Contents

CHAPTER 1

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.

ACRONYM

AMD = 'Advanced Micro Devices, Inc.'
SPARC = 'Scalable Processor ARChitecture'

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.

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.

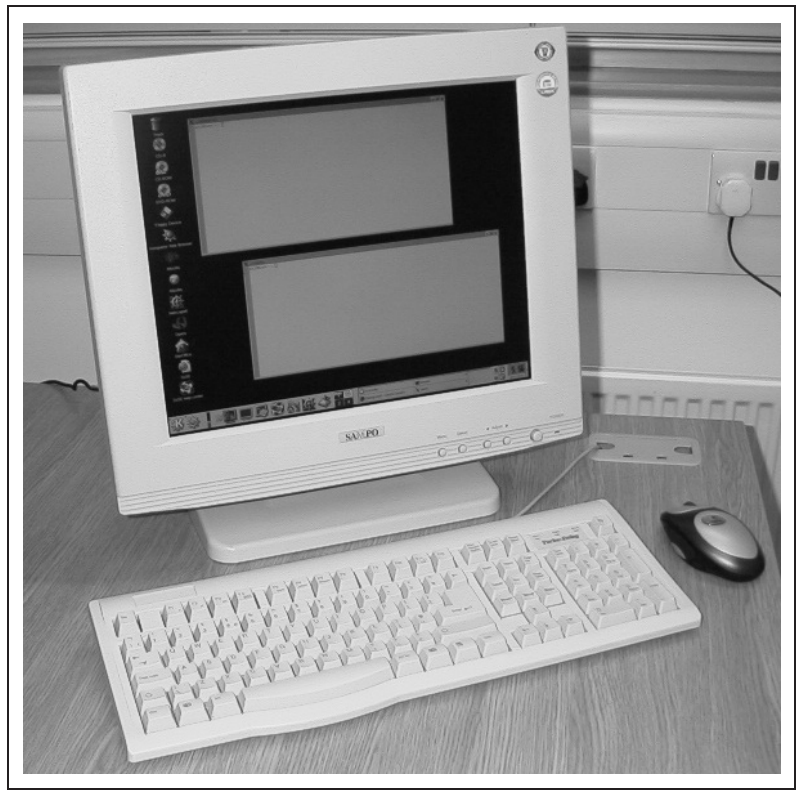

Figure 1.1 A typical computer, with screen, keyboard and mouse

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').

ACRONYM

ASCII = 'American Standard Code for Information Interchange'

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.

NOTE

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

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 →, 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 ↩) is pressed. The escape character, which would not normally be

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

ACRONYM

MIT = 'Massachusetts Institute of Technology'

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.

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.

NOTE

The kernel is discussed in Chapter 2.1

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.

ACRONYM

BSD = 'Berkeley System Distribution'

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.

ACRONYM

IEEE = 'Institute of Electrical and Electronics Engineers, Inc.'
PASC = 'Portable Application Standards Committee'

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

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

A

B

C

D

E

F

M

N

O

P

Q

R

S

T

U

V

W

X

Y

Z