

Contents

Preface	viii
Chapter 1 The Computing Environment	1
<i>Chapter overview</i>	1
1.1 What is a Computer?	1
1.2 Hardware	2
1.3 Software	4
1.4 History of UNIX and Linux	8
1.5 Conventions	10
<i>Chapter summary</i>	10
Chapter 2 UNIX and Linux Design and Organisation	11
<i>Chapter overview</i>	11
2.1 The Kernel and Shell	11
2.2 Files	13
2.3 Technical Basics	14
2.4 How to get Linux	16
<i>Chapter summary</i>	16
Chapter 3 Installing Linux	17
<i>Chapter overview</i>	17
3.1 Starting out	17
3.2 Preliminaries	18
3.3 Single boot	19
3.4 Dual boot	19
3.5 Emulators	21
3.6 Installing Linux	22
3.7 Using Linux	25
3.8 KDE	26
<i>Chapter summary</i>	30

Chapter 4	Getting started	31
	<i>Chapter overview</i>	31
4.1	Using UNIX	31
4.2	Logging out	33
4.3	Commands	33
4.4	Communication with other users	36
4.5	Files	39
4.6	Input and output	44
4.7	Emergencies	54
4.8	Getting help	55
	<i>Chapter summary</i>	57
Chapter 5	Files	59
	<i>Chapter overview</i>	59
5.1	The UNIX directory hierarchy	59
5.2	Filesystems	62
5.3	Manipulating files	64
5.4	Protecting files	67
5.5	File contents	73
5.6	Printing files	81
5.7	File archives and file compression	83
5.8	Other relevant commands	84
	<i>Chapter summary</i>	86
Chapter 6	Processes and devices	88
	<i>Chapter overview</i>	88
6.1	Processes	88
6.2	Environment	92
6.3	Program control	100
6.4	Quotes and escapes	110
6.5	Devices	111
6.6	Backquotes	113
	<i>Chapter summary</i>	115
Chapter 7	Introduction to shells	117
	<i>Chapter overview</i>	117
7.1	Why do we need a shell?	117
7.2	Shell syntax	118
7.3	Arithmetic	126
7.4	Making decisions	128
7.5	Loops	134

7.6	Searching for files	137
7.7	Formatted output	139
7.8	Passing information to scripts	142
	<i>Chapter summary</i>	148
Chapter 8	More on shells	150
	<i>Chapter overview</i>	150
8.1	Simple arithmetic	150
8.2	Pattern matching	154
8.3	Entering and leaving the shell	159
8.4	More about scripts with options	162
8.5	Symbolic links	165
8.6	Setting up terminals	166
8.7	Conventions used in UNIX file systems	168
	<i>Chapter summary</i>	170
Chapter 9	Advanced shell programming	173
	<i>Chapter overview</i>	173
9.1	Sending and trapping signals	173
9.2	Functions	175
9.3	Aliases	177
9.4	The ‘exec’ mechanism	178
9.5	The ‘eval’ mechanism	179
9.6	Sending data across networks	180
9.7	Makefiles	183
9.8	Safe programming	186
9.9	Setting up a terminal	187
9.10	More on files	188
9.11	Miscellaneous utilities	191
	<i>Chapter summary</i>	192
Chapter 10	Regular expressions and filters	194
	<i>Chapter overview</i>	194
10.1	Using filters	194
10.2	Character-to-character transformation	196
10.3	Selecting lines by content	198
10.4	Stream editor	203
10.5	Splitting a file according to context	206
10.6	Choosing between the three filters	210
10.7	More on Vi	210
	<i>Chapter summary</i>	212

Chapter 11 Awk	214
<i>Chapter overview</i>	214
11.1 What is ‘awk’?	214
11.2 Invoking ‘awk’	215
11.3 Naming the fields	216
11.4 Formatted output	217
11.5 Patterns	220
11.6 Variables	222
11.7 Arguments to ‘awk’ scripts	225
11.8 Arrays	226
11.9 Field and record separators	229
11.10 Functions	233
<i>Chapter summary</i>	236
Chapter 12 Perl	239
<i>Chapter overview</i>	239
12.1 Introduction	239
12.2 Variables	241
12.3 Input and output	242
12.4 Fields	246
12.5 Control structures	247
12.6 Predefined Perl	249
12.7 Regular expressions	251
12.8 Perl and the Kernel	253
12.9 Quality code	254
12.10 When do I use Perl?	256
<i>Chapter summary</i>	257
Chapter 13 Maintaining your Linux OS	258
<i>Chapter overview</i>	258
13.1 Basic management	259
13.2 Linux file management	262
13.3 Linux networking	264
13.4 Security	268
13.5 Uninstalling Linux	269
<i>Chapter summary</i>	270
Chapter 14 Other Issues	271
<i>Chapter overview</i>	271
14.1 Programming languages	271
14.2 Document Preparation	273

14.3	Other Software	274
14.4	Useful Resources	275
	<i>Chapter summary</i>	277
	Answers to problems	278
	Appendix – summary of utilities	291
	Index	296

The Computing Environment

CHAPTER OVERVIEW

This chapter

- ▶ reviews basic notions of computer hardware and software;
- ▶ outlines the different kinds of software program;
- ▶ introduces the basic philosophy of UNIX and Linux; and
- ▶ provides a brief description of the history of UNIX and Linux.

If you pick up any book over ten years old on the subject of computing, you could get quite different ideas of how people use their computers. The basic ways of using computers haven't changed, but modern computing places an unimagined amount of control and power with the individual user. This means that the user now has the ability (and quite often the need) to deal with issues relating to the administration of the computer to get the best out of it. In this book, we'll be explaining just how to understand what this involves, and how to minimise the amount of effort required for effective use of your computer.

We start in this chapter by reviewing some basic concepts of computing in a non-technical way, so that if you really are a beginner, reading through this chapter should bring you up to speed. If you are already familiar with the ideas of hardware and software, input and output, processors, systems software, and applications programs, you may choose instead to move swiftly on to Chapter 2.1, or simply to skim this chapter.

1.1 What is a Computer?

In very basic terms, there are essentially two kinds of “thing” involved in computing. There are things you can kick, actual bits of machinery that you can pick up and take away, including the computer itself, printers,

screens and other physical devices (digital cameras, scanners, disk drives, CD drives, etc.), which are collectively and individually known as **hardware**. Thus, hardware includes the devices you use to communicate with a computer system (such as the mouse, keyboard), the actual components that make up that system, and any other devices.

Unfortunately, the hardware won't work by itself and needs detailed instructions, or **programs**, to make it do what it should. In addition to the hardware, therefore, it is also necessary to have a set of programs that tell the hardware what to do. These programs, which refer to the actual instructions rather than the medium on which they are stored, are collectively known as **software**. Software is needed for the basic operation of computers (like the software that is the subject of this book, UNIX and Linux) as well as for the more common applications that you may already be familiar with, such as word-processing, spreadsheets, games, MP3 playing, and limitless other possibilities. By themselves, hardware and software are not enough to do the things we want of computers — it is the combination of hardware and software that enables effective use of modern computers.

Below, we describe the different kinds of hardware and software in a little more detail.

1.2 Hardware

1.2.1 Processors

The most important part of the overall system is the **processor** (or **central processing unit**, **CPU**) on which the computer is based, and which does the main work of the system. In recent years, the advance of the PC has been fuelled by progress in such processors, which are becoming ever faster and more powerful. In PCs, these have included the series of Pentium processors developed by Intel, with alternatives from companies like AMD. Other computers have different processors, like Sun's SPARC processor. Whichever processor your machine uses is not important for now — there may be variations in speed and power, as well as in some other more technical differences, but the key point to note is that this is the main component of the machine.

1.2.2 Input Devices

Although the processor is most critical, it is of little use if you can't display the *results* of the computation it performs, or if you can't specify and modify the *kinds* of computation you want it to perform. For this reason, we need **input** and **output** devices — hardware components that allow users to interact with the processor in easy and convenient ways.

ACRONYM

AMD = 'Advanced
Micro Devices, Inc.'

SPARC = 'Scalable
Processor ARChitecture'

In order to instruct a computer to perform a task, we require a way to provide instructions as input. Perhaps the most recognisable input device is the **keyboard** (typically of the ‘QWERTY’ variety because of the layout of the keys, similar to a typewriter), which nearly all computers use to receive textual and numeric input. The keyboard is the most usual way in which people write programs or enter data on which those programs might operate. However, there are also many other ways to provide input to a computer. For example, many people now have scanners to enable graphical images to be provided as input. Similarly, digital cameras, bar-code readers in shops, and even sound recorders offer different ways of getting data to the processor. In this book, we will focus on the standard keyboard as our main input device, but we also note that the **mouse**, with the purpose of enabling the selection and movement of items displayed on the screen, is a vital part of modern computer systems.

1.2.3 Output Devices

Output devices are also varied, and we will focus primarily on the **screen** or **monitor** (or even **visual display unit** — **VDU** — to use a somewhat out-of-date expression) that typically comes as part of the package with the processor and the keyboard.

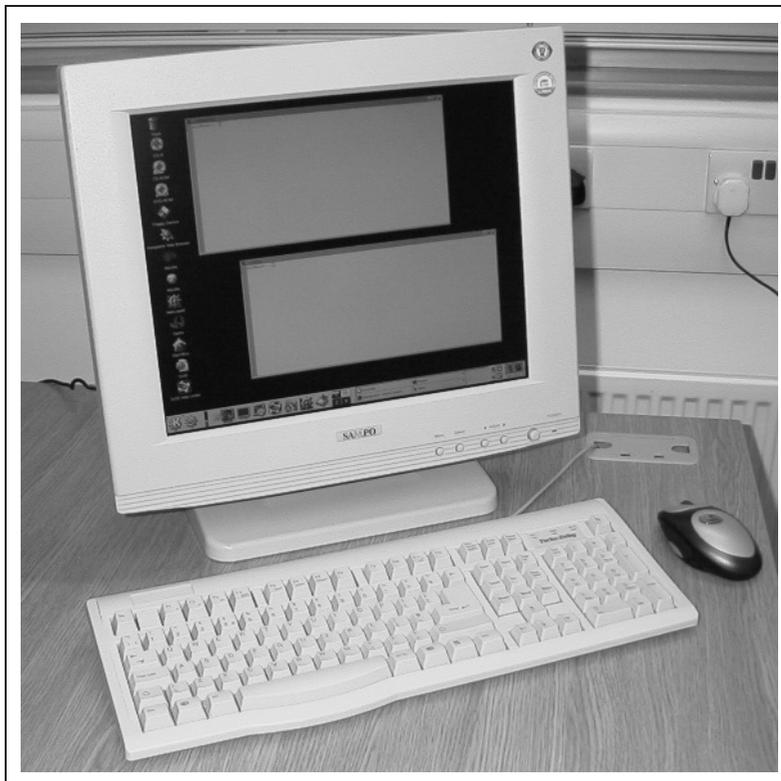
In the past, people used so-called **dumb terminals**, which are largely redundant now. A dumb terminal consists of a keyboard and a screen, and can display only the same sort of text and simple characters as a typewriter. On the screen is a **cursor**, which is either a block (a filled rectangle the size of a letter) or an underscore, marking the point on the screen at which typed characters appear, and also where any message the computer writes will begin. The equivalent of a dumb terminal is now more commonly referred to as a **command window** in many systems.

These days, modern computers typically use a screen (to display both the input that is provided through keyboards, for example, and any results of the processing performed), a keyboard and a mouse. The configuration is sometimes referred to as a **graphics terminals**, to distinguish it from a dumb terminal. These are capable of much more sophisticated output. In particular, the screen is a high-resolution display (nearly always colour), allowing complex graphics to be drawn as well as simple characters. Usually, a system is employed by which the screen is divided up into rectangular areas called **windows**, with which to communicate individually. The computer itself may be a PC, as shown in Figure 1.1, a **workstation** or a laptop, but that is not important. We will assume the use of a command window, which applies to all equally. If you are using any of these, you can create such a window, which behaves as if it were itself a dumb terminal, having its own cursor. Typically, there is also a global cursor that moves each time you move the mouse; you can

select which window the keyboard will communicate with by moving the global cursor so that it is within the chosen window.

The look and feel of the various kinds of computers can vary enormously, as can the way in which windows on a screen are manipulated. Either a **window manager** or a **desktop manager** can be used to control the windows on a screen, and though the distinction between the two kinds of software is somewhat blurred, a desktop manager typically has more features and capabilities than a window manager. We will go into more details about these later on in Chapter 3.

Figure 1.1 A typical computer, with screen, keyboard and mouse



These basic components of processor, screen and keyboard are the key pieces of the modern computing system, and their combination underlies all of the details that follow in this book.

1.3 Software

1.3.1 Input and Characters

When communicating with UNIX, users send to the system a **stream** of characters. Each time a key on the keyboard is pressed, a character is sent to the machine. Usually, the computer **echoes** the character so that it is

displayed on the screen. Similarly, to communicate with the user the system sends a stream of characters to the user's computer, which is able to interpret the particular character coding and display the characters on the screen accordingly.

While interacting with the system, a user types in lines of text from the keyboard, terminating each line by pressing the **Return** (or **Enter**) key. These lines are interpreted as instructions to UNIX, which then responds accordingly, and displays messages on the screen. On a graphics terminal, this dialogue is between the keyboard and a specific window (typically the window over which the cursor has been placed). Manipulation of other devices such as a mouse also results in the transmission of characters to the UNIX machine. However, these are interpreted as relating to the management and display of the windows only.

Commands sent to UNIX are executed by a program called the **shell**. We shall see later that the shell is just one example of a program that can run on a UNIX system, but is special since it is the principal interface between a user and the very heart of the system software, known as the **kernel**.

Most characters that we shall use are the **printing characters**. These, which include letters, digits, punctuation marks and the other symbols marked on the keyboard, are displayed in the obvious way. However, other characters, known as **control characters** (listed in Table 1.1), are sometimes required by a UNIX system. Each is stored as a number using a **character encoding** such as **ASCII**. For instance, the character whose code is 7 and is sometimes referred to as 'bell', if printed on your terminal, normally causes the terminal to make a noise (typically a 'beep').

Each control character has a name, typically an acronym of its description. For character number 7, this is **BEL**, short for 'bell'. Control characters can be typed by pressing a key while holding down the **Ctrl** key. For BEL, this key is G, and for this reason BEL is often written as **ctrl-G** or **^G**. Some of the other control characters have anachronistic names that also relate to the functioning of a teletype, but most of them will not concern us here.

Control characters have purposes which, for the most part, are obscure. Many are used by operating systems (not necessarily UNIX) to structure data, and have meanings with historical relevance only. Some of the more useful control characters include the following. The character **TAB** has the effect, when sent to the screen, of moving the cursor to the next tab position (usually columns 8, 16, 24, 32, etc.). The key marked TAB or \rightarrow , when pressed, transmits a TAB character to the computer. The character **NL** (Newline) causes the cursor to move down to the left-hand side of the next row on the screen. This character is provided as input to the machine whenever the key marked RETURN (or ENTER or \leftarrow) is pressed. The escape character, which would not normally be

ACRONYM

ASCII = 'American Standard Code for Information Interchange'

NOTE

The bell character was originally used on 'teletype' terminals to attract the attention of the user in the days of the telegraph

Table 1.1 ASCII control characters

Code	ctrl-key	Name	Description
0	^@	NUL	null
1	^A	SOH	start of heading
2	^B	STX	start of text
3	^C	ETX	end of text
4	^D	EOT	end of transmission
5	^E	ENQ	enquiry
6	^F	ACK	acknowledge
7	^G	BEL	bell
8	^H	BS	backspace
9	^I	HT	horizontal tab
10	^J	NL	newline (linefeed)
11	^K	VT	vertical tab
12	^L	NP	new page (formfeed)
13	^M	CR	carriage return
14	^N	SO	shift out
15	^O	SI	shift in
16	^P	DLE	data link escape
17	^Q	DC1	device control 1
18	^R	DC2	device control 2
19	^S	DC3	device control 3
20	^T	DC4	device control 4
21	^U	NAK	negative acknowledgement
22	^V	SYN	synchronous idle
23	^W	ETB	end of transmission block
24	^X	CAN	cancel
25	^Y	EM	end of medium
26	^Z	SUB	substitute
27	^[ESC	escape
28	^	FS	file separator
29	^]	GS	group separator
30	^^	RS	record separator
31	^_	US	unit separator
127	^?	DEL	delete

displayed on a screen at all, is sometimes required when typing in data. There should be a key on your keyboard marked ESC or ESCAPE.

1.3.2 Application Programs

As mentioned earlier, the hardware alone is not enough. To do anything useful with a computer, you need to run software, or programs, on the hardware. Programs can be used to do pretty much anything you want, but commonly include word-processing, scientific calculations and games, and even support the development of yet more programs. What is important to note here is that for each different application, you need a different **application program** to run on the computer. Thus, if you want to do some word-processing, you'll need to get a word-processing program to execute; word-processing can't be done directly by the computer otherwise.

Programming Languages

The processing units inside a computer understand a language called **machine code**, and all the calculations that a computer performs use this code. Machine code, which is a 'low-level' language, is specific to the particular make and model of computer on which it runs, and is not designed to be read by humans. Any instruction given to a computer must be translated (somehow) to machine code before the computer will understand it. It is unlikely you will ever need to come into direct contact with machine code.

Typically, programs are written in high-level languages that are easily readable by humans, but not by computers. They require **compilers** and **interpreters** to perform a translation into the machine code that computers can understand.

1.3.3 The Operating System

The final piece of the jigsaw of modern computing, to make the hardware and software work together, and to make the different bits of hardware like the screen, keyboard and processor talk to each other, is what is known as the **operating system**, which is **system software** as opposed to application software. The operating system is a complex program (or collection of programs) that controls the internal operation of the computer to ensure that all of the different things that are taking place at the same time are done effectively and sensibly. For example, while the computer accepts input from the keyboard and displays output on the screen, it may also be processing some data and accessing the hard disk, all at the same time.

Just as there can be different processors, different screens and keyboards, and different application programs, so there can be different

ACRONYM

DOS = ‘Disk Operating System’

CPM = ‘Control Program for

Microcomputers’

VMS = ‘Virtual Memory System’

operating systems. If you’ve got this far, then we should be able to assume that you know that your particular operating system is (or is likely to become) UNIX or Linux, but you may also be familiar with Microsoft’s **Windows**, with **DOS**, or with some other operating systems such as **CPM**, **MacOS**, **Multics**, **BeOS**, **PalmOS** or **VMS**.

1.3.4 System Administration

Traditionally, UNIX systems have been multi-user systems with individuals simply gaining access as users. In these situations, there is usually someone, somewhere, who is in day-to-day charge of the system, and known as the **system administrator**. If you are using this kind of system and have problems that neither you nor your colleagues are able to resolve, then the system administrator will either be able to help, or at least point you in the direction of someone who can. You should find out who your system administrator is, and make sure that you are in possession of any documents that he or she wishes users of the system to have.

More recently, however, there has been a move towards the use of UNIX for individually-run personal computers, especially with the recent success of Linux. If this is your situation, then it is you who will act as the system administrator for your machine, and will be responsible for its maintenance. In particular, if you are using Linux on your own personal computer, make sure you read the handbook supplied with the operating system in conjunction with this book. If there are any differences, it will be an invaluable help.

Finally, there is one user of the system who is called the **super-user**. He or she has special privileges on the system, and is allowed to perform certain actions forbidden to ordinary users, such as having the unrestricted right to change and to delete files on the system. The super-user may or may not be the same person as the system administrator.

1.4 History of UNIX and Linux

The first UNIX system was built at **Bell Labs**, the research division of the US telephone corporation **AT&T**, in 1969. Prior to that date, Bell (together with **General Electric** and **MIT**) had been engaged in developing a large-scale operating system, known as ‘Multics’. This collaboration between industry and the academic community had begun in 1964, but five years later it became clear that the three participants had different goals for the project. By this time a vast amount of work had gone into Multics, but more needed to be done for it to fulfil the aspirations of any of the participants. Accordingly, Bell Labs pulled out of the project.

ACRONYM

MIT = ‘Massachusetts Institute of Technology’

Faced with not having a state-of-the-art operating system with which to work, a number of researchers at Bell, led by Ken Thompson and Dennis Ritchie, decided to create a new operating system ‘from scratch’. Multics had become complex, and it was felt that a much simpler system was needed — the name ‘UNIX’ arose to emphasise that difference between it and Multics. The experience gained during the development of Multics contributed much to the design of UNIX.

A number of fundamental design decisions that were taken pervade the whole of UNIX. Programs written for UNIX should be simple, and should each do a single task well. This was different from the style adopted in some other operating systems, where large programs would be developed with many different capabilities, and would be commensurately complex. Also, programs should be designed so that they could easily be linked together, the output from one becoming the input to another. Thus it would be possible to build more complex programs by joining simple ones together.

Part of the philosophy underlying the design of UNIX was that the core system software, or kernel, should be as small as possible, and only perform those functions that are absolutely necessary — all other tasks should be the responsibility of the shell. At the same time as UNIX was being written, the language C was being designed, and in 1973 a UNIX kernel was written using C. C is a high-level language, and as such is machine-independent, so the new (small) kernel and shell could be transferred to a different machine easily. This was found to work well, and Bell Labs was happy to allow the source code for the kernel to be distributed to universities.

In the next few years, work on UNIX was undertaken principally by Bell Labs and by the University of California at Berkeley. These two organisations, however, developed their own versions of UNIX, known respectively as **System V** and **BSD**. Industrial users tended to use System V, whereas BSD UNIX was common in universities and colleges.

By the late 1980s UNIX had been implemented by many manufacturers, each of whom had developed versions which, although based either on System V or on BSD, had their own features. It became apparent that the popularity of UNIX, coupled with the proliferation of ‘dialects’, had resulted in a pressing need for a recognised standard for UNIX to be developed. This was taken on board by the **IEEE** under the name **POSIX**. POSIX consists of a number of interrelated standards. Now part of the **PASC** project, there are more than nine proposed POSIX standards, but not all are yet completed. In this book we only deal with POSIX.2, since the other standards are not necessary for understanding the UNIX shell.

In 1991, a computer science student at the University of Helsinki in Finland, Linus Torvalds, decided to create his own version of UNIX, which he named Linux. It was in 1994 that he released version 1.0 of

NOTE

The kernel is discussed in Chapter 2.1

ACRONYM

BSD = ‘Berkeley System Distribution’

ACRONYM

IEEE = ‘Institute of Electrical and Electronics Engineers, Inc.’

PASC = ‘Portable Application Standards Committee’

Linux. Very quickly it became clear that Torvalds alone would not be able to develop a complete operating system, so he chose to open up his project to allow others to contribute to its development. On the Internet, Torvalds announced his project and called for volunteers to assist; in doing so, the source code was made freely available.

As a result of this model of allowing developers from around the world to contribute to the development of Linux, a Linux community was born, and has now grown to millions of users, numerous different Linux distributions, and over a hundred developers just for the Linux kernel. It is now an effective and successful operating system that competes on many platforms with commercial offerings. The latest version at the time of writing is version 2.4.

1.5 Conventions

Several different fonts are used in this book. **Bold face** is used when names or concepts are first introduced, and occasionally for emphasis. When dialogue with a machine is displayed, **fixed width font** is used for messages that the UNIX system prints, and **(bold) keyboard font** for instructions typed by a user. If a word that would normally appear in such a dialogue appears in the text, **fixed width font** is again used.

For most purposes in the book, the words ‘UNIX’ and ‘Linux’ are interchangeable, and unless otherwise stated use of the word ‘UNIX’ should be understood as meaning ‘UNIX or Linux’.

CHAPTER SUMMARY

- ▶ Modern computer systems are made up of both hardware and software.
- ▶ Hardware comprises processors, and input and output devices.
- ▶ Software can be application programs or system software like operating systems.
- ▶ UNIX and Linux are operating systems with a long academic tradition.

Index

!= (arithmetic expansion) 151
!= (Awk) 219
! 156
" 110
(string length) 147
118
\$ (BRE) 200
\$ (Perl) 241
\$ (prompt) 33
\$ (variable) 93
\$ (Vi) 41, 43
\$((command substitution) 114
\$(114
\$(150
\$* 142
\$0 (Awk) 216
\$0 142
\$1, \$2, *etc* (Awk) 216
\$1, \$2, *etc* 142
\$? 123
\$@ 143
\$[150
\$# 142
\${ 146
% 102
% (bc) 127
% (arithmetic expansion) 151
% (Awk) 219
% (conversion specification) 140
% (Perl) 242
% (prompt) 33
%% 102
& 91
& (Sed) 205
&& (Awk) 219, 221
&& 124
' 94, 110
((command grouping) 125
) 176
) (case) 157
) 150
* (bc) 127
* (tr) 197
* (arithmetic expansion) 151
* (Awk) 219
* (BRE) 200
* (pattern matching) 154
+ (bc) 127
+ (arithmetic expansion) 151
+ (Awk) 219
+ (ERE) 200
+= (Awk) 227
- 13
- (bc) 127
- (Awk) 219
- (BRE) 199
- (option) 34
- (pattern matching) 156
- (standard input) 47
-empty (find) 139
-eq (test) 131
-exec (find) 139
-ge (test) 131
-gid (find) 139
-group (find) 139
-gt (test) 131
-inum (find) 139
-le (test) 131
-links (find) 139
-lt (test) 131
-name (find) 139
-ne (test) 131
-perm (find) 139
-print (find) 138

```

-printf (find) 139
-size (find) 139
-type (find) 139
-user (find) 139
. 13
. (BRE) 200
. (command) 99
. (directory) 60
. (filename) 60
. (Perl) 242
. (vi address) 211
.. (directory) 60
.a (filename suffix) 155, 189
.c (filename suffix) 155
.f (filename suffix) 158
.o (filename suffix) 155
.p (filename suffix) 158
/ (bc) 127
/ (arithmetic expansion) 151
/ (Awk) 219
/ (root directory) 61
/ (Vi) 41, 42
/dev 112
/dev/audio 113
/dev/console 113
/dev/null 113
/dev/rst8 113
/dev/tty 112
/etc/passwd 230
/tmp 169
/usr/dict/words 54
0 (Vi) 43
0< 49
1> 48
2> 48
:+ 147
:- 146
:0 (Vi) 42
:= 146
:? 147
: (Vi) 40
: 126
:$ (Vi) 42
:n (Vi) 42
:] 196
:q (Vi) 43
:q! (Vi) 43
:w (Vi) 43
:wq (Vi) 43
;; 157
< (arithmetic expansion) 151
< (Awk) 219
< (input redirection) 46
< (Perl) 243
<< 51
<= (arithmetic expansion) 151
<= (Awk) 219
= (alias) 177
= (test) 131
= (variable assignment) 93
== (arithmetic expansion) 151
== (Awk) 219
> (arithmetic expansion) 151
> (Awk) 219
> (output redirection) 46
> (Perl) 243
> (prompt) 111
>= (arithmetic expansion) 151
>= (Awk) 219
>> (Perl) 243
>> 49
? (Vi) 42
? (more) 54
? (ERE) 200
? (pattern matching) 154
? (Vi) 41
@ (Perl) 241
[ (test) 129
[ (BRE) 199
[ (pattern matching) 154
[: 196
\ 110
\\ (backslash) 140
\a (alert) 140
\b (backspace) 140
\f (formfeed) 140
\n (newline) 140
\r (carriage return) 140
\t (tab) 140
\v (vertical tab) 140
] (arithmetic expansion) 150
] (test) 129
^ (Awk) 219
^ (bc) 127
^ (BRE) 199
^ (Vi) 41, 42
_ 13
__END__ (Perl) 245

```

- ' 114
- { (csplit) 207
- } (find) 138
- | 51, 121
- | (ERE) 201
- || 124
- || (Awk) 219, 221
- ~ 36
- ~ (directory) 61
- ~v (Vi) 37

A

- a (Vi) 40, 42
- a (bc arctan function) 127
- absolute filename 61
- access privileges 69
- Acrobat 273
- action (Awk) 215
- adduser 261
- administrator account 24
- alias 177
- alias 177
- and-list 124
- Apache 276
- append 49
- application program 7
- ar 185
- archive 83
- argument 34
- arithmetic expansion 150
- array (Awk) 226
- array index (Awk) 226
- ASCII 5, 14, 191
- associative array (Awk) 226
- asynchronous list 120
- at 104
- at queue 105
- AT&T 8
- atan2 (Awk arctan function) 233
- awk 214
- Awk 214

B

- b (more) 54
- b (Vi) 41, 42
- background 90
- backquote 113
- backslash 110
- backup 83
- base 84
- basename 85
- bash 12
- bash 33
- basic regular expression 199
- batch queue 105
- batch 105
- baud 187
- bc 126
- BEL 5
- Bell Labs 8
- BeOS 8
- bg 102
- binding operator (Perl) 251
- bit 14
- block 191
- boot loader 24
- Bourne shell 12
- bracket expression 199
- BRE 199
- break 136
- BSD 9
- buffer 44
- buffered input 44
- byte 14

C

- C (programming language) 9
- C (Vi) 42
- C shell 12
- c (bc cosine function) 127
- case 157
- cat 45
- cc 184
- cd 61
- CDE 25
- central processing unit 2
- character 14

character encoding 5
 character class 196
 character special file 112
 check digit 14
 checksum 183
chgrp 70
 child process 100
chmod 71
chown 71
cksum 183
cmp 76
 collating sequence 195
 colon 126
 colon-mode (Vi) 40
comm 189
 command 120
command 177
 command argument 34
 command grouping 123
 command history list 103
 command option 34
 command substitution 114
 command window 3
 command-mode (Vi) 40
 comment 118
 compilation 88
 compiler 7
 compound command 120
compress 83, 262
 compression 83
 Concurrent Versions System 274
 continuation prompt 111
continue 136
 control center (KDE) 29
 control character 5
 control structure (Perl) 247
 controlling terminal 89
 conversion specification 140
core 137, 174
coredump 174
cos (Awk cosine function) 233
cp 65
cpio 83
 CPM 8
 CPU 2
crontab 106, 262
csh 12
csplit 207
ctags 189

Ctrl key 5
 ctrl-\ 174
 ctrl-C 54, 187
 ctrl-D 33, 45
 ctrl-D (Vi) 41, 42
 ctrl-G 5
 ctrl-U (Vi) 41, 42
 current directory 59
 current message (Mailx) 37
 cursor 3
 customized (installation) 22
cut 78
 CVS 274
cw (Vi) 42

D

D (Vi) 40, 42
DATA (Perl) 245
date 33
dbx 174
dd (Vi) 40, 42
dd 191
 debugging shell scripts 161
 default job 102
 definition 176
 DEL key 44
 delimiter 78
DESCRIPTION (manual page) 56
 desktop (KDE) 26
 desktop manager 4
 development (installation) 22
 device 111
df 63, 260
DIAGNOSTICS (manual page) 56
diff 76, 86, 189
 directory 59, 64
 directory hierarchy 59
dirname 85
 DNS 266
 DOS 8
 DOS.SYS 24
 dotdot 60
 double quote 110
du 85
 dual boot 19
 dumb terminal 3

`dw` (Vi) 42

E

`e` (Vi) 41, 42
`e` (bc exponential function) 127
EBCDIC 16
`echo` 4
`echo` 50, 139
`ed` 43
EDITOR 93
editors 39
`egrep` 201
elapsed time 108
electronic mail 36
Elm 37
Emacs 43, 273
`emacs` 43
email 36, 180
emergencies 54
emulator 19, 21
Enter (key) 5
`env` 94, 98
ENVIRON (Awk) 228
environment 92
ERE 199, 200, 225
`esac` 157
ESC (Vi) 40
escape character 139
escape sequence 81
`eval` 179
`ex` 43
`exec` 178
executable shell script 98
execute permission (file) 69
executing 34
EXIT 175
`exit` 33, 160
exit status 123
`exp` (Awk exponential function) 233
`expand` 189
expert (installation) 22
`export` 95
`expr` 152
extended regular expression 199,

200, 225

F

`false` 126
`fc` 103
FDISK 19, 23
`fdisk` 269
`fg` 102
`fgrep` 201
field 78, 215
field delimiter 78
field separator (Awk) 229
field width 140
fields (Perl) 246
FIFO 188
FIFO file 130
file 13, 73
file access control 68
file group 69
file owner 69
FILENAME (Awk) 223
FILENAME 224
filename 62
filename suffix 155
FILES (manual page) 56
filesystem 63
filter 194
`find` 138, 191
floating point number 218
FNR (Awk) 223, 224
`fold` 78
`for` 134
foreground 90
Framemaker 273
FS (Awk) 223, 229
FTP 277
`ftp` 277
function 175
function (bc) 126
function (Awk) 233
function definition 120, 175, 176
FVWM 25

G

Galeon 276
 GCC 272
 GCL 272
 General Electric 8
`getconf` 186
`getline` (Awk) 234
`getopts` 162
 Ghostscript 273
 Ghostview 273
 GID 68
 gigabyte 59
 GIMP 274
 global variable 95
 GNOME 25
 graphics terminal 3
 Grep 201
`grep` 201
 group 68
 group (file) 69
 group-id 68
 GSView 273
 GUI 20
 gzip 274

H

`h` (Vi) 40
 hang 54
 hangup 107
 hard link 66
 hardware 2
`head` 74
 help 55
 here-document 50
 HOME 93
 home directory 61
 hyphen (standard input) 47

I

`i` (Vi) 40, 42
`id` 68
 idle time 36
 IEEE 9

`if` 132
`include` (directory) 168
 index (array) 226
 inode 62
 input 44
 input device 2
 input mode (Vi) 40
`int` (Awk truncation function) 233
 interpreter 7, 88
 interrupt 54
 IO.SYS 24
 IP-chain 269

J

`j` (Vi) 40
`J` (Vi) 42
 JAPH 254
 Java 272
 job 100
 job control 92, 100
 jobnumber 91
 jobs 101
 join 189

K

`k` (Vi) 40
 KDE 24, 25
 KEdit 44
 kernel 5, 11
 keyboard 3
`kill` 92, 175, 261
 killed process 89
 KOffice 273
 Konqueror 276
 Korn shell 12
`ksh` 12

L

`l` (Vi) 40
`l` (`bc` logarithm function) 127
 LaTeX 273
 LATIN-1 16

- length (of a string) 147
- length** (bc) 127
- length** (Awk) 234
- less** 53
- lib** (directory) 168
- LILO 24
- lines 74
- link 66, 70, 165
- linking (object files) 183
- Linus Torvalds 9
- Linux 9
- linuxconf** 259
- Lisp 272
- list command 120, 124
- list command grouping 125
- ln** 66
- loadlin 20
- locale 192, 195
- locale** 192
- localedef** 192
- local variable 95
- log** (Awk logarithm function) 233
- log in 32
- logger** 192
- logging in 32
- logging out 33
- login 32
- login shell 50, 89
- logname** 58
- LOGNAME** 93
- logout 33
- loop 134
- lp** 81
- lpr** 82
- ls** 39, 62
- Lynx 276

M

- MAC 265
- machine code 7
- MacOS 8
- Mail 37
- mailbox 168
- Mailx 36
- mailx** 36
- make** 184

- Makefile** 185
- makefile** 185
- makefile 184
- man** 55
- manual page 55
- manual volume 55
- match** (Awk) 234
- matching list 199
- mesg** 38
- MIT 8
- mkdir** 64
- mkfifo** 188
- module (Perl) 250
- monitor 3
- more** 53
- mounting 23
- mouse 3
- Mozilla 276
- Multics 8
- Mush 37
- mv** 65, 85

N

- name (array) 226
- NAME** (manual page) 56
- NAME** 145
- named pipe 188
- ncftp** 277
- NEdit 44
- Netscape 276
- network 13
- Network Information Service 230
- newgrp** 68
- Newline 74
- NF** (Awk) 223
- NF** 224
- nice** 107
- NIS** 230
- NL** 5
- nm** 189
- nohup** 107
- nonmatching list 199
- NOTES** (manual page) 56
- NR** (Awk) 223

O

o (Vi) 42
 octal dump 80
 od 80
 OFS (Awk) 232, 223
 open source 16
 OpenOffice 273
 Opera 276
 operating system 7, 11
 OPTARG 163
 OPTIND 164
 OPTIONS (manual page) 56
 options 34, 162
 or-list 124
 ORS (Awk) 223, 232
 other (file) 69
 output 44
 output device 2, 3
 output field separator 232
 output record separator 232
 owner (file) 69

P

pager 53
 PalmOS 8
 panel (KDE) 26
 parameter expansion 144
 parent process 100
 parent (directory) 60
 parity bit 14
 partitioning 20, 23
 PASC 9
 passwd 34, 260
 password 31
 paste 77, 190
 patch 86
 PATH 94, 99
 pathchk 186
 pathname component 94
 pattern (Awk) 215
 pattern binding operator 251
 pattern matching 154
 pattern space (Sed) 203
 pax 83
 Perl 239, 272

perldoc 241
 perm symbol 72
 pg 53
 PHP 272
 PID 89
 Pine 37
 pipe 51
 pipeline 120, 121
 positional parameters 142
 POSIX 9
 PostScript 82
 pr 82
 precedence 127
 prefix (csplit) 207
 prepend 49
 print (Awk) 216
 PRINTER 93
 printf (Awk) 217
 printf 139
 printing character 5
 prioritise 107
 process 88
 process-id 89
 processing time 108
 processor 2, 88
 program 2
 program (Perl) 240
 program control 100
 Prolog 272
 prompt 33
 prosplitter 274
 ps 89, 260
 PS1 93
 PS2 93, 111
 putty 277
 pwd 61

Q

q (more) 54
 Q (Vi) 43
 q (Vi) 43
 q! (Vi) 43
 queue 105

R

- RAM 18
- rand** (Awk random number) 233
- range 195
- RAR 274
- rcp** 277
- RE 199
- read** 96
- readonly** 111
- read permission (file) 69
- real time 108
- recommended (installation) 22
- record 215
- record separator 231
- Red Hat 22
- redirection 45
- regular expression 199
- regular file 129
- relative filename 61
- release 34
- renice** 107
- repetitive strain injury 32
- Return (key) 5
- Return (**more**) 54
- return** 177
- rlogin** 277
- rm** 48
- rmdir** 64
- root 60
- root (account) 24
- RS 231
- RSI 32
- running process 89
- running program 34

S

- s** (**bc** sine function) 127
- scale** (**bc**) 126, 127
- scandisk** 23
- scheduling 92
- scp** 277
- screen 3
- screensaver 32
- script 50, 159
- script (Awk) 215

- script (Grep) 201
- script (Sed) 203
- Sed 203
- SEE ALSO** (manual page) 56
- sequential list 120
- server 22
- set** 161
- sftp** 277
- sh 12
- sh** 50
- shell 5, 11, 12
- SHELL** 93
- shell options 161
- shift** 144
- shutdown** 261
- SIGALRM** 175
- SIGEXIT** 175
- SIGHUP** 107, 175
- SIGINT** 174, 175
- SIGKILL** 92, 175
- signals 92
- SIGQUIT** 174, 175
- SIGTERM** 175
- SIGTTIN** 103
- simple command 120, 121
- sin** (Awk sine function) 233
- single boot 19
- single quote 94, 110
- slash 61
- sleep** 91
- slogin** 277
- soft link 165
- software 2, 4
- solidus 61
- sort** 79
- Space (**more**) 54
- SPARC 2
- split** (Awk) 234
- split** (Perl) 246
- split** 182, 206, 264
- spool (directory) 168
- sqrt** (**bc**) 127
- sqrt** (Awk square root) 233
- ssh** 268
- SSH 277
- standard error 44
- standard input 44
- standard output 44
- StarOffice 273

startx 24
stderr 44
stdin 44
stdout 44
stopped (process) 89
stopped job 101
stopped process 89
stream 4, 44
strings 81
strip 185
stty 187
Stuffit 274
sub (Awk) 234
subdirectory 66
substr (Awk) 234
suffix 84, 155
sum 183
super-user 8
SuSE 22
suspended (process) 89
symbol 189
symbolic link 165
SYNOPSIS (manual page) 56
syntax 118
system 13
system (Awk) 234
system administration 8
system administrator 8
system software 7
system time 108
System V 9

T

TAB 5
TAB key 167, 189
tab position 167
Tab Window Manager 25
tabs 167, 189
tail 74
talk 38
tar 83, 262
target 184
taskbar (KDE) 26
Tcl/Tk 272
tcsh 12
tee 53

Telnet 277
telnet 277
temporary files 169
TERM 93, 95
terminal 35
test 129
TeX 273
text files 74
theme (FVWM) 25
tilde (directory) 61
time 108
time-sharing 89
tmp (directory) 169
tolower (Awk) 234
Tom's Window Manager 25
top 90
touch 81
toupper (Awk) 234
tput 166, 187
tr 196
trap 174
trapping signals 173
trees 60
true 126
tty 35
TWM 25

U

UID 68
umask 72
UMSDOS 20
unalias 178
uname 34, 56
uncompress 84
unexpand 189
Unicode 16
uniq 76, 189
Universal Coordinated Time 34
UNIX 9
unset 145
unset variable 145
until 136
USER 93
user accounts 24
user-id 68
useradd 261

`userdel` 261
`username` 31, 35
UTC 34
utility 120
`uudecode` 180
`uuencode` 180

V

value 93
`var` (directory) 168
variable 92
VDU 3
version 34
Vi 39, 210
vi 39
`vilearn` 42
Vim 44
virtual computer 21
Virtual Network Computing 274
VISUAL 93
visual display unit 3
VMS 8
VMware 21, 274
VNC 274
VTWM 25

W

w (Vi) 41, 42, 43
`wait` 109
`wc` 75
`while` 135
whitespace 167
`who` 35, 55
who symbol 72
window 3, 35
window manager 4, 25
Windows 8
WINE 21, 274
`wine` 22
word 14
workstation 3
workstation (installation) 22
wq (Vi) 43
`write` 37

write permission (file) 69
`wvdial` 267

X

x (Vi) 40, 42
`xargs` 191
Xmail 37
XMMS 274
X MultiMedia System 274
X windows 24
XV 274

Y

`ypcat` 231
`ypmatch` 266

Z

`zsh` 12
Z shell 12
ZZ (Vi) 40, 42