data on the hard disk like device with older Linux systems
ext4	sharing of data on the hard disk like device with current Linux systems
btrfs	sharing of data on the hard disk like device with current Linux systems with read-only snapshots

Table 10.3: List of filesystem choices for removable storage devices with typical usage scenarios

Tip
See Section 9.9.1 for cross platform sharing of data using device level encryption.

The FAT filesystem is supported by almost all modern operating systems and is quite useful for the data exchange purpose via removable hard disk like media.

When formatting removable hard disk like devices for cross platform sharing of data with the FAT filesystem, the following should be safe choices.

- Partitioning them with fdisk(8), cfdisk(8) or parted(8) (see Section 9.6.2) into a single primary partition and to mark it as the following.
  - Type ”6” for FAT16 for media smaller than 2GB.
  - Type ”c” for FAT32 (LBA) for larger media.
- Formatting the primary partition with mkfs.vfat(8) with the following.
  - Just its device name, e.g. ”`/dev/sda1`” for FAT16
  - The explicit option and its device name, e.g. ”`-F 32 /dev/sda1`” for FAT32

When using the FAT or ISO9660 filesystems for sharing data, the following should be the safe considerations.

- Archiving files into an archive file first using tar(1), or cpio(1) to retain the long filename, the symbolic link, the original Unix file permission and the owner information.
- Splitting the archive file into less than 2 GiB chunks with the split(1) command to protect it from the file size limitation.
- Encrypting the archive file to secure its contents from the unauthorized access.

Note
For FAT filesystems by its design, the maximum file size is `(2^32 - 1) bytes = (4GiB - 1 byte)`. For some applications on the older 32 bit OS, the maximum file size was even smaller `(2^31 - 1) bytes = (2GiB - 1 byte)`. Debian does not suffer the latter problem.

Note
Microsoft itself does not recommend to use FAT for drives or partitions of over 200 MB. Microsoft highlights its short comings such as inefficient disk space usage in their "Overview of FAT, HPFS, and NTFS File Systems". Of course, we should normally use the ext4 filesystem for Linux.

Tip
For more on filesystems and accessing filesystems, please read "Filesystems HOWTO".

10.1.9 Sharing data via network

When sharing data with other system via network, you should use common service. Here are some hints.

network service	description of typical usage scenario
SMB/CIFS network mounted filesystem with Samba	sharing files via "Microsoft Windows Network", see smb.conf(5) and The Official Samba 3.x.x HOWTO and Reference Guide or the `samba-doc` package
NFS network mounted filesystem with the Linux kernel	sharing files via "Unix/Linux Network", see exports(5) and Linux NFS-HOWTO
HTTP service	sharing file between the web server/client
HTTPS service	sharing file between the web server/client with encrypted Secure Sockets Layer (SSL) or Transport Layer Security (TLS)
FTP service	sharing file between the FTP server/client

Table 10.4: List of the network service to choose with the typical usage scenario

Although these filesystems mounted over network and file transfer methods over network are quite convenient for sharing data, these may be insecure. Their network connection must be secured by the following.

- Encrypt it with SSL/TLS
- Tunnel it via SSH
- Tunnel it via VPN
- Limit it behind the secure firewall

See also Section 6.5 and Section 6.6.

10.2 Backup and recovery

We all know that computers fail sometime or human errors cause system and data damages. Backup and recovery operations are the essential part of successful system administration. All possible failure modes hit you some day.

Tip
Keep your backup system simple and backup your system often. Having backup data is more important than how technically good your backup method is.

10.2.1 Backup and recovery policy

There are 3 key factors which determine actual backup and recovery policy.

1. Knowing what to backup and recover.
   - Data files directly created by you: data in "`~/`"
   - Data files created by applications used by you: data in "`/var/`" (except "`/var/cache/`", "`/var/run/`", and "`/var/tmp/`")
   - System configuration files: data in "`/etc/`"
   - Local programs: data in "`/usr/local/`" or "`/opt/`"
   - System installation information: a memo in plain text on key steps (partition, …)
   - Proven set of data: confirmed by experimental recovery operations in advance
     - Cron job as a user process: files in "`/var/spool/cron/crontabs`" directory and restart cron(8). See Section 9.4.14 for cron(8) and crontab(1).
     - Systemd timer jobs as user processes: files in "`~/.config/systemd/user`" directory. See systemd.timer(5) and systemd.service(5).
     - Autostart jobs as user processes: files in "`~/.config/autostart`" directory. See Desktop Application Autostart Specification.
2. Knowing how to backup and recover.
   - Secure storage of data: protection from overwrite and system failure
   - Frequent backup: scheduled backup
   - Redundant backup: data mirroring
   - Fool proof process: easy single command backup
3. Assessing risks and costs involved.
   - Risk of data when lost
     - Data should be at least on different disk partitions preferably on different disks and machines to withstand the filesystem corruption. Important data are best stored on a read-only filesystem. [1]
   - Risk of data when breached
     - Sensitive identity data such as "`/etc/ssh/ssh_host_*_key`", "`~/.gnupg/*`", "`~/.ssh/*`", "`~/.local/share`", "`/etc/passwd`", "`/etc/shadow`", "`popularity-contest.conf`", "`/etc/ppp/pap-secrets`", and "`/etc/ex`" should be backed up as encrypted. [2] (See Section 9.9.)
     - Never hard code system login password nor decryption passphrase in any script even on any trusted system. (See Section 10.3.6.)

1A write-once media such as CD/DVD-R can prevent overwrite accidents. (See Section 9.8 for how to write to the storage media from the shell commandline. GNOME desktop GUI environment gives you easy access via menu: "Places→CD/DVD Creator".)

2Some of these data can not be regenerated by entering the same input string to the system.

- Failure mode and their possibility
  - Hardware (especially HDD) will break
  - Filesystem may be corrupted and data in it may be lost
  - Remote storage system can't be trusted for security breaches
  - Weak password protection can be easily compromised
  - File permission system may be compromised
- Required resources for backup: human, hardware, software, …
  - Automatic scheduled backup with cron job or systemd timer job

Tip
You can recover debconf configuration data with "`debconf-set-selections debconf-selections`" and dpkg selection data with "`dpkg --set-selection <dpkg-selections.list`".

Note
Do not back up the pseudo-filesystem contents found on `/proc`, `/sys`, `/tmp`, and `/run` (see Section 1.2.12 and Section 1.2.13). Unless you know exactly what you are doing, they are huge useless data.

Note
You may wish to stop some application daemons such as MTA (see Section 6.2.4) while backing up data.

10.2.2 Backup utility suites

Here is a select list of notable backup utility suites available on the Debian system.

Backup tools have their specialized focuses.

- Mondo Rescue is a backup system to facilitate restoration of complete system quickly from backup CD/DVD etc. without going through normal system installation processes.
- Bacula, Amanda, and BackupPC are full featured backup suite utilities which are focused on regular backups over network.
- Duplicity, and Borg are simpler backup utilities for typical workstations.

10.2.3 Backup tips

For a personal workstation, full featured backup suite utilities designed for the server environment may not serve well. At the same time, existing backup utilities for workstations may have some shortcomings.

Here are some tips to make backup easier with minimal user efforts. These techniques may be used with any backup utilities.

For demonstration purpose, let's assume the primary user and group name to be `penguin` and create a backup and snapshot script example "`/usr/local/bin/bkss.sh`" as:

```
#!/bin/sh -e
SRC="$1" # source data path
DSTFS="$2" # backup destination filesystem path
DSTSV="$3" # backup destination subvolume name
DSTSS="${DSTFS}/${DSTSV}-snapshot" # snapshot destination path
if [ "$(stat -f -c %T "$DSTFS")" != "btrfs" ]; then
  echo "E: $DESTFS needs to be formatted to btrfs" >&2 ; exit 1
```

package	popcon	size	description
bacula-common	V:6.0, I:6.7	2501	Bacula: network backup, recovery and verification - common support files
bacula-client	V:0.2, I:1.8	199	Bacula: network backup, recovery and verification - client meta-package
bacula-console	V:0.6, I:2.1	112	Bacula: network backup, recovery and verification - text console
bacula-server	I:0.59	199	Bacula: network backup, recovery and verification - server meta-package
amanda-common	V:0.6, I:2.1	9951	Amanda: Advanced Maryland Automatic Network Disk Archiver (Libs)
amanda-client	V:0.5, I:2.1	1109	Amanda: Advanced Maryland Automatic Network Disk Archiver (Client)
amanda-server	V:0.11, I:0.22	1095	Amanda: Advanced Maryland Automatic Network Disk Archiver (Server)
backuppc	V:1.4, I:1.6	3088	BackupPC is a high-performance, enterprise-grade system for backing up PCs (disk based)
duplicity	V:5, I:44	2662	(remote) incremental backup
deja-dup	V:28, I:41	5274	GUI frontend for duplicity
borgbackup	V:13, I:28	3532	(remote) deduplicating backup
borgmatic	V:3.2, I:4.4	946	borgbackup helper
rdiff-backup	V:2.1, I:6.1	1169	(remote) incremental backup
restic	V:8, I:15	41617	(remote) incremental backup
backupninja	V:2.0, I:2.4	360	lightweight, extensible **meta-backup** system
slbackup	V:0.10, I:0.13	147	(remote) incremental backup
backup-manager	V:0.41, I:0.70	573	command-line backup tool
backup2l	V:0.35, I:0.46	110	low-maintenance backup/restore tool for mountable media (disk based)

Table 10.5: List of backup suite utilities

```
fi
MSGID=$(notify-send -p "bkup.sh $DSTSV" "in progress ...")
if [ ! -d "$DSTFS/$DSTSV" ]; then
  btrfs subvolume create "$DSTFS/$DSTSV"
  mkdir -p "$DSTSS"
fi
rsync -aHxS --delete --mkpath "${SRC}/" "${DSTFS}/${DSTSV}"
btrfs subvolume snapshot -r "${DSTFS}/${DSTSV}" ${DSTSS}/$(date -u --iso=min)
notify-send -r "$MSGID" "bkup.sh $DSTSV" "finished!"
```

Here, only the basic tool rsync(1) is used to facilitate system backup and the storage space is efficiently used by Btrfs.

Tip
FYI: This author uses his own similar shell script "bss: Btrfs Subvolume Snapshot Utility" for his workstation.

10.2.3.1 GUI backup

Here is an example to setup the single GUI click backup.

- Prepare a USB storage device to be used for backup.
  - Format a USB storage device with one partition in btrfs with its label name as "BKUP". This can be encrypted (see Section 9.9.1).
  - Plug this in to your system. The desktop system should automatically mount it as "`/media/penguin/BKUP`".
  - Execute "`sudo chown penguin:penguin /media/penguin/BKUP`" to make it writable by the user.
- Create "`~/.local/share/applications/BKUP.desktop`" following techniques written in Section 9.4.10 as:

```
[Desktop Entry]
Name=bkss
Comment=Backup and snapshot of ~/Documents
Exec=/usr/local/bin/bkss.sh /home/penguin/Documents /media/penguin/BKUP Documents
Type=Application
```

For each GUI click, your data is backed up from "`~/Documents`" to a USB storage device and a read-only snapshot is created.

10.2.3.2 Mount event triggered backup

Here is an example to setup for the automatic backup triggered by the mount event.

- Prepare a USB storage device to be used for backup as in Section 10.2.3.1.
- Create a systemd service unit file "`~/.config/systemd/user/back-BKUP.service`" as:

```
[Unit]
Description=USB Disk backup
Requires=media-%u-BKUP.mount
After=media-%u-BKUP.mount

[Service]
ExecStart=/usr/local/bin/bkss.sh %h/Documents /media/%u/BKUP Documents
StandardOutput=append:%h/.cache/systemd-snap.log
StandardError=append:%h/.cache/systemd-snap.log

[Install]
WantedBy=media-%u-BKUP.mount
```

- Enable this systemd unit configuration with the following:

```
$ systemctl --user enable bkup-BKUP.service
```

For each mount event, your data is backed up from "`~/Documents`" to a USB storage device and a read-only snapshot is created.

Here, names of systemd mount units that systemd currently has in memory can be asked to the service manager of the calling user with "`systemctl --user list-units --type=mount`".

10.2.3.3 Timer event triggered backup

Here is an example to setup for the automatic backup triggered by the timer event.

- Prepare a USB storage device to be used for backup as in Section 10.2.3.1.
- Create a systemd timer unit file "`~/.config/systemd/user/snap-Documents.timer`" as:

```
[Unit]
Description=Run btrfs subvolume snapshot on timer
Documentation=man:btrfs(1)

[Timer]
OnStartupSec=30
OnUnitInactiveSec=900

[Install]
WantedBy=timers.target
```

- Create a systemd service unit file "`~/.config/systemd/user/snap-Documents.service`" as:

```
[Unit]
Description=Run btrfs subvolume snapshot
Documentation=man:btrfs(1)

[Service]
Type=oneshot
Nice=15
ExecStart=/usr/local/bin/bkss.sh %h/Documents /media/%u/BKUP Documents
IOSchedulingClass=idle
CPUSchedulingPolicy=idle
StandardOutput=append:%h/.cache/systemd-snap.log
StandardError=append:%h/.cache/systemd-snap.log
```

- Enable this systemd unit configuration with the following:

```
$ systemctl --user enable snap-Documents.timer
```

For each timer event, your data is backed up from "`~/Documents`" to a USB storage device and a read-only snapshot is created.

Here, names of systemd timer user units that systemd currently has in memory can be asked to the service manager of the calling user with "`systemctl --user list-units --type=timer`".

For the modern desktop system, this systemd approach can offer more fine grained control than the traditional Unix ones using at(1), cron(8), or anacron(8).

package	popcon	size	command	description
gnupg	V:418, I:860	465	gpg(1)	GNU Privacy Guard - OpenPGP encryption and signing tool
gpgv	V:203, I:923	577	gpgv(1)	GNU Privacy Guard - OpenPGP signature verification tool
sq	V:1, I:33	24447	sq(1)	Sequoia-PGP - OpenPGP encryption and signing tool
sqv	V:479, I:554	1919	sqv(1)	Sequoia-PGP - OpenPGP signature verification tool
paperkey	V:1, I:12	58	paperkey(1)	extract just the secret information out of OpenPGP secret keys
cryptsetup	V:34, I:77	469	cryptsetup(8), …	utilities for dm-crypt block device encryption supporting LUKS
coreutils	V:912, I:1000	17994	md5sum(1)	compute and check MD5 message digest
coreutils	V:912, I:1000	17994	sha1sum(1)	compute and check SHA1 message digest
openssl	V:846, I:996	2564	openssl(1ssl)	compute message digest with "openssl dgst" (OpenSSL)
libsecret-tools	V:2, I:13	44	secret-tool(1)	store and retrieve passwords (CLI)
seahorse	V:76, I:260	7999	seahorse(1)	key management tool (GNOME)

Table 10.6: List of data security infrastructure tools

10.3 Data security infrastructure

The data security infrastructure is provided by the combination of data encryption tool, message digest tool, and signature tool.

See Section 9.9 on dm-crypt and fscrypt which implement automatic data encryption infrastructure via Linux kernel modules.

10.3.1 Key management for GnuPG

Here are GNU Privacy Guard commands for the basic key management.

Here is the meaning of the trust code.

The following generates my key "9563FC29932C409F1A77F9C1AB8A1522D8234C6A".

```
$ gpg --gen-key
gpg (GnuPG) 2.4.7; Copyright (C) 2024 g10 Code GmbH
...
GnuPG needs to construct a user ID to identify your key.

Real name: Osamu Aoki
Email address: osamu@debian.org
You selected this USER-ID:
    "Osamu Aoki <osamu@debian.org>"

Change (N)ame, (E)mail, or (O)kay/(Q)uit? o
...
public and secret key created and signed.

pub   ed25519 2026-02-14 [SC] [expires: 2029-02-13]
      9563FC29932C409F1A77F9C1AB8A1522D8234C6A
```

command	description
`gpg --gen-key`	generate a new key
`gpg --gen-revoke my_user_ID`	generate revoke key for my_user_ID
`gpg --edit-key user_ID`	edit key interactively, ”help” for help
`gpg -o file --export`	export all public keys to file
`gpg -o file --export-secret-keys`	export all private key to file
`gpg --import file`	import all keys from file
`gpg --send-keys user_ID`	send key of user_ID to keyserver
`gpg --recv-keys user_ID`	recv. key of user_ID from keyserver
`gpg --list-keys user_ID`	list keys of user_ID
`gpg --list-sigs user_ID`	list sig. of user_ID
`gpg --check-sigs user_ID`	check sig. of user_ID
`gpg --fingerprint user_ID`	check fingerprint of user_ID
`gpg --refresh-keys`	update local keyring

Table 10.7: List of GNU Privacy Guard commands for the key management

code	description of trust
`-`	no owner trust assigned / not yet calculated
`e`	trust calculation failed
`q`	not enough information for calculation
`n`	never trust this key
`m`	marginally trusted
`f`	fully trusted
`u`	ultimately trusted

Table 10.8: List of the meaning of the trust code

```
uid                      Osamu Aoki <osamu@debian.org>
sub   cv25519 2026-02-14 [E] [expires: 2029-02-13]
```

The following uploads my key "9563FC29932C409F1A77F9C1AB8A1522D8234C6A" to the popular keyserver "hkp://key

```
$ gpg --keyserver hkp://keys.gnupg.net --send-keys 9563FC29932C409F1A77F9C1AB8A1522D8234C6A
```

A good default keyserver set up in "~/.gnupg/gpg.conf" (or old location "~/.gnupg/options") contains the following.

```
keyserver hkp://keys.gnupg.net
```

The following obtains unknown keys from the keyserver.

```
$ gpg --list-sigs --with-colons | grep '^sig.*\[User ID not found\]' |\
          cut -d ':' -f 5| sort | uniq | xargs gpg --recv-keys
```

There was a bug in OpenPGP Public Key Server (pre version 0.9.6) which corrupted key with more than 2 sub-keys. The newer gnupg (>1.2.1-2) package can handle these corrupted subkeys. See gpg(1) under "--repair-pks-subkey-bu option.

Use of SHA-1 for hash has been deprecated. If your old RSA based openPGP key uses SHA-1 for hash, fix it using FixKeyWithSha1.

Tip

The sq(1) and sqv(1) commands are an alternative set of openPGP commands. See sq user documentation and A Practical Introduction to using sq, Sequoia PGP's CLI.

10.3.2 Using GnuPG on files

Here are examples for using GNU Privacy Guard commands on files.

10.3.3 Using GnuPG with Mutt

Add the following to "~/.muttrc" to keep a slow GnuPG from automatically starting, while allowing it to be used by typing "S" at the index menu.

```
macro index S ":toggle pgp_verify_sig\n"
set pgp_verify_sig=no
```

10.3.4 Using GnuPG with Vim

The gnupg plugin let you run GnuPG transparently for files with extension ".gpg", ".asc", and ".pgp".[3]

```
$ sudo aptitude install vim-scripts
$ echo "packadd! gnupg" >> ~/.vim/vimrc
```

[3] If you use "~/.vimrc" instead of "~/.vim/vimrc", please substitute accordingly.

command	description
`gpg -a -s file`	sign file into ASCII armored file.asc
`gpg --armor --sign file`	,,
`gpg --clearsign file`	clear-sign message
`gpg --clearsign file|mail foo@example.org`	mail a clear-signed message to `foo@example.org`
`gpg --clearsign --not-dash-escaped patchfile`	clear-sign patchfile
`gpg --verify file`	verify clear-signed file
`gpg -o file.sig -b file`	create detached signature
`gpg -o file.sig --detach-sign file`	,,
`gpg --verify file.sig file`	verify file with file.sig
`gpg -o crypt_file.gpg -r name -e file`	public-key encryption intended for name from file to binary crypt_file.gpg
`gpg -o crypt_file.gpg --recipient name --encrypt file`	,,
`gpg -o crypt_file.asc -a -r name -e file`	public-key encryption intended for name from file to ASCII armored crypt_file.asc
`gpg -o crypt_file.gpg -c file`	symmetric encryption from file to crypt_file.gpg
`gpg -o crypt_file.gpg --symmetric file`	,,
`gpg -o crypt_file.asc -a -c file`	symmetric encryption intended for name from file to ASCII armored crypt_file.asc
`gpg -o file -d crypt_file.gpg -r name`	decryption
`gpg -o file --decrypt crypt_file.gpg`	,,

Table 10.9: List of GNU Privacy Guard commands on files

10.3.5 The MD5 sum

md5sum(1) provides utility to make a digest file using the method in rfc1321 and verifying each file with it.

```
$ md5sum foo bar >baz.md5
$ cat baz.md5
d3b07384d113edec49eaa6238ad5ff00  foo
c157a79031e1c40f85931829bc5fc552  bar
$ md5sum -c baz.md5
foo: OK
bar: OK
```

Note
The computation for the MD5 sum is less CPU intensive than the one for the cryptographic signature by GNU Privacy Guard (GnuPG). Usually, only the top level digest file is cryptographically signed to ensure data integrity.

10.3.6 Password keyring

On GNOME system, the GUI tool seahorse(1) manages passwords and stores them securely in the keyring `~/.local/shar`

secret-tool(1) can store password to the keyring from the command line.

Let's store passphrase used for LUKS/dm-crypt encrypted disk image

```
$ secret-tool store --label='LUKS passphrase for disk.img' LUKS my_disk.img
Password: ********
```

This stored password can be retrieved and fed to other programs, e.g., cryptsetup(8).

```
$ secret-tool lookup LUKS my_disk.img | \
  cryptsetup open disk.img disk_img --type luks --keyring -
$ sudo mount /dev/mapper/disk_img /mnt
```

Tip
Whenever you need to provide password in a script, use `secret-tool` and avoid directly hardcoding the passphrase in it.

10.4 Source code merge tools

There are many merge tools for the source code. Following commands caught my eyes.

10.4.1 Extracting differences for source files

The following procedures extract differences between two source files and create unified diff files "`file.patch0`" or "`file.patch1`" depending on the file location.

```
$ diff -u file.old file.new > file.patch0
$ diff -u old/file new/file > file.patch1
```

package	popcon	size	command	description
patch	V:102, I:727	242	patch(1)	apply a diff file to an original
vim	V:83, I:338	4164	vimdiff(1)	compare 2 files side by side in vim
imediff	V:0.02, I:0.25	329	imediff(1)	interactive full screen 2/3-way merge tool
meld	V:5, I:23	3996	meld(1)	compare and merge files (GTK)
wiggle	V:0.02, I:0.12	175	wiggle(1)	apply rejected patches
diffutils	V:896, I:998	1768	diff(1)	compare files line by line
diffutils	V:896, I:998	1768	diff3(1)	compare and merges three files line by line
quilt	V:2, I:16	880	quilt(1)	manage series of patches
wdiff	V:5, I:38	580	wdiff(1)	display word differences between text files
diffstat	V:9, I:93	79	diffstat(1)	produce a histogram of changes by the diff
patchutils	V:11, I:92	269	combinediff(1)	create a cumulative patch from two incremental patches
patchutils	V:11, I:92	269	dehtmldiff(1)	extract a diff from an HTML page
patchutils	V:11, I:92	269	filterdiff(1)	extract or excludes diffs from a diff file
patchutils	V:11, I:92	269	fixcvsdiff(1)	fix diff files created by CVS that patch(1) mis-interprets
patchutils	V:11, I:92	269	flipdiff(1)	exchange the order of two patches
patchutils	V:11, I:92	269	grepdiff(1)	show which files are modified by a patch matching a regex
patchutils	V:11, I:92	269	interdiff(1)	show differences between two unified diff files
patchutils	V:11, I:92	269	lsdiff(1)	show which files are modified by a patch
patchutils	V:11, I:92	269	recountdiff(1)	recompute counts and offsets in unified context diffs
patchutils	V:11, I:92	269	rediff(1)	fix offsets and counts of a hand-edited diff
patchutils	V:11, I:92	269	splitdiff(1)	separate out incremental patches
patchutils	V:11, I:92	269	unwrapdiff(1)	demangle patches that have been word-wrapped
dirdiff	V:0.1, I:1.3	167	dirdiff(1)	display differences and merge changes between directory trees
docdiff	V:0.01, I:0.16	554	docdiff(1)	compare two files word by word / char by char
makepatch	V:0.01, I:0.14	99	makepatch(1)	generate extended patch files
makepatch	V:0.01, I:0.14	99	applypatch(1)	apply extended patch files

Table 10.10: List of source code merge tools

10.4.2 Merging updates for source files

The diff file (alternatively called patch file) is used to send a program update. The receiving party applies this update to another file by the following.

```
$ patch -p0 file < file.patch0
$ patch -p1 file < file.patch1
```

10.4.3 Interactive merge

If you have two versions of a source code, you can perform 2-way merge interactively using imediff(1) by the following.

```
$ imediff -o file.merged file.old file.new
```

If you have three versions of a source code, you can perform 3-way merge interactively using imediff(1) by the following.

```
$ imediff -o file.merged file.yours file.base file.theirs
```

10.5 Git

Git is the tool of choice these days for the version control system (VCS) since Git can do everything for both local and remote source code management.

Debian provides free Git services via Debian Salsa service. Its documentation can be found at https://wiki.debian.org/Salsa .

Here are some Git related packages.

package	popcon	size	command	description
git	V:422, I:625	52512	git(7)	Git, the fast, scalable, distributed revision control system
gitk	V:3, I:26	2045	gitk(1)	GUI Git repository browser with history
qgit	V:0.2, I:1.9	1431	qgit(1)	GUI Git repository browser with history
git-cola	V:0.8, I:4.5	5129	git-cola(1)	GUI Git repository browser with history
tig	V:2, I:11	1245	tig(1)	console Git repository browser with history
lazygit	V:1.0, I:4.1	24167	lazygit(1)	console Git repository browser with history
git-gui	V:1, I:16	2549	git-gui(1)	GUI for Git (No history)
git-email	V:0.3, I:9.3	1229	git-send-email(1)	send a collection of patches as email from the Git
git-buildpackage	V:1.2, I:6.8	2027	git-buildpackage(1)	automate the Debian packaging with the Git
dgit	V:0.12, I:0.97	757	dgit(1)	git interoperability with the Debian archive
imediff	V:0.02, I:0.25	329	git-ime(1)	interactive git commit split helper tool
stgit	V:0.08, I:0.50	6253	stg(1)	quilt on top of git (Python)
git-doc	I:10	15192	N/A	official documentation for Git
gitmagic	I:0.48	721	N/A	”Git Magic”, easier to understand guide for Git

Table 10.11: List of git related packages and commands

10.5.1 Configuration of Git client

You may wish to set several global configuration in "`~/.gitconfig`" such as your name and email address used by Git by the following.

```
$ git config --global user.name "Name Surname"
$ git config --global user.email yourname@example.com
```

You may also customize the Git default behavior by the following.

```
$ git config --global init.defaultBranch main
$ git config --global pull.rebase true
$ git config --global push.default current
```

If you are too used to CVS or Subversion commands, you may wish to set several command aliases by the following.

```
$ git config --global alias.ci "commit -a"
$ git config --global alias.co checkout
```

You can check your global configuration by the following.

```
$ git config --global --list
```

10.5.2 Basic Git commands

Git operation involves several data.

- The working tree which holds user facing files and to which you make changes.
  - The changes to be recorded must be explicitly selected and staged to the index. This is `git add` and `git rm` commands.
- The index which holds staged files.
  - Staged files will be committed to the local repository upon the subsequent request. This is `git commit` command.
- The local repository which holds committed files.
  - Git records the linked history of the committed data and organizes them as branches in the repository.
  - The local repository can send data to the remote repository by `git push` command.
  - The local repository can receive data from the remote repository by `git fetch` and `git pull` commands.
    * The `git pull` command performs `git merge` or `git rebase` command after `git fetch` command.
    * Here, `git merge` combines two separate branches of history at the end to a point. (This is default of `git pull` without customization and may be good for upstream people who publish branch to many people.)
    * Here, `git rebase` creates one single branch of sequential history of the remote branch one followed by the local branch one. (This is `pull.rebase true` customization case and may be good for rest of us.)
- The remote repository which holds committed files.
  - The communication to the remote repository uses secure communication protocols such as SSH or HTTPS.

The working tree is files outside of the `.git/` directory. Files inside of the `.git/` directory hold the index, the local repository data, and some git configuration text files.

Here is an overview of main Git commands.

Git command	function
`git init`	create the (local) repository
`git clone URL`	clone the remote repository to a local repository with the working tree
`git pull origin main`	update the local `main` branch by the remote repository `origin`
`git add .`	add file(s) in the working tree to the index for pre-existing files in index only
`git add -A .`	add file(s) in the working tree to the index for all files including removals
`git rm filename`	remove file(s) from the working tree and the index
`git commit`	commit staged changes in the index to the local repository
`git commit -a`	add all changes in the working tree to the index and commit them to the local repository (add + commit)
`git push -u origin branch_name`	update the remote repository `origin` by the local `branch_name` branch (initial invocation)
`git push origin branch_name`	update the remote repository `origin` by the local `branch_name` branch (subsequent invocation)
`git diff treeish1 treeish2`	show difference between *treeish1* commit and *treeish2* commit
`gitk`	GUI display of VCS repository branch history tree

Table 10.12: Main Git commands

10.5.3 Git tips

Here are some Git tips.



Warning
Do not use the tag string with spaces in it even if some tools such as gitk(1) allow you to use it. It may choke some other `git` commands.



Caution
If a local branch which has been pushed to remote repository is rebased or squashed, pushing this branch has risks and requires `--force` option. This is usually not an acceptable for `main` branch but may be acceptable for a topic branch before merging to `main` branch.



Caution
Invoking a `git` subcommand directly as "`git-xyz`" from the command line has been deprecated since early 2006.

Tip
If there is a executable file `git-foo` in the path specified by `$PATH`, entering "`git foo`" without hyphen to the command line invokes this `git-foo`. This is a feature of the `git` command.

10.5.4 Git references

See the following.

Git command line	function
`gitk --all`	see complete Git history and operate on them such as resetting HEAD to another commit, cheery-picking patches, creating tags and branches ...
`git stash`	get the clean working tree without losing data
`git remote -v`	check settings for remote
`git branch -vv`	check settings for branch
`git status`	show working tree status
`git config -l`	list git settings
`git reset --hard HEAD; git clean -x -d -f`	revert all working tree changes and clean them up completely
`git rm --cached filename`	revert staged index changed by `git add filename`
`git reflog`	get reference log (useful for recovering commits from the removed branch)
`git branch new_branch_name HEAD@{6}`	create a new branch from reflog information
`git remote add new_remote URL`	add a `new_remote` remote repository pointed by URL
`git remote rename origin upstream`	rename the remote repository name from `origin` to `upstream`
`git branch -u upstream/branch_name`	set the remote tracking to the remote repository `upstream` and its branch name `branch_name`.
`git remote set-url origin https://foo/bar.git`	change URL of `origin`
`git remote set-url --push upstream DISABLED`	disable push to `upstream` (Edit `.git/config` to re-enable)
`git remote update upstream`	fetch updates of all remote branches in the `upstream` repository
`git fetch upstream foo:upstream-foo`	create a local (possibly orphan) `upstream-foo` branch as a copy of `foo` branch in the `upstream` repository
`git checkout -b topic_branch ; git push -u topic_branch origin`	make a new `topic_branch` and push it to `origin`
`git branch -m oldname newname`	rename local branch name
`git push -d origin branch_to_be_removed`	remove remote branch (new method)
`git push origin :branch_to_be_removed`	remove remote branch (old method)
`git checkout --orphan unconnected`	create a new `unconnected` branch
`git rebase -i origin/main`	reorder/drop/squish commits from `origin/main` to clean branch history
`git reset HEAD^; git commit --amend`	squash last 2 commits into one
`git checkout topic_branch ; git merge --squash topic_branch`	squash entire `topic_branch` into a commit
`git fetch --unshallow --update-head-ok origin '+refs/heads/*:refs/heads/*'`	convert a shallow clone to the full clone of all branches
`git ime`	split the last commit into a series of file-by-file smaller commits etc. (`imediff` package required)
`git repack -a -d; git prune`	repack the local repository into single pack (this may limit chance of lost data recovery from erased branch etc.)

Table 10.13: Git tips

- manpage: git(1) (`/usr/share/doc/git-doc/git.html`)
- Git User's Manual (`/usr/share/doc/git-doc/user-manual.html`)
- A tutorial introduction to git (`/usr/share/doc/git-doc/gittutorial.html`)
- A tutorial introduction to git: part two (`/usr/share/doc/git-doc/gittutorial-2.html`)
- Everyday GIT With 20 Commands Or So (`/usr/share/doc/git-doc/giteveryday.html`)
- Git Magic (`/usr/share/doc/gitmagic/html/index.html`)

10.5.5 Other version control systems

The version control systems (VCS) is sometimes known as the revision control system (RCS), or the software configuration management (SCM).

Here is a summary of the notable other non-Git VCS on the Debian system.

package	popcon	size	tool	VCS type	comment
`mercurial`	V:3, I:25	1375	Mercurial	distributed	DVCS in Python and some C
`darcs`	V:0.2, I:3.1	38856	Darcs	distributed	DVCS with smart algebra of patches (slow)
`tla`	V:0.03, I:0.67	1022	GNU arch	distributed	DVCS mainly by Tom Lord (historic)
`bzr`	V:0.0, I:3.9	28	GNU Bazaar	distributed	DVCS influenced by `tla` written in Python (historic)
`subversion`	V:9, I:54	4856	Subversion	remote	"CVS done right", newer standard remote VCS (historic)
`cvs`	V:3, I:23	4835	CVS	remote	previous standard remote VCS (historic)
`tkcvs`	V:0.11, I:0.83	34	CVS, …	remote	GUI display of VCS (CVS, Subversion, RCS) repository tree
`rcs`	V:1.6, I:9.0	578	RCS	local	"Unix SCCS done right" (historic)
`cssc`	V:0.01, I:0.28	2044	CSSC	local	clone of the Unix SCCS (historic)

Table 10.14: List of other version control system tools

Chapter 11

Data conversion

Tools and tips for converting data formats on the Debian system are described.

Standard based tools are in very good shape but support for proprietary data formats are limited.

11.1 Text data conversion tools

Following packages for the text data conversion caught my eyes.

package	popcon	size	keyword	description
libc6	V:936, I:1000	5483	charset	text encoding converter between locales by iconv(1) (fundamental)
recode	V:2, I:12	528	charset+eol	text encoding converter between locales (versatile, more aliases and features)
konwert	V:2, I:42	137	charset	text encoding converter between locales (fancy)
nkf	V:0.4, I:8.6	360	charset	character set translator for Japanese
tcs	V:0.01, I:0.14	518	charset	character set translator
unaccent	V:0.03, I:0.27	35	charset	replace accented letters by their unaccented equivalent
tofrodos	V:1, I:12	50	eol	text format converter between DOS and Unix: fromdos(1) and todos(1)
macutils	V:0.04, I:0.39	319	eol	text format converter between Macintosh and Unix: frommac(1) and tomac(1)

Table 11.1: List of text data conversion tools

11.1.1 Converting a text file with iconv

Tip

iconv(1) is provided as a part of the libc6 package and it is always available on practically all Unix-like systems to convert the encoding of characters.

You can convert encodings of a text file with iconv(1) by the following.

```
$ iconv -f encoding1 -t encoding2 input.txt >output.txt
```

Encoding values are case insensitive and ignore "-" and "_" for matching. Supported encodings can be checked by the "`iconv -l`" command.

encoding value	usage
ASCII	American Standard Code for Information Interchange, 7 bit code w/o accented characters
UTF-8	current multilingual standard for all modern OSs
ISO-8859-1	old standard for western European languages, ASCII + accented characters
ISO-8859-2	old standard for eastern European languages, ASCII + accented characters
ISO-8859-15	old standard for western European languages, ISO-8859-1 with euro sign
CP850	code page 850, Microsoft DOS characters with graphics for western European languages, ISO-8859-1 variant
CP932	code page 932, Microsoft Windows style Shift-JIS variant for Japanese
CP936	code page 936, Microsoft Windows style GB2312, GBK or GB18030 variant for Simplified Chinese
CP949	code page 949, Microsoft Windows style EUC-KR or Unified Hangul Code variant for Korean
CP950	code page 950, Microsoft Windows style Big5 variant for Traditional Chinese
CP1251	code page 1251, Microsoft Windows style encoding for the Cyrillic alphabet
CP1252	code page 1252, Microsoft Windows style ISO-8859-15 variant for western European languages
KOI8-R	old Russian UNIX standard for the Cyrillic alphabet
ISO-2022-JP	standard encoding for Japanese email which uses only 7 bit codes
eucJP	old Japanese UNIX standard 8 bit code and completely different from Shift-JIS
Shift-JIS	JIS X 0208 Appendix 1 standard for Japanese (see CP932)

Table 11.2: List of encoding values and their usage

Note
Some encodings are only supported for the data conversion and are not used as locale values (Section 8.1).

For character sets which fit in single byte such as ASCII and ISO-8859 character sets, the character encoding means almost the same thing as the character set.

For character sets with many characters such as JIS X 0213 for Japanese or Universal Character Set (UCS, Unicode, ISO-10646-1) for practically all languages, there are many encoding schemes to fit them into the sequence of the byte data.

- EUC and ISO/IEC 2022 (also known as JIS X 0202) for Japanese
- UTF-8, UTF-16/UCS-2 and UTF-32/UCS-4 for Unicode

For these, there are clear differentiations between the character set and the character encoding.

The code page is used as the synonym to the character encoding tables for some vendor specific ones.

Note
Please note most encoding systems share the same code with ASCII for the 7 bit characters. But there are some exceptions. If you are converting old Japanese C programs and URLs data from the casually-called shift-JIS encoding format to UTF-8 format, use "`CP932`" as the encoding name instead of "`shift-JIS`" to get the expected results: `0x5C` → "\" and `0x7E` → "~". Otherwise, these are converted to wrong characters.

Tip
recode(1) may be used too and offers more than the combined functionality of iconv(1), fromdos(1), todos(1), frommac(1), and tomac(1). For more, see "`info recode`".

11.1.2 Checking file to be UTF-8 with iconv

You can check if a text file is encoded in UTF-8 with iconv(1) by the following.

```
$ iconv -f utf8 -t utf8 input.txt >/dev/null || echo "non-UTF-8 found"
```

Tip
Use "`--verbose`" option in the above example to find the first non-UTF-8 character.

11.1.3 Converting file names with iconv

Here is an example script to convert encoding of file names from ones created under older OS to modern UTF-8 ones in a single directory.

```
#!/bin/sh
ENCDN=iso-8859-1
for x in *;
 do
 mv "$x" "$(echo "$x" | iconv -f $ENCDN -t utf-8)"
done
```

The "`$ENCDN`" variable specifies the original encoding used for file names under older OS as in Table 11.2.

For more complicated case, please mount a filesystem (e.g. a partition on a disk drive) containing such file names with proper encoding as the mount(8) option (see Section 8.1.3) and copy its entire contents to another filesystem mounted as UTF-8 with "`cp -a`" command.

11.1.4 EOL conversion

The text file format, specifically the end-of-line (EOL) code, is dependent on the platform.

platform	EOL code	control	decimal	hexadecimal
Debian (unix)	LF	^J	10	0A
MSDOS and Windows	CR-LF	^M^J	13 10	0D 0A
Apple's Macintosh	CR	^M	13	0D

Table 11.3: List of EOL styles for different platforms

The EOL format conversion programs, fromdos(1), todos(1), frommac(1), and tomac(1), are quite handy. recode(1) is also useful.

Note
Some data on the Debian system, such as the wiki page data for the `python-moinmoin` package, use MSDOS style CR-LF as the EOL code. So the above rule is just a general rule.

Note
Most editors (eg. `vim`, `emacs`, `gedit`, ...) can handle files in MSDOS style EOL transparently.

Tip
The use of "`sed -e '/\r$/!s/$/\r/'`" instead of todos(1) is better when you want to unify the EOL style to the MSDOS style from the mixed MSDOS and Unix style. (e.g., after merging 2 MSDOS style files with diff3(1).) This is because `todos` adds CR to all lines.

11.1.5 TAB conversion

There are few popular specialized programs to convert the tab codes.

function	`bsdmainutils`	`coreutils`
expand tab to spaces	"`col -x`"	`expand`
unexpand tab from spaces	"`col -h`"	`unexpand`

Table 11.4: List of TAB conversion commands from `bsdmainutils` and `coreutils` packages

indent(1) from the `indent` package completely reformats whitespaces in the C program.

Editor programs such as `vim` and `emacs` can be used for TAB conversion, too. For example with `vim`, you can expand TAB with "`:set expandtab`" and "`:%retab`" command sequence. You can revert this with "`:set noexpandtab`" and "`:%retab!`" command sequence.

11.1.6 Editors with auto-conversion

Intelligent modern editors such as the `vim` program are quite smart and copes well with any encoding systems and any file formats. You should use these editors under the UTF-8 locale in the UTF-8 capable console for the best compatibility.

An old western European Unix text file, "`u-file.txt`", stored in the latin1 (iso-8859-1) encoding can be edited simply with `vim` by the following.

```
$ vim u-file.txt
```

This is possible since the auto detection mechanism of the file encoding in `vim` assumes the UTF-8 encoding first and, if it fails, assumes it to be latin1.

An old Polish Unix text file, "`pu-file.txt`", stored in the latin2 (iso-8859-2) encoding can be edited with `vim` by the following.

```
$ vim '+e ++enc=latin2 pu-file.txt'
```

An old Japanese unix text file, "`ju-file.txt`", stored in the eucJP encoding can be edited with `vim` by the following.

```
$ vim '+e ++enc=eucJP ju-file.txt'
```

An old Japanese MS-Windows text file, "`jw-file.txt`", stored in the so called shift-JIS encoding (more precisely: CP932) can be edited with `vim` by the following.

```
$ vim '+e ++enc=CP932 ++ff=dos jw-file.txt'
```

When a file is opened with "`++enc`" and "`++ff`" options, "`:w`" in the Vim command line stores it in the original format and overwrite the original file. You can also specify the saving format and the file name in the Vim command line, e.g., "`:w ++enc=utf8 new.txt`".

Please refer to the mbyte.txt "multi-byte text support" in `vim` on-line help and Table 11.2 for locale values used with "`++enc`".

The `emacs` family of programs can perform the equivalent functions.

11.1.7 Plain text extraction

The following reads a web page into a text file. This is very useful when copying configurations off the Web or applying basic Unix text tools such as grep(1) on the web page.

```
$ w3m -dump https://www.remote-site.com/help-info.html >textfile
```

Similarly, you can extract plain text data from other formats using the following.

package	popcon	size	keyword	function
`w3m`	V:10, I:129	2853	`html→text`	HTML to text converter with the "`w3m -dump`" command
`html2text`	V:3, I:66	298	`html→text`	advanced HTML to text converter (ISO 8859-1)
`lynx`	V:29, I:455	2031	`html→text`	HTML to text converter with the "`lynx -dump`" command
`elinks`	V:2, I:15	1791	`html→text`	HTML to text converter with the "`elinks -dump`" command
`links`	V:2, I:20	2321	`html→text`	HTML to text converter with the "`links -dump`" command
`links2`	V:0.8, I:9.4	5466	`html→text`	HTML to text converter with the "`links2 -dump`" command
`catdoc`	V:16, I:170	682	`MSWord→text,TeX`	convert MSWord files to plain text or TeX
`antiword`	V:0.9, I:6.3	596	`MSWord→text,ps`	convert MSWord files to plain text or ps
`unhtml`	V:0.04, I:0.49	40	`html→text`	remove the markup tags from an HTML file
`odt2txt`	V:1, I:19	60	`odt→text`	converter from OpenDocument Text to text

Table 11.5: List of tools to extract plain text data

11.1.8 Highlighting and formatting plain text data

You can highlight and format plain text data by the following.

11.2 XML data

The Extensible Markup Language (XML) is a markup language for documents containing structured information.

See introductory information at XML.COM.

- "What is XML?"

package	popcon	size	keyword	description
vim-runtime	V:17, I:355	38849	highlight	Vim MACRO to convert source code to HTML with ":source $VIMRUNTIME/syntax/html.vim"
cxref	V:0.01, I:0.21	1191	c→html	converter for the C program to latex and HTML (C language)
src2tex	V:0.02, I:0.16	1799	highlight	convert many source codes to TeX (C language)
source-highlight	V:0.4, I:2.9	2131	highlight	convert many source codes to HTML, XHTML, LaTeX, Texinfo, ANSI color escape sequences and DocBook files with highlight (C++)
highlight	V:0.4, I:2.8	1412	highlight	convert many source codes to HTML, XHTML, RTF, LaTeX, TeX or XSL-FO files with highlight (C++)
grc	V:0.9, I:5.7	208	text→color	generic colouriser for everything (Python)
pandoc	V:11, I:47	209559	text→any	general markup converter (Haskell)
python3-docutils	V:13, I:51	2013	text→any	ReStructured Text document formatter to XML (Python)
markdown	V:0.4, I:5.7	56	text→html	Markdown text document formatter to (X)HTML (Perl)
asciidoctor	V:0.5, I:4.6	101	text→any	AsciiDoc text document formatter to XML/HTML (Ruby)
python3-sphinx	V:7, I:27	3234	text→any	ReStructured Text based document publication system (Python)
hugo	V:0.8, I:5.0	66708	text→html	Markdown based static site publication system (Go)

Table 11.6: List of tools to highlight plain text data

- "What Is XSLT?"
- "What Is XSL-FO?"
- "What Is XLink?"

11.2.1 Basic hints for XML

XML text looks somewhat like HTML. It enables us to manage multiple formats of output for a document. One easy XML system is the `docbook-xsl` package, which is used here.

Each XML file starts with standard XML declaration as the following.

```
<?xml version="1.0" encoding="UTF-8"?>
```

The basic syntax for one XML element is marked up as the following.

```
<name attribute="value">content</name>
```

XML element with empty content is marked up in the following short form.

```
<name attribute="value" />
```

The "`attribute="value"`" in the above examples are optional.

The comment section in XML is marked up as the following.

```
<!-- comment -->
```

predefined entity	character to be converted into
`"`	" : quote
`'`	' : apostrophe
`<`	< : less-than
`>`	> : greater-than
`&`	& : ampersand

Table 11.7: List of predefined entities for XML

Other than adding markups, XML requires minor conversion to the content using predefined entities for following characters.



Caution
"<" or "&" can not be used in attributes or elements.

Note
When SGML style user defined entities, e.g. "`&some-tag;`", are used, the first definition wins over others. The entity definition is expressed in "`<!ENTITY some-tag "entity value">`".

Note
As long as the XML markup are done consistently with certain set of the tag name (either some data as content or attribute value), conversion to another XML is trivial task using Extensible Stylesheet Language Transformations (XSLT).

11.2.2 XML processing

There are many tools available to process XML files such as the Extensible Stylesheet Language (XSL).

Basically, once you create well formed XML file, you can convert it to any format using Extensible Stylesheet Language Transformations (XSLT).

The Extensible Stylesheet Language for Formatting Objects (XSL-FO) is supposed to be solution for formatting. The `fop` package is new to the Debian `main` archive due to its dependence to the Java programming language. So the LaTeX code is usually generated from XML using XSLT and the LaTeX system is used to create printable file such as DVI, PostScript, and PDF.

Since XML is subset of Standard Generalized Markup Language (SGML), it can be processed by the extensive tools available for SGML, such as Document Style Semantics and Specification Language (DSSSL).

Tip
GNOME's `yelp` is sometimes handy to read DocBook XML files directly since it renders decently on X.

11.2.3 The XML data extraction

You can extract HTML or XML data from other formats using followings.

package	popcon	size	keyword	description
docbook-xml	V:15, I:407	2126	xml	XML document type definition (DTD) for DocBook
docbook-xsl	V:15, I:144	14823	xml/xslt	XSL stylesheets for processing DocBook XML to various output formats with XSLT
xsltproc	V:15, I:71	83	xslt	XSLT command line processor (XML → XML, HTML, plain text, etc.)
xmlto	V:0.5, I:8.0	124	xml/xslt	XML-to-any converter with XSLT
fop	V:0.7, I:8.0	281	xml/xsl-fo	convert Docbook XML files to PDF
dblatex	V:0.9, I:5.4	4635	xml/xslt	convert Docbook files to DVI, PostScript, PDF documents with XSLT
dbtoepub	V:0.04, I:0.54	37	xml/xslt	DocBook XML to .epub converter

Table 11.8: List of XML tools

package	popcon	size	keyword	description
openjade	V:1, I:21	1066	dsssl	ISO/IEC 10179:1996 standard DSSSL processor (latest)
docbook-dsssl	V:0.5, I:7.4	2594	xml/dsssl	DSSSL stylesheets for processing DocBook XML to various output formats with DSSSL
docbook-utils	V:0.3, I:5.2	287	xml/dsssl	utilities for DocBook files including conversion to other formats (HTML, RTF, PS, man, PDF) with docbook2* commands with DSSSL

Table 11.9: List of DSSSL tools

package	popcon	size	keyword	description
man2html	V:0.1, I:1.2	142	manpage→html	converter from manpage to HTML (CGI support)
doclifter	V:0.00, I:0.05	487	troff→xml	converter from troff to DocBook XML
texi2html	V:0.2, I:2.8	1847	texi→html	converter from Texinfo to HTML
info2www	V:0.8, I:1.4	76	info→html	converter from GNU info to HTML (CGI support)
wv	V:0.2, I:2.3	733	MSWord→any	document converter from Microsoft Word to HTML, LaTeX, etc.
unrtf	V:0.3, I:2.8	159	rtf→html	document converter from RTF to HTML, etc
wp2x	V:0.00, I:0.08	200	WordPerfect→any	WordPerfect 5.0 and 5.1 files to TeX, LaTeX, troff, GML and HTML

Table 11.10: List of XML data extraction tools

11.2.4 The XML data lint

For non-XML HTML files, you can convert them to XHTML which is an instance of well formed XML. XHTML can be processed by XML tools.

Syntax of XML files and goodness of URLs found in them may be checked.

package	popcon	size	function	description
libxml2-utils	V:60, I:207	213	xml↔html↔xhtml	command line XML tool with xmllint(1) (syntax check, reformat, lint, …)
tidy	V:0.9, I:6.9	79	xml↔html↔xhtml	HTML syntax checker and reformatter
weblint-perl	V:0.05, I:0.90	32	lint	syntax and minimal style checker for HTML
linklint	V:0.04, I:0.44	343	link check	fast link checker and web site maintenance tool

Table 11.11: List of XML pretty print tools

Once proper XML is generated, you can use XSLT technology to extract data based on the mark-up context etc.

11.3 Type setting

The Unix troff program originally developed by AT&T can be used for simple typesetting. It is usually used to create manpages.

TeX created by Donald Knuth is a very powerful type setting tool and is the de facto standard. LaTeX originally written by Leslie Lamport enables a high-level access to the power of TeX.

package	popcon	size	keyword	description
texlive	V:1, I:27	55	(La)TeX	TeX system for typesetting, previewing and printing
lilypond	V:0.7, I:5.4	16924	(La)TeX	music typesetter
groff	V:2, I:23	16536	troff	GNU troff text-formatting system

Table 11.12: List of type setting tools

11.3.1 roff typesetting

Traditionally, roff is the main Unix text processing system. See roff(7), groff(7), groff(1), grotty(1), troff(1), groff_mdoc(7), groff_man(7), groff_ms(7), groff_me(7), groff_mm(7), and "`info groff`".

You can read or print a good tutorial and reference on "`-me`" macro in "`/usr/share/doc/groff/`" by installing the `groff` package.

Tip
"`groff -Tascii -me -`" produces plain text output with ANSI escape code. If you wish to get manpage like output with many "^H" and "_", use "`GROFF_NO_SGR=1 groff -Tascii -me -`" instead.

Tip
To remove "^H" and "_" from a text file generated by `groff`, filter it by "`col -b -x`".

11.3.2 TeX/LaTeX

The TeX Live software distribution offers a complete TeX system. The `texlive` metapackage provides a decent selection of the TeX Live packages which should suffice for the most common tasks.

There are many references available for TeX and LaTeX.

- The teTeX HOWTO: The Linux-teTeX Local Guide
- tex(1)
- latex(1)
- texdoc(1)
- texdoctk(1)
- "The TeXbook", by Donald E. Knuth, (Addison-Wesley)
- "LaTeX - A Document Preparation System", by Leslie Lamport, (Addison-Wesley)
- "The LaTeX Companion", by Goossens, Mittelbach, Samarin, (Addison-Wesley)

This is the most powerful typesetting environment. Many SGML processors use this as their back end text processor. Lyx provided by the `lyx` package and GNU TeXmacs provided by the `texmacs` package offer nice WYSIWYG editing environment for LaTeX while many use Emacs and Vim as the choice for the source editor.

There are many online resources available.

- The TEX Live Guide - TEX Live 2007 ("`/usr/share/doc/texlive-doc-base/english/texlive-en/live.html`") (`texlive-doc-base` package)
- A Simple Guide to Latex/Lyx
- Word Processing Using LaTeX

When documents become bigger, sometimes TeX may cause errors. You must increase pool size in "`/etc/texmf/texmf.c`" (or more appropriately edit "`/etc/texmf/texmf.d/95NonPath`" and run update-texmf(8)) to fix this.

Note
The TeX source of "The TeXbook" is available at www.ctan.org tex-archive site for texbook.tex. This file contains most of the required macros. I heard that you can process this document with tex(1) after commenting lines 7 to 10 and adding "`\input manmac \proofmodefalse`". It's strongly recommended to buy this book (and all other books from Donald E. Knuth) instead of using the online version but the source is a great example of TeX input!

11.3.3 Pretty print a manual page

You can print a manual page in PostScript nicely by one of the following commands.

```
$ man -Tps some_manpage | lpr
```

11.3.4 Creating a manual page

Although writing a manual page (manpage) in the plain troff format is possible, there are few helper packages to create it.

package	popcon	size	keyword	description
docbook-to-man	V:0.6, I:5.3	189	SGML → manpage	converter from DocBook SGML into roff man macros
help2man	V:0.5, I:6.1	542	text → manpage	automatic manpage generator from --help
info2man	V:0.02, I:0.18	134	info → manpage	converter from GNU info to POD or man pages
txt2man	V:0.04, I:0.61	112	text → manpage	convert flat ASCII text to man page format

Table 11.13: List of packages to help creating the manpage

11.4 Printable data

Printable data is expressed in the PostScript format on the Debian system. Common Unix Printing System (CUPS) uses Ghostscript as its rasterizer backend program for non-PostScript printers.

Printable data may also be expressed in the PDF format on the recent Debian system.

PDF files can displayed and its form entries may be filled using GUI viewer tools such as Evince and Okular (see Section 7.4); and modern browsers such as Chromium.

PDF files can be edited using some graphics tools such as LibreOffice, Scribus, and Inkscape (see Section 11.6).

Tip
You can read a PDF file with GIMP and convert it into PNG format using higher than 300 dpi resolution. This may be used as a background image for LibreOffice to produce a desirable altered printout with minimum efforts.

11.4.1 Ghostscript

The core of printable data manipulation is the Ghostscript PostScript (PS) interpreter which generates raster image.

package	popcon	size	description
ghostscript	V:165, I:559	177	The GPL Ghostscript PostScript/PDF interpreter
ghostscript-x	V:0, I:13	88	GPL Ghostscript PostScript/PDF interpreter - X display support
libpoppler162	V:14, I:22	5313	PDF rendering library forked from the xpdf PDF viewer
libpoppler-glib8t64	V:77, I:323	612	PDF rendering library (GLib-based shared library)
poppler-data	V:161, I:582	13086	CMaps for PDF rendering library (for CJK support: Adobe-*)

Table 11.14: List of Ghostscript PostScript interpreters

Tip
"gs -h" can display the configuration of Ghostscript.

11.4.2 Merge two PS or PDF files

You can merge two PostScript (PS) or Portable Document Format (PDF) files using gs(1) of Ghostscript.

```
$ gs -q -dNOPAUSE -dBATCH -sDEVICE=pswrite -sOutputFile=bla.ps -f foo1.ps foo2.ps
$ gs -q -dNOPAUSE -dBATCH -sDEVICE=pdfwrite -sOutputFile=bla.pdf -f foo1.pdf foo2.pdf
```

Note
The PDF, which is a widely used cross-platform printable data format, is essentially the compressed PS format with few additional features and extensions.

Tip
For command line, psmerge(1) and other commands from the `psutils` package are useful for manipulating PostScript documents. pdftk(1) from the `pdftk` package is useful for manipulating PDF documents, too.

11.4.3 Printable data utilities

The following packages for the printable data utilities caught my eyes.

package	popcon	size	keyword	description
`poppler-utils`	V:169, I:476	804	`pdf`→`ps`,`text`,...	PDF utilities: `pdftops`, `pdfinfo`, `pdfimages`, `pdftotext`, `pdffonts`
`psutils`	V:4, I:48	34	`ps`→`ps`	PostScript document conversion tools
`poster`	V:0.1, I:1.6	57	`ps`→`ps`	create large posters out of PostScript pages
`enscript`	V:1, I:10	2138	text→ps, html, rtf	convert ASCII text to PostScript, HTML, RTF or Pretty-Print
`a2ps`	V:0.7, I:6.7	4109	`text`→`ps`	'Anything to PostScript' converter and pretty-printer
`pdftk`	V:1, I:20	28	`pdf`→`pdf`	PDF document conversion tool: `pdftk`
`html2ps`	V:0.2, I:1.7	256	`html`→`ps`	converter from HTML to PostScript
`gnuhtml2latex`	V:0.03, I:0.57	26	`html`→`latex`	converter from html to latex
`latex2rtf`	V:0.1, I:1.9	495	`latex`→`rtf`	convert documents from LaTeX to RTF which can be read by MS Word
`ps2eps`	V:2, I:31	95	`ps`→`eps`	converter from PostScript to EPS (Encapsulated PostScript)
`e2ps`	V:0.01, I:0.11	104	`text`→`ps`	Text to PostScript converter with Japanese encoding support
`impose+`	V:0.1, I:1.6	118	`ps`→`ps`	PostScript utilities
`trueprint`	V:0.00, I:0.07	148	`text`→`ps`	pretty print many source codes (C, C++, Java, Pascal, Perl, Pike, Sh, and Verilog) to PostScript. (C language)
`pdf2svg`	V:0.2, I:2.9	33	`pdf`→`svg`	converter from PDF to Scalable vector graphics format
`pdftoipe`	V:0.01, I:0.43	69	`pdf`→`ipe`	converter from PDF to IPE's XML format

Table 11.15: List of printable data utilities

11.4.4 Printing with CUPS

Both lp(1) and lpr(1) commands offered by Common Unix Printing System (CUPS) provides options for customized printing the printable data.

You can print 3 copies of a file collated using one of the following commands.

```
$ lp -n 3 -o Collate=True filename
```

```
$ lpr -#3 -o Collate=True filename
```

You can further customize printer operation by using printer option such as "`-o number-up=2`", "`-o page-set=even`", "`-o page-set=odd`", "`-o scaling=200`", "`-o natural-scaling=200`", etc., documented at Command-Line Printing and Options.

11.5 The mail data conversion

The following packages for the mail data conversion caught my eyes.

package	popcon	size	keyword	description
`sharutils`	V:2, I:27	1436	mail	shar(1), unshar(1), uuencode(1), uudecode(1)
`mpack`	V:0.7, I:7.4	109	`MIME`	encoding and decoding of MIME messages: mpack(1) and munpack(1)
`tnef`	V:0.3, I:3.7	103	`ms-tnef`	unpacking MIME attachments of type "application/ms-tnef" which is a Microsoft only format
`uudeview`	V:0.2, I:1.7	105	`mail`	encoder and decoder for the following formats: uuencode, xxencode, BASE64, quoted printable, and BinHex

Table 11.16: List of packages to help mail data conversion

Tip
The Internet Message Access Protocol version 4 (IMAP4) server may be used to move mails out from proprietary mail systems if the mail client software can be configured to use IMAP4 server too.

11.5.1 Mail data basics

Mail (SMTP) data should be limited to series of 7 bit data. So binary data and 8 bit text data are encoded into 7 bit format with the Multipurpose Internet Mail Extensions (MIME) and the selection of the charset (see Table 11.2).

The standard mail storage format is mbox formatted according to RFC2822 (updated RFC822). See mbox(5) (provided by the `mutt` package).

For European languages, "`Content-Transfer-Encoding: quoted-printable`" with the ISO-8859-1 charset is usually used for mail since there are not much 8 bit characters. If European text is encoded in UTF-8, "`Content-Transfer-Encoding: quoted-printable`" is likely to be used since it is mostly 7 bit data.

For Japanese, traditionally "`Content-Type: text/plain; charset=ISO-2022-JP`" is usually used for mail to keep text in 7 bits. But older Microsoft systems may send mail data in Shift-JIS without proper declaration. If Japanese text is encoded in UTF-8, Base64 is likely to be used since it contains many 8 bit data. The situation of other Asian languages is similar.

Note
If your non-Unix mail data is accessible by a non-Debian client software which can talk to the IMAP4 server, you may be able to move them out by running your own IMAP4 server.

Note
If you use other mail storage formats, moving them to mbox format is the good first step. The versatile client program such as mutt(1) may be handy for this.

You can split mailbox contents to each message using procmail(1) and formail(1).

Each mail message can be unpacked using munpack(1) from the `mpack` package (or other specialized tools) to obtain the MIME encoded contents.

11.6 Graphic data tools

Although GUI programs such as gimp(1) are very powerful, command line tools such as imagemagick(1) are quite useful for automating image manipulation via scripts.

The de facto image file format of the digital camera is the Exchangeable Image File Format (EXIF) which is the JPEG image file format with additional metadata tags. It can hold information such as date, time, and camera settings.

The Lempel-Ziv-Welch (LZW) lossless data compression patent has been expired. Graphics Interchange Format (GIF) utilities which use the LZW compression method are now freely available on the Debian system.

Tip
Any digital camera or scanner with removable recording media works with Linux through USB storage readers since it follows the Design rule for Camera Filesystem and uses FAT filesystem. See Section 10.1.7.

11.6.1 Graphic data tools (metapackage)

The following metapackages are good starting points for searching graphics data tools using aptitude(8). "Packages overview for Debian PhotoTools Maintainers" can be another starting point.

package	popcon	size	keyword	description
`education-graphics`	I:0.34	26	`svg,jpeg,…`	metapackage for teaching graphics and pictural art.
`open-font-design-toolkit`	I:0.04	9	`ttf,ps,…`	metapackage for open font design

Table 11.17: List of graphics data tools (metapackage)

Tip
Search more image tools using regex "`~Gworks-with::image`" in aptitude(8) (see Section 2.2.6).

11.6.2 Graphic data tools (GUI)

The following packages for the GUI graphics data conversion, editing, and organization tools caught my eyes.

11.6.3 Graphic data tools (CLI)

The following packages for the CLI graphics data conversion, editing, and organization tools caught my eyes.

package	popcon	size	keyword	description
gimp	V:34, I:211	33315	image(bitmap)	GNU Image Manipulation Program
xsane	V:9, I:128	1512	image(bitmap)	GTK-based X11 frontend for SANE (Scanner Access Now Easy)
scribus	V:1, I:12	32942	ps/pdf/SVG/...	Scribus DTP editor
libreoffice-draw	V:89, I:418	11418	image(vector)	LibreOffice office suite - drawing
inkscape	V:12, I:76	113353	image(vector)	SVG (Scalable Vector Graphics) editor
dia	V:1, I:16	3846	image(vector)	diagram editor (Gtk)
xfig	V:0.5, I:8.5	7951	image(vector)	Facility for Interactive Generation of figures under X11
gocr	V:0.5, I:3.6	549	image→text	free OCR software
eog	V:23, I:129	10535	image(Exif)	Eye of GNOME graphics viewer program
gthumb	V:3, I:11	5162	image(Exif)	image viewer and browser (GNOME)
geeqie	V:3, I:11	3367	image(Exif)	image viewer using GTK
shotwell	V:14, I:246	6334	image(Exif)	digital photo organizer (GNOME)
gwenview	V:41, I:114	6057	image(Exif)	image viewer (KDE)
kamera	I:114	992	image(Exif)	digital camera support for KDE applications
digikam	V:1.5, I:8.3	333	image(Exif)	digital photo management application for KDE
darktable	V:4, I:11	36092	image(Exif)	virtual lighttable and darkroom for photographers
hugin	V:0.5, I:5.3	6524	image(Exif)	panorama photo stitcher
librecad	V:1, I:13	9164	DXF,...	2D CAD data editor
freecad	V:1, I:20	112	DXF,...	3D CAD data editor
blender	V:2, I:20	171893	blend,TIFF,JPG,...	3D content editor for animation etc
mm3d	V:0.02, I:0.22	4123	ms3d,obj,dxf,...	OpenGL based 3D model editor
fontforge	V:0.6, I:5.3	4736	ttf,ps,...	font editor for PS, TrueType and OpenType fonts
xgridfit	V:0.02, I:0.08	878	ttf	program for gridfitting and hinting TrueType fonts

Table 11.18: List of graphics data tools (GUI)

package	popcon	size	keyword	description
imagemagick	V:7, I:272	80	image(bitmap)	image manipulation programs
graphicsmagick	V:1.0, I:8.3	5945	image(bitmap)	image manipulation programs (fork of imagemagick)
netpbm	V:27, I:282	8435	image(bitmap)	graphics conversion tools
libheif-examples	V:0.3, I:3.6	528	heif → jpeg(bitmap)	convert High Efficiency Image File Format (HEIF) to JPEG, PNG, or Y4M formats with heif-convert(1) command
icoutils	V:3, I:33	221	png ↔ ico(bitmap)	convert MS Windows icons and cursors to and from PNG formats (favicon.ico)
pstoedit	V:2, I:37	1075	ps/pdf → image(vector)	PostScript and PDF files to editable vector graphics converter (SVG)
libwmf-bin	V:5, I:80	149	Windows/image(vector)	Windows metafile (vector graphics data) conversion tools
fig2sxd	V:0.01, I:0.16	158	fig → sxd(vector)	convert XFig files to OpenOffice.org Draw format
unpaper	V:2, I:16	417	image → image	post-processing tool for scanned pages for OCR
tesseract-ocr	V:8, I:34	2209	image → text	free OCR software based on the HP's commercial OCR engine
tesseract-ocr-eng	V:9, I:35	4032	image → text	OCR engine data: tesseract-ocr language files for English text
ocrad	V:0.3, I:2.2	608	image → text	free OCR software
exif	V:3, I:49	335	image(Exif)	command-line utility to show EXIF information in JPEG files
exiv2	V:1, I:18	436	image(Exif)	EXIF/IPTC metadata manipulation tool
exiftran	V:1, I:11	81	image(Exif)	transform digital camera jpeg images
exiftags	V:0.2, I:2.5	309	image(Exif)	utility to read Exif tags from a digital camera JPEG file
exifprobe	V:0.2, I:2.1	506	image(Exif)	read metadata from digital pictures
dcraw	V:0.7, I:7.2	428	image(Raw) → ppm	decode raw digital camera images
findimagedupes	V:0.11, I:0.96	76	image → fingerprint	find visually similar or duplicate images
ale	V:0.00, I:0.14	850	image → image	merge images to increase fidelity or create mosaics
imageindex	V:0.1, I:1.1	143	image(Exif) → html	generate static HTML galleries from images
outguess	V:0.1, I:1.0	230	jpeg,png	universal Steganographic tool
jpegoptim	V:0.6, I:5.8	59	jpeg	optimize JPEG files
optipng	V:2, I:39	187	png	optimize PNG files, lossless compression
pngquant	V:1, I:11	62	png	optimize PNG files, lossy compression

Table 11.19: List of graphics data tools (CLI)

11.7 Miscellaneous data conversion

There are many other programs for converting data. Following packages caught my eyes using regex "~Guse::converting in aptitude(8) (see Section 2.2.6).

package	popcon	size	keyword	description
alien	V:1, I:12	150	rpm/tgz → deb	converter for the foreign package into the Debian package
freepwing	V:0.00, I:0.01	447	EB → EPWING	converter from "Electric Book" (popular in Japan) to a single JIS X 4081 format (a subset of the EPWING V1)
calibre	V:7, I:24	71216	any → EPUB	e-book converter and library management

Table 11.20: List of miscellaneous data conversion tools

You can also extract data from RPM format with the following.

```
$ rpm2cpio file.src.rpm | cpio --extract
```

Chapter 12

Programming

I provide some pointers for people to learn programming on the Debian system enough to trace the packaged source code. Here are notable packages and corresponding documentation packages for programming.

Online references are available by typing "`man name`" after installing `manpages` and `manpages-dev` packages. Online references for the GNU tools are available by typing "`info program_name`" after installing the pertinent documentation packages. You may need to include the `contrib` and `non-free` archives in addition to the `main` archive since some GFDL documentations are not considered to be DFSG compliant.

Please consider to use version control system tools. See Section 10.5.



Warning
Do not use "`test`" as the name of an executable test file. "`test`" is a shell builtin.



Caution
You should install software programs directly compiled from source into "`/usr/local`" or "`/opt`" to avoid collision with system programs.

Tip
Code examples of creating "Song 99 Bottles of Beer" should give you good ideas of practically all the programming languages.

12.1 The shell script

The shell script is a text file with the execution bit set and contains the commands in the following format.

```
#!/bin/sh
 ... command lines
```

The first line specifies the shell interpreter which read and execute this file contents.

Reading shell scripts is the **best** way to understand how a Unix-like system works. Here, I give some pointers and reminders for shell programming. See "Shell Mistakes" (https://www.greenend.org.uk/rjk/2001/04/shell.html) to learn from mistakes.

Unlike shell interactive mode (see Section 1.5 and Section 1.6), shell scripts frequently use parameters, conditionals, and loops.

12.1.1 POSIX shell compatibility

Many system scripts may be interpreted by any one of POSIX shells (see Table 1.13).

- The default non-interactive POSIX shell "`/usr/bin/sh`" is a symlink pointing to `/usr/bin/dash` and used by many system programs.
- The default interactive POSIX shell is `/usr/bin/bash`.

Avoid writing a shell script with **bashisms** or **zshisms** to make it portable among all POSIX shells. You can check it using checkbashisms(1).

Good: POSIX	Avoid: bashism
`if [ "$foo" = "$bar" ] ; then …`	`if [ "$foo" == "$bar" ] ; then …`
`diff -u file.c.orig file.c`	`diff -u file.c{.orig,}`
`mkdir /foobar /foobaz`	`mkdir /foo{bar,baz}`
`funcname() { … }`	`function funcname() { … }`
octal format: "`\377`"	hexadecimal format: "`\xff`"

Table 12.1: List of typical bashisms

The "`echo`" command must be used with following cares since its implementation differs among shell builtin and external commands.

- Avoid using any command options except "`-n`".
- Avoid using escape sequences in the string since their handling varies.

Note
Although "`-n`" option is **not** really POSIX syntax, it is generally accepted.

Tip
Use the "`printf`" command instead of the "`echo`" command if you need to embed escape sequences in the output string.

12.1.2 Shell parameters

Special shell parameters are frequently used in the shell script.

Basic **parameter expansions** to remember are as follows.

Here, the colon "`:`" in all of these operators is actually optional.

- **with** "`:`" = operator test for **exist** and **not null**
- **without** "`:`" = operator test for **exist** only

shell parameter	value
`$0`	name of the shell or shell script
`$1`	first (1st) shell argument
`$9`	ninth (9th) shell argument
`$#`	number of positional parameters
`"$*"`	`"$1 $2 $3 $4 …"`
`"$@"`	`"$1" "$2" "$3" "$4" …`
`$?`	exit status of the most recent command
`$$`	PID of this shell script
`$!`	PID of most recently started background job

Table 12.2: List of shell parameters

parameter expression form	value if `var` is set	value if `var` is not set
`${var:-string}`	"`$var`"	"`string`"
`${var:+string}`	"`string`"	"`null`"
`${var:=string}`	"`$var`"	"`string`" (and run "`var=string`")
`${var:?string}`	"`$var`"	echo "`string`" to **stderr** (and exit with error)

Table 12.3: List of shell parameter expansions

parameter substitution form	result
`${var%suffix}`	remove smallest suffix pattern
`${var%%suffix}`	remove largest suffix pattern
`${var#prefix}`	remove smallest prefix pattern
`${var##prefix}`	remove largest prefix pattern

Table 12.4: List of key shell parameter substitutions

12.1.3 Shell conditionals

Each command returns an **exit status** which can be used for conditional expressions.

- Success: 0 (”True”)
- Error: non 0 (”False”)

Note
”0” in the shell conditional context means ”True”, while ”0” in the C conditional context means ”False”.

Note
”`[`” is the equivalent of the `test` command, which evaluates its arguments up to ”`]`” as a conditional expression.

Basic **conditional idioms** to remember are the following.

- ”*`command`* `&&` *`if_success_run_this_command_too`* `||` `true`”
- ”*`command`* `||` *`if_not_success_run_this_command_too`* `||` `true`”
- A multi-line script snippet as the following

```
if [ conditional_expression ]; then
 if_success_run_this_command
else
 if_not_success_run_this_command
fi
```

Here trailing ”`|| true`” was needed to ensure this shell script does not exit at this line accidentally when shell is invoked with ”`-e`” flag.

equation	condition to return logical true
`-e` *`file`*	*file* exists
`-d` *`file`*	*file* exists and is a directory
`-f` *`file`*	*file* exists and is a regular file
`-w` *`file`*	*file* exists and is writable
`-x` *`file`*	*file* exists and is executable
`file1` `-nt` *`file2`*	*file1* is newer than *file2* (modification)
`file1` `-ot` *`file2`*	*file1* is older than *file2* (modification)
`file1` `-ef` *`file2`*	*file1* and *file2* are on the same device and the same inode number

Table 12.5: List of file comparison operators in the conditional expression

Arithmetic integer comparison operators in the conditional expression are ”`-eq`”, ”`-ne`”, ”`-lt`”, ”`-le`”, ”`-gt`”, and ”`-ge`”.

12.1.4 Shell loops

There are several loop idioms to use in POSIX shell.

- ”`for x in foo1 foo2 … ; do command ; done`” loops by assigning items from the list ”`foo1 foo2 …`” to variable ”x” and executing ”`command`”.

equation	condition to return logical true
`-z` *str*	the length of *str* is zero
`-n` *str*	the length of *str* is non-zero
str1 `=` *str2*	*str1* and *str2* are equal
str1 `!=` *str2*	*str1* and *str2* are not equal
str1 `<` *str2*	*str1* sorts before *str2* (locale dependent)
str1 `>` *str2*	*str1* sorts after *str2* (locale dependent)

Table 12.6: List of string comparison operators in the conditional expression

- "`while condition ; do command ; done`" repeats "`command`" while "`condition`" is true.
- "`until condition ; do command ; done`" repeats "`command`" while "`condition`" is not true.
- "`break`" enables to exit from the loop.
- "`continue`" enables to resume the next iteration of the loop.

Tip
The C-language like numeric iteration can be realized by using seq(1) as the "`foo1 foo2 …`" generator.

Tip
See Section 9.4.9.

12.1.5 Shell environment variables

Some popular environment variables for the normal shell command prompt may not be available under the execution environment of your script.

- For "`$USER`", use "`$(id -un)`"
- For "`$UID`", use "`$(id -u)`"
- For "`$HOME`", use "`$(getent passwd "$(id -u)"|cut -d ":" -f 6)`" (this works also on Section 4.5.2)

12.1.6 The shell command-line processing sequence

The shell processes a script roughly as the following sequence.

- The shell reads a line.
- The shell groups a part of the line as **one token** if it is within "…" or '…'.
- The shell splits other part of a line into **tokens** by the following.
  - Whitespaces: *space tab newline*
  - Metacharacters: `< > | ; & ( )`
- The shell checks the **reserved word** for each token to adjust its behavior if not within "…" or '…'.
  - **reserved word**: `if then elif else fi for in while unless do done case esac`
- The shell expands **alias** if not within "…" or '…'.

- The shell expands **tilde** if not within "…" or '…'.
  - "~" → current user's home directory
  - "~*user*" → *user*'s home directory
- The shell expands **parameter** to its value if not within '…'.
  - **parameter**: "$PARAMETER" or "${PARAMETER}"
- The shell expands **command substitution** if not within '…'.
  - "$(command)" → the output of "command"
  - "` command `" → the output of "command"
- The shell expands **pathname glob** to matching file names if not within "…" or '…'.
  - * → any characters
  - ? → one character
  - […] → any one of the characters in "…"
- The shell looks up **command** from the following and execute it.
  - **function** definition
  - **builtin** command
  - **executable file** in "$PATH"
- The shell goes to the next line and repeats this process again from the top of this sequence.

Single quotes within double quotes have no effect.

Executing "set -x" in the shell or invoking the shell with "-x" option make the shell to print all of commands executed. This is quite handy for debugging.

12.1.7 Utility programs for shell script

In order to make your shell program as portable as possible across Debian systems, it is a good idea to limit utility programs to ones provided by **essential** packages.

- "aptitude search ~E" lists **essential** packages.
- "dpkg -L *package_name* |grep '/man/man.*/'" lists manpages for commands offered by *package_name* package.

Tip
Although moreutils may not exist outside of Debian, it offers interesting small programs. Most notable one is sponge(8) which is quite useful when you wish to overwrite original file.

See Section 1.6 for examples.

package	popcon	size	description
dash	V:929, I:999	207	small and fast POSIX-compliant shell for sh
coreutils	V:912, I:1000	17994	GNU core utilities
grep	V:782, I:1000	1297	GNU grep, egrep and fgrep
sed	V:769, I:1000	987	GNU sed
mawk	V:495, I:998	299	small and fast awk
debianutils	V:935, I:996	228	miscellaneous utilities specific to Debian
bsdutils	V:421, I:999	338	basic utilities from 4.4BSD-Lite
bsdextrautils	V:761, I:870	382	extra utilities from 4.4BSD-Lite
moreutils	V:18, I:39	232	additional Unix utilities

Table 12.7: List of packages containing small utility programs for shell scripts

package	popcon	size	documentation
dash	V:929, I:999	207	sh: small and fast POSIX-compliant shell for sh
bash	V:886, I:999	7308	sh: "info bash" provided by bash-doc
mawk	V:495, I:998	299	AWK: small and fast awk
gawk	V:247, I:294	3289	AWK: "info gawk" provided by gawk-doc
perl	V:663, I:990	840	Perl: perl(1) and html pages provided by perl-doc and perl-doc-html
libterm-readline-gnu-perl	V:2, I:27	439	Perl extension for the GNU ReadLine/History Library: perlsh(1)
libreply-perl	V:0.01, I:0.07	171	REPL for Perl: reply(1)
libdevel-repl-perl	V:0.02, I:0.50	237	REPL for Perl: re.pl(1)
python3	V:720, I:975	81	Python: python3(1) and html pages provided by python3-doc
tcl	V:24, I:173	20	Tcl: tcl(3) and detail manual pages provided by tcl-doc
tk	V:18, I:167	20	Tk: tk(3) and detail manual pages provided by tk-doc
ruby	V:72, I:154	32	Ruby: ruby(1), erb(1), irb(1), rdoc(1), ri(1)

Table 12.8: List of interpreter related packages

12.2 Scripting in interpreted languages

When you wish to automate a task on Debian, you should script it with an interpreted language first. The guide line for the choice of the interpreted language is:

- Use `dash`, if the task is a simple one which combines CLI programs with a shell program.
- Use `python3`, if the task isn't a simple one and you are writing it from scratch.
- Use `perl`, `tcl`, `ruby`, ... if there is an existing code using one of these languages on Debian which needs to be touched up to do the task.

If the resulting code is too slow, you can rewrite only the critical portion for the execution speed in a compiled language and call it from the interpreted language.

12.2.1 Debugging interpreted language codes

Most interpreters offer basic syntax check and code tracing functionalities.

- "**dash -n** *script.sh*" - Syntax check of a Shell script
- "**dash -x** *script.sh*" - Trace a Shell script
- "**python -m py_compile** *script.py*" - Syntax check of a Python script
- "**python -mtrace --trace** *script.py*" - Trace a Python script
- "**perl -I ../libpath -c** *script.pl*" - Syntax check of a Perl script
- "**perl -d:Trace** *script.pl*" - Trace a Perl script

For testing code for `dash`, try Section 9.1.4 which accommodates `bash`-like interactive environment.

For testing code for `perl`, try REPL environment for Perl which accommodates Python-like REPL (=READ + EVAL + PRINT + LOOP) environment for Perl.

12.2.2 GUI program with the shell script

The shell script can be improved to create an attractive GUI program. The trick is to use one of so-called dialog programs instead of dull interaction using `echo` and `read` commands.

package	popcon	size	description
`x11-utils`	V:228, I:555	651	xmessage(1): display a message or query in a window (X)
`whiptail`	V:307, I:997	61	displays user-friendly dialog boxes from shell scripts (newt)
`dialog`	V:8, I:77	520	displays user-friendly dialog boxes from shell scripts (ncurses)
`zenity`	V:68, I:341	193	display graphical dialog boxes from shell scripts (GTK)
`ssft`	V:0.01, I:0.16	80	Shell Scripts Frontend Tool (wrapper for zenity, kdialog, and dialog with gettext)
`gettext`	V:49, I:210	20549	"`/usr/bin/gettext.sh`": translate message

Table 12.9: List of dialog programs

Here is an example of GUI program to demonstrate how easy it is just with a shell script.

This script uses `zenity` to select a file (default `/etc/motd`) and display it.

GUI launcher for this script can be created following Section 9.4.10.

```
#!/bin/sh -e
# Copyright (C) 2021 Osamu Aoki <osamu@debian.org>, Public Domain
# vim:set sw=2 sts=2 et:
DATA_FILE=$(zenity --file-selection --filename="/etc/motd" --title="Select a file to check ↩
    ") || \
  ( echo "E: File selection error" >&2 ; exit 1 )
# Check size of archive
if ( file -ib "$DATA_FILE" | grep -qe '^text/' ) ; then
  zenity --info --title="Check file: $DATA_FILE" --width 640  --height 400 \
    --text="$(head -n 20 "$DATA_FILE")"
else
  zenity --info --title="Check file: $DATA_FILE" --width 640  --height 400 \
    --text="The data is MIME=$(file -ib "$DATA_FILE")"
fi
```

This kind of approach to GUI program with the shell script is useful only for simple choice cases. If you are to write any program with complexities, please consider writing it on more capable platform.

12.2.3 Custom actions for GUI filer

GUI filer programs can be extended to perform some popular actions on selected files using additional extension packages. They can also made to perform very specific custom actions by adding your specific scripts.

- For GNOME, see NautilusScriptsHowto.
- For KDE, see Creating Dolphin Service Menus.
- For Xfce, see Thunar - Custom Actions and https://help.ubuntu.com/community/ThunarCustomActions.
- For LXDE, see Custom Actions.

12.2.4 Perl short script madness

In order to process data, `sh` needs to spawn sub-process running `cut`, `grep`, `sed`, etc., and is slow. On the other hand, `perl` has internal capabilities to process data, and is fast. So many system maintenance scripts on Debian use `perl`.

Let's think following one-liner AWK script snippet and its equivalents in Perl.

```
awk '($2=="1957") { print $3 }' |
```

This is equivalent to any one of the following lines.

```
perl -ne '@f=split; if ($f[1] eq "1957") { print "$f[2]\n"}' |
```

```
perl -ne 'if ((@f=split)[1] eq "1957") { print "$f[2]\n"}' |
```

```
perl -ne '@f=split; print $f[2] if ( $f[1]==1957 )' |
```

```
perl -lane 'print $F[2] if $F[1] eq "1957"' |
```

```
perl -lane 'print$F[2]if$F[1]eq+1957' |
```

The last one is a riddle. It took advantage of following Perl features.

- The whitespace is optional.

- The automatic conversion exists from number to the string.
- Perl execution tricks via command line options: perlrun(1)
- Perl special variables: perlvar(1)

This flexibility is the strength of Perl. At the same time, this allows us to create cryptic and tangled codes. So be careful.

12.3 Coding in compiled languages

package	popcon	size	description
gcc	V:156, I:561	36	GNU C compiler
libc6-dev	V:258, I:580	12863	GNU C Library: Development Libraries and Header Files
g++	V:61, I:529	13	GNU C++ compiler
libstdc++-16-dev	V:3, I:19	24398	GNU Standard C++ Library v3 (development files)
cpp	V:337, I:720	18	GNU C preprocessor
gettext	V:49, I:210	20549	GNU Internationalization utilities
glade	V:0.2, I:2.6	1613	GTK User Interface Builder
valac	V:0.2, I:3.0	533	C# like language for the GObject system
flex	V:6, I:67	1247	LEX-compatible fast lexical analyzer generator
bison	V:7, I:71	3122	YACC-compatible parser generator
susv2	I:0.03	16	fetch "The Single UNIX Specifications v2"
susv3	I:0.05	16	fetch "The Single UNIX Specifications v3"
susv4	I:0.02	16	fetch "The Single UNIX Specifications v4"
golang	I:21	11	Go programming language compiler
rustc	V:5, I:20	13195	Rust systems programming language
gfortran	V:4, I:49	15	GNU Fortran 95 compiler
fpc	I:2.2	101	Free Pascal

Table 12.10: List of compiler related packages

Here, Section 12.3.3 and Section 12.3.4 are included to indicate how compiler-like program can be written in C language by compiling higher level description into C language.

12.3.1 C

You can set up proper environment to compile programs written in the C programming language by the following.

```
# apt-get install glibc-doc manpages-dev libc6-dev gcc build-essential
```

The `libc6-dev` package, i.e., GNU C Library, provides C standard library which is collection of header files and library routines used by the C programming language.

See references for C as the following.

- "`info libc`" (C library function reference)
- gcc(1) and "`info gcc`"
- each_C_library_function_name(3)
- Kernighan & Ritchie, "The C Programming Language", 2nd edition (Prentice Hall)

12.3.2 Simple C program (gcc)

A simple example "`example.c`" can compiled with a library "`libm`" into an executable "`run_example`" by the following.

```
$ cat > example.c << EOF
#include <stdio.h>
#include <math.h>
#include <string.h>

int main(int argc, char **argv, char **envp){
        double x;
        char y[11];
        x=sqrt(argc+7.5);
        strncpy(y, argv[0], 10); /* prevent buffer overflow */
        y[10] = '\0'; /* fill to make sure string ends with '\0' */
        printf("%5i, %5.3f, %10s, %10s\n", argc, x, y, argv[1]);
        return 0;
}
EOF
$ gcc -Wall -g -o run_example example.c -lm
$ ./run_example
        1, 2.915, ./run_exam,     (null)
$ ./run_example 1234567890qwerty
        2, 3.082, ./run_exam, 1234567890qwerty
```

Here, "`-lm`" is needed to link library "`/usr/lib/libm.so`" from the `libc6` package for sqrt(3). The actual library is in "`/lib/`" with filename "`libm.so.6`", which is a symlink to "`libm-2.7.so`".

Look at the last parameter in the output text. There are more than 10 characters even though "`%10s`" is specified.

The use of pointer memory operation functions without boundary checks, such as sprintf(3) and strcpy(3), is deprecated to prevent buffer overflow exploits that leverage the above overrun effects. Instead, use snprintf(3) and strncpy(3).

12.3.3 Flex — a better Lex

Flex is a Lex-compatible fast lexical analyzer generator.

Tutorial for flex(1) can be found in "`info flex`".

Many simple examples can be found under "`/usr/share/doc/flex/examples/`". 1

12.3.4 Bison — a better Yacc

Several packages provide a Yacc-compatible lookahead LR parser or LALR parser generator in Debian.

package	popcon	size	description
`bison`	V:7, I:71	3122	GNU LALR parser generator
`byacc`	V:0.1, I:2.9	263	Berkeley LALR parser generator
`btyacc`	V:0.01, I:0.04	247	backtracking parser generator based on `byacc`

Table 12.11: List of Yacc-compatible LALR parser generators

Tutorial for bison(1) can be found in "`info bison`".

1Some tweaks may be required to get them work under the current system.

You need to provide your own "`main()`" and "`yyerror()`". "`main()`" calls "`yyparse()`" which calls "`yylex()`", usually created with Flex.

Here is an example to create a simple terminal calculator program.

Let's create `example.y`:

```
/* calculator source for bison */
%{
#include <stdio.h>
extern int yylex(void);
extern int yyerror(char *);
%}

/* declare tokens */
%token NUMBER
%token OP_ADD OP_SUB OP_MUL OP_RGT OP_LFT OP_EQU

%%
calc:
 | calc exp OP_EQU    { printf("Y: RESULT = %d\n", $2); }
 ;

exp: factor
 | exp OP_ADD factor  { $$ = $1 + $3; }
 | exp OP_SUB factor  { $$ = $1 - $3; }
 ;

factor: term
 | factor OP_MUL term { $$ = $1 * $3; }
 ;

term: NUMBER
 | OP_LFT exp OP_RGT  { $$ = $2; }
  ;
%%

int main(int argc, char **argv)
{
  yyparse();
}

int yyerror(char *s)
{
  fprintf(stderr, "error: '%s'\n", s);
}
```

Let's create, `example.l`:

```
/* calculator source for flex */
%{
#include "example.tab.h"
%}

%%
[0-9]+ { printf("L: NUMBER = %s\n", yytext); yylval = atoi(yytext); return NUMBER; }
"+"    { printf("L: OP_ADD\n"); return OP_ADD; }
"-"    { printf("L: OP_SUB\n"); return OP_SUB; }
"*"    { printf("L: OP_MUL\n"); return OP_MUL; }
"("    { printf("L: OP_LFT\n"); return OP_LFT; }
")"    { printf("L: OP_RGT\n"); return OP_RGT; }
"="    { printf("L: OP_EQU\n"); return OP_EQU; }
```

```
"exit" { printf("L: exit\n");   return YYEOF; } /* YYEOF = 0 */
.      { /* ignore all other */ }
%%
```

Then execute as follows from the shell prompt to try this:

```
$ bison -d example.y
$ flex example.l
$ gcc -lfl example.tab.c lex.yy.c -o example
$ ./example
1 + 2 * ( 3 + 1 ) =
L: NUMBER = 1
L: OP_ADD
L: NUMBER = 2
L: OP_MUL
L: OP_LFT
L: NUMBER = 3
L: OP_ADD
L: NUMBER = 1
L: OP_RGT
L: OP_EQU
Y: RESULT = 9

exit
L: exit
```

12.4 Static code analysis tools

Lint like tools can help automatic static code analysis.

Indent like tools can help human code reviews by reformatting source codes consistently.

Ctags like tools can help human code reviews by generating an index (or tag) file of names found in source codes.

Tip
Configuring your favorite editor (`emacs` or `vim`) to use asynchronous lint engine plugins helps your code writing. These plugins are getting very powerful by taking advantage of Language Server Protocol. Since they are moving fast, using their upstream code instead of Debian package may be a good option.

12.5 Debug

Debug is important part of programming activities. Knowing how to debug programs makes you a good Debian user who can produce meaningful bug reports.

12.5.1 Basic gdb execution

Primary debugger on Debian is gdb(1) which enables you to inspect a program while it executes.

Let's install `gdb` and related programs by the following.

```
# apt-get install gdb gdb-doc build-essential devscripts
```

Good tutorial of `gdb` can be found:

package	popcon	size	description
vim-ale	I:0.74	2833	Asynchronous Lint Engine for Vim 8 and NeoVim
vim-syntastic	I:1.9	1379	Syntax checking hacks for vim
elpa-flycheck	V:0.1, I:1.6	1130	modern on-the-fly syntax checking for Emacs
elpa-relint	V:0.00, I:0.04	150	Emacs Lisp regexp mistake finder
cppcheck-gui	V:0.11, I:0.97	7973	tool for static C/C++ code analysis (GUI)
shellcheck	V:4, I:17	22860	lint tool for shell scripts
pyflakes3	V:2, I:14	20	passive checker of Python 3 programs
pylint	V:4, I:19	2104	Python code static checker
perl	V:663, I:990	840	interpreter with internal static code checker: `B::Lint`(3perl)
rubocop	V:0.1, I:1.0	4502	Ruby static code analyzer
clang-tidy	V:2, I:11	20	clang-based C++ linter tool
splint	V:0.05, I:0.79	2325	tool for statically checking C programs for bugs
flawfinder	V:0.05, I:0.43	187	tool to examine C/C++ source code and looks for security weaknesses
black	V:3, I:15	5868	uncompromising Python code formatter
perltidy	V:0.4, I:2.9	3086	Perl script indenter and reformatter
indent	V:0.3, I:4.7	438	C language source code formatting program
astyle	V:0.2, I:2.5	770	Source code indenter for C, C++, Objective-C, C#, and Java
bcpp	V:0.02, I:0.22	114	C(++) beautifier
xmlindent	V:0.07, I:0.73	52	XML stream reformatter
global	V:0.2, I:1.5	1923	Source code search and browse tools
exuberant-ctags	V:1, I:12	341	build tag file indexes of source code definitions
universal-ctags	V:1, I:12	4238	build tag file indexes of source code definitions

Table 12.12: List of tools for static code analysis

package	popcon	size	documentation
gdb	V:92, I:162	13472	"`info gdb`" provided by `gdb-doc`
ddd	V:0.3, I:4.9	5094	"`info ddd`" provided by `ddd-doc`

Table 12.13: List of debug packages

- “info gdb”
- “Debugging with GDB” in /usr/share/doc/gdb-doc/html/gdb/index.html
- “tutorial on the web”

Here is a simple example of using gdb(1) on a ”program” compiled with the ”-g” option to produce debugging information.

```
$ gdb program
(gdb) b 1                # set break point at line 1
(gdb) run args           # run program with args
(gdb) next               # next line
...
(gdb) step               # step forward
...
(gdb) p parm             # print parm
...
(gdb) p parm=12          # set value to 12
...
(gdb) quit
```

Tip
Many gdb(1) commands can be abbreviated. Tab expansion works as in the shell.

12.5.2 Debugging the Debian package

Since all installed binaries should be stripped on the Debian system by default, most debugging symbols are removed in the normal package. In order to debug Debian packages with gdb(1), *-dbgsym packages need to be installed (e.g. coreutils-dbgsym in the case of coreutils). The source packages generate *-dbgsym packages automatically along with normal binary packages and those debug packages are placed separately in debian-debug archive. Please refer to articles on Debian Wiki for more information.

If a package to be debugged does not provide its *-dbgsym package, you need to install it after rebuilding it by the following.

```
$ mkdir /path/new ; cd /path/new
$ sudo apt-get update
$ sudo apt-get dist-upgrade
$ sudo apt-get install fakeroot devscripts build-essential
$ apt-get source package_name
$ cd package_name*
$ sudo apt-get build-dep ./
```

Fix bugs if needed.

Bump package version to one which does not collide with official Debian versions, e.g. one appended with ”+debug1” when recompiling existing package version, or one appended with ”~pre1” when compiling unreleased package version by the following.

```
$ dch -i
```

Compile and install packages with debug symbols by the following.

```
$ export DEB_BUILD_OPTIONS="nostrip noopt"
$ debuild
$ cd ..
$ sudo debi package_name*.changes
```

You need to check build scripts of the package and ensure to use ”CFLAGS=-g -Wall” for compiling binaries.

12.5.3 Obtaining backtrace

When you encounter program crash, reporting bug report with cut-and-pasted backtrace information is a good idea.

The backtrace can be obtained by gdb(1) using one of the following approaches:

- Crash-in-GDB approach:
  - Run the program from GDB.
  - Crash the program.
  - Type "`bt`" at the GDB prompt.
- Crash-first approach:
  - Update the "**/etc/security/limits.conf**" file to include the following:

```
* soft core unlimited
```

  - Type "`ulimit -c unlimited`" to the shell prompt.
  - Run the program from this shell prompt.
  - Crash the program to produce a core dump file.
  - Load the core dump file to GDB as "`gdb gdb ./program_binary core`".
  - Type "`bt`" at the GDB prompt.

For infinite loop or frozen keyboard situation, you can force to crash the program by pressing `Ctrl-\` or `Ctrl-C` or executing "`kill -ABRT` *`PID`*". (See Section 9.4.12)

Tip

Often, you see a backtrace where one or more of the top lines are in "`malloc()`" or "`g_malloc()`". When this happens, chances are your backtrace isn't very useful. The easiest way to find some useful information is to set the environment variable "`$MALLOC_CHECK_`" to a value of 2 (malloc(3)). You can do this while running `gdb` by doing the following.

```
$ MALLOC_CHECK_=2 gdb hello
```

12.5.4 Advanced gdb commands

command	description for command objectives
`(gdb) thread apply all bt`	get a backtrace for all threads for multi-threaded program
`(gdb) bt full`	get parameters came on the stack of function calls
`(gdb) thread apply all bt full`	get a backtrace and parameters as the combination of the preceding options
`(gdb) thread apply all bt full 10`	get a backtrace and parameters for top 10 calls to cut off irrelevant output
`(gdb) set logging on`	write log of `gdb` output to a file (the default is "`gdb.txt`")

Table 12.14: List of advanced gdb commands

12.5.5 Check dependency on libraries

Use ldd(1) to find out a program's dependency on libraries by the followings.

```
$ ldd /usr/bin/ls
        librt.so.1 => /lib/librt.so.1 (0x4001e000)
        libc.so.6 => /lib/libc.so.6 (0x40030000)
        libpthread.so.0 => /lib/libpthread.so.0 (0x40153000)
        /lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0x40000000)
```

For ls(1) to work in a `chroot`ed environment, the above libraries must be available in your `chroot`ed environment. See Section 9.4.6.

12.5.6 Dynamic call tracing tools

There are several dynamic call tracing tools available in Debian. See Section 9.4.

12.5.7 Debugging X Errors

If a GNOME program `preview1` has received an X error, you should see a message as follows.

```
The program 'preview1' received an X Window System error.
```

If this is the case, you can try running the program with "`--sync`", and break on the "`gdk_x_error`" function in order to obtain a backtrace.

12.5.8 Memory leak detection tools

There are several memory leak detection tools available in Debian.

package	popcon	size	description
libc6-dev	V:258, I:580	12863	mtrace(1): malloc debugging functionality in glibc
valgrind	V:4, I:32	89318	memory debugger and profiler
electric-fence	V:0.2, I:2.2	69	malloc(3) debugger
libdmalloc5	V:0.01, I:0.57	373	debug memory allocation library
duma	V:0.00, I:0.05	297	library to detect buffer overruns and under-runs in C and C++ programs
leaktracer	V:0.01, I:0.27	56	memory-leak tracer for C++ programs

Table 12.15: List of memory leak detection tools

12.5.9 Disassemble binary

You can disassemble binary code with objdump(1) by the following.

```
$  objdump -m i386 -b binary -D /usr/lib/grub/x86_64-pc/stage1
```

Note
gdb(1) may be used to disassemble code interactively.

12.6 Build tools

package	popcon	size	documentation
make	V:145, I:562	1762	"info make" provided by make-doc
autoconf	V:27, I:193	2244	"info autoconf" provided by autoconf-doc
automake	V:26, I:192	1958	"info automake" provided by automake1.10-doc
libtool	V:23, I:176	1276	"info libtool" provided by libtool-doc
cmake	V:22, I:123	59735	cmake(1) cross-platform, open-source make system
ninja-build	V:8, I:54	456	ninja(1) small build system closest in spirit to Make
meson	V:6, I:29	4402	meson(1) high productivity build system on top of ninja
xutils-dev	V:0.6, I:6.7	1495	imake(1), xmkmf(1), etc.

Table 12.16: List of build tool packages

12.6.1 Make

Make is a utility to maintain groups of programs. Upon execution of make(1), make read the rule file, "Makefile", and updates a target if it depends on prerequisite files that have been modified since the target was last modified, or if the target does not exist. The execution of these updates may occur concurrently.

The rule file syntax is the following.

```
target: [ prerequisites ... ]
 [TAB]  command1
 [TAB]  -command2 # ignore errors
 [TAB]  @command3 # suppress echoing
```

Here "[TAB]" is a TAB code. Each line is interpreted by the shell after make variable substitution. Use "\" at the end of a line to continue the script. Use "$$" to enter "$" for environment values for a shell script.

Implicit rules for the target and prerequisites can be written, for example, by the following.

```
%.o: %.c header.h
```

Here, the target contains the character "%" (exactly one of them). The "%" can match any nonempty substring in the actual target filenames. The prerequisites likewise use "%" to show how their names relate to the actual target name.

automatic variable	value
$@	target
$<	first prerequisite
$?	all newer prerequisites
$^	all prerequisites
$*	"%" matched stem in the target pattern

Table 12.17: List of make automatic variables

variable expansion	description
foo1 := bar	one-time expansion
foo2 = bar	recursive expansion
foo3 += bar	append

Table 12.18: List of make variable expansions

Run "make -p -f/dev/null" to see automatic internal rules.

12.6.2 Autotools

Autotools is a suite of programming tools designed to assist in making source code packages portable to many Unix-like systems.

- Autoconf is a tool to produce a shell script "`configure`" from "`configure.ac`".
  - "`configure`" is used later to produce "`Makefile`" from "`Makefile.in`" template.
- Automake is a tool to produce "`Makefile.in`" from "`Makefile.am`".
- Libtool is a shell script to address the software portability problem when compiling shared libraries from source code.

12.6.2.1 Compile and install a program



Warning
Do not overwrite system files with your compiled programs when installing them.

Debian does not touch files in "`/usr/local/`" or "`/opt`". So if you compile a program from source, install it into "`/usr/local/`" so it does not interfere with Debian.

```
$ cd src
$ ./configure --prefix=/usr/local
$ make # this compiles program
$ sudo make install # this installs the files in the system
```

12.6.2.2 Uninstall program

If you have the original source and if it uses autoconf(1)/automake(1) and if you can remember how you configured it, execute as follows to uninstall the program.

```
$ ./configure all-of-the-options-you-gave-it
$ sudo make uninstall
```

Alternatively, if you are absolutely sure that the install process puts files only under "`/usr/local/`" and there is nothing important there, you can erase all its contents by the following.

```
# find /usr/local -type f -print0 | xargs -0 rm -f
```

If you are not sure where files are installed, you should consider using checkinstall(8) from the `checkinstall` package, which provides a clean path for the uninstall. It now supports to create a Debian package with "`-D`" option.

12.6.3 Meson

The software build system has been evolving:

- Autotools on the top of Make has been the de facto standard for the portable build infrastructure since 1990s. This is extremely slow.
- CMake initially released in 2000 improved speed significantly but was originally built on the top of inherently slow Make. (Now Ninja can be its backend.)

- Ninja initially released in 2012 is meant to replace Make for the further improved build speed and is designed to have its input files generated by a higher-level build system.
- Meson initially released in 2013 is the new popular and fast higher-level build system which uses Ninja as its backend.

See documents found at "The Meson Build system" and "The Ninja build system".

12.7 Web

Basic interactive dynamic web pages can be made as follows.

- Queries are presented to the browser user using HTML forms.
- Filling and clicking on the form entries sends one of the following URL string with encoded parameters from the browser to the web server.
  - "`https://www.foo.dom/cgi-bin/program.pl?VAR1=VAL1&VAR2=VAL2&VAR3=VAL3`"
  - "`https://www.foo.dom/cgi-bin/program.py?VAR1=VAL1&VAR2=VAL2&VAR3=VAL3`"
  - "`https://www.foo.dom/program.php?VAR1=VAL1&VAR2=VAL2&VAR3=VAL3`"
- "%nn" in URL is replaced with a character with hexadecimal `nn` value.
- The environment variable is set as: "`QUERY_STRING="VAR1=VAL1 VAR2=VAL2 VAR3=VAL3"`".
- CGI program (any one of "`program.*`") on the web server executes itself with the environment variable "`$QUERY_STRING`".
- `stdout` of CGI program is sent to the web browser and is presented as an interactive dynamic web page.

For security reasons it is better not to hand craft new hacks for parsing CGI parameters. There are established modules for them in Perl and Python. PHP comes with these functionalities. When client data storage is needed, HTTP cookies are used. When client side data processing is needed, Javascript is frequently used.

For more, see the Common Gateway Interface, The Apache Software Foundation, and JavaScript.

Searching "CGI tutorial" on Google by typing encoded URL https://www.google.com/search?hl=en&ie=UTF-8&q=CGI+tutoria directly to the browser address is a good way to see the CGI script in action on the Google server.

12.8 The source code translation

There are programs to convert source codes.

package	popcon	size	keyword	description
`perl`	V:663, I:990	840	AWK → PERL	convert source codes from AWK to PERL: a2p(1)
`f2c`	V:0.1, I:1.9	443	FORTRAN → C	convert source codes from FORTRAN 77 to C/C++: f2c(1)
`intel2gas`	V:0.00, I:0.16	178	NASM → GAS	converter from NASM (Intel format) to the GNU Assembler (GAS)

Table 12.19: List of source code translation tools

12.9 Making Debian package

If you want to make a Debian package, read followings.

- Chapter 2 to understand the basic package system
- Section 2.7.13 to understand basic porting process
- Section 9.11.4 to understand basic chroot techniques
- debuild(1), and sbuild(1)
- Section 12.5.2 for recompiling for debugging
- Guide for Debian Maintainers (the `debmake-doc` package)
- Debian Developer's Reference (the `developers-reference` package)
- Debian Policy Manual (the `debian-policy` package)

There are packages such as `debmake`, `dh-make`, `dh-make-perl`, etc., which help packaging.

Appendix A

Appendix

Here are backgrounds of this document.

A.1 The Debian maze

The Linux system is a very powerful computing platform for a networked computer. However, learning how to use all its capabilities is not easy. Setting up the LPR printer queue with a non-PostScript printer was a good example of stumble points. (There are no issues anymore since newer installations use the new CUPS system.)

There is a complete, detailed map called the ”SOURCE CODE”. This is very accurate but very hard to understand. There are also references called HOWTO and mini-HOWTO. They are easier to understand but tend to give too much detail and lose the big picture. I sometimes have a problem finding the right section in a long HOWTO when I need a few commands to invoke.

I hope this ”Debian Reference (version 2.152)” (2026-10-10 14:56:25 UTC) provides a good starting direction for people in the Debian maze.

A.2 Copyright history

The Debian Reference was initiated by me, Osamu Aoki <osamu at debian dot org>, as a personal system administration memo. Many contents came from the knowledge I gained from the debian-user mailing list and other Debian resources.

Following a suggestion from Josip Rodin, who was very active with the Debian Documentation Project (DDP), ”Debian Reference (version 1, 2001-2007)” was created as a part of DDP documents.

After 6 years, I realized that the original ”Debian Reference (version 1)” was outdated and started to rewrite many contents. New ”Debian Reference (version 2)” is released in 2008.

I have updated ”Debian Reference (version 2)” to address new topics (Systemd, Wayland, IMAP, PipeWire, Linux kernel 5.10) and removed outdated topics (SysV init, CVS, Subversion, SSH protocol 1, Linux kernels before 2.5). References to Jessie 8 (2015-2020) release situation or older are mostly removed.

This ”Debian Reference (version 2.152)” (2026-10-10 14:56:25 UTC) covers mostly Trixie (`=stable`) and Forky (`=testing`) Debian releases.

The tutorial contents can trace its origin and its inspiration in followings.

- ”Linux User’s Guide” by Larry Greenfield (December 1996)
  - obsoleted by ”Debian Tutorial”

- ”Debian Tutorial” by Havoc Pennington. (11 December, 1998)
  - partially written by Oliver Elphick, Ole Tetlie, James Treacy, Craig Sawyer, and Ivan E. Moore II
  - obsoleted by ”Debian GNU/Linux: Guide to Installation and Usage”
- ”Debian GNU/Linux: Guide to Installation and Usage” by John Goerzen and Ossama Othman (1999)
  - obsoleted by ”Debian Reference (version 1)”

The package and archive description can trace some of their origin and their inspiration in following.

- ”Debian FAQ” (March 2002 version, when this was maintained by Josip Rodin)

The other contents can trace some of their origin and their inspiration in following.

- ”Debian Reference (version 1)” by Osamu Aoki (2001–2007)
  - obsoleted by the newer ”Debian Reference (version 2)” in 2008.

The previous ”Debian Reference (version 1)” was created with many contributors.

- the major contents contribution on network configuration topics by Thomas Hood
- significant contents contribution on X and VCS related topics by Brian Nelson
- the help on the build scripts and many content corrections by Jens Seidel
- extensive proofreading by David Sewell
- many contributions by the translators, contributors, and bug reporters

Many manual pages and info pages on the Debian system as well as upstream web pages and Wikipedia documents were used as the primary references to write this document. To the extent Osamu Aoki considered within the fair use, many parts of them, especially command definitions, were used as phrase pieces after careful editorial efforts to fit them into the style and the objective of this document.

The gdb debugger description was expanded using Debian wiki contents on backtrace with consent by Ari Pollak, Loïc Minier, and Dafydd Harries.

Contents of the current ”Debian Reference (version 2.152)” (2026-10-10 14:56:25 UTC) are mostly my own work except as mentioned above. These has been updated by the contributors too.

The author, Osamu Aoki, thanks all those who helped make this document possible.

A.3 Document format

The source of the English original document is currently written in DocBook XML files. This Docbook XML source are converted to HTML, plain text, PostScript, and PDF. (Some formats may be skipped for distribution.)