1

INTRODUCTION

Every computer needs an operating system to manage its memory, control its I/O devices, implement its file system and provide an interface to its users. Many operating systems exist, such as MS-DOS, OS/2, and UNIX. This manual describes yet another operating system called MINIX. Although MINIX is entirely new, it was inspired by UNIX, and has many features in common with UNIX. For this reason it is fitting that we begin this introduction with a brief history of UNIX, and its ancestors. This will be followed by an equally brief history of MINIX. Finally, the chapter concludes with a summary of the rest of the manual.

1.1. HISTORY OF UNIX

Prior to 1950, all computers were personal computers. At least in the sense that only one person could use a computer at a time. Only research laboratories owned computers in those days. The usual method of operation was for the programmer to sign up for an hour of machine time on a sign-up sheet posted on the bulletin board. When his time came, the programmer chased everybody else out of the machine room and went to work.

During the 1950s, **batch systems** were invented. With a batch system, the programmers punched their programs onto 80-column cards, and deposited them in a tray in the computer room. Every half hour or so, the computer operator would go

to the tray, pick up a batch of jobs, and read them in. The printed output was brought back later for the programmers to collect at their leisure.

While the batch system made more efficient use of the computer, it was not wildly popular with programmers, who often lost hours due to a single typing error that caused their compilations to fail. In the early 1960s, researchers at Dartmouth College and M.I.T. devised **timesharing systems**, in which many users could log into the same computer at once from remote terminals. The Dartmouth project led to the development of BASIC; the M.I.T. project was the great grandfather of MINIX.

The M.I.T. system, called **CTSS** (**Compatible Time Sharing System**) because it was more-or-less compatible with the batch system then in use at M.I.T., was a success beyond the wildest expectations of its designers. They quickly decided to build a new and much more powerful timesharing system on what was then one of the largest computers in the world, the GE 645, a machine about as fast as a modern PC/AT. This project, called **MULTICS** (**MULTiplexed Information and Computing Service**) was a joint effort of M.I.T., General Electric (then a computer vendor), and AT&T Bell Labs.

To make a long story short, the project was so ambitious that the system was difficult to program. Bell Labs eventually pulled out and GE sold its computer business to Honeywell. M.I.T. and Honeywell went on to complete MULTICS, and it ran at many installations for 25 years. In a way, it continues to live, since OS/2 has many ideas and features taken straight from MULTICS .

Meanwhile, one of the Bell Labs researchers who had worked on MULTICS, Ken Thompson, was hunting around for something interesting to do next. He spied an old PDP-7 minicomputer at Bell Labs that nobody was using, and decided to implement a stripped down version MULTICS on his own on this tiny minicomputer. The system he produced, while clearly not full MULTICS, did work and supported one user (Thompson). One of the other people at the Labs, Brian Kernighan, somewhat jokingly called it **UNICS** (**UNiplexed Information and Computing Service**). Bell Labs management was so impressed that they bought Thompson a more modern minicomputer, a PDP-11, to continue his work.

The initial implementation was in PDP-7 assembly code. When this had to be completely rewritten for the PDP-11, it became clear that it would be much better to write the system in a high level language to make it portable. At this point, another Bell Labs Researcher, Dennis Ritchie, designed and implemented a new language called **C** in which the two of them rewrote the system, whose spelling had now changed to UNIX.

In 1974, Ritchie and Thompson wrote a now-classic paper about UNIX that attracted a great deal of attention. Many universities asked them if they could have a copy of the system, including all the source code. AT&T agreed. Within a few years, UNIX had become established as the de facto standard in hundreds of university departments around the world. The source code was widely available, and often studied in university courses. International meetings were organized, in which

speakers would get up and describe how they had modified some system routine to be a bit fancier. UNIX had by now become something of a cult item, with numerous fanatically devoted followers.

As UNIX kept spreading, AT&T began to see how valuable it was and began restricting access to the source code of new versions, starting with the Seventh Edition (usually referred to as **V7**). It was no longer permitted for universities to teach courses using the source code as an example, and public discussions of the internal workings of the code were severely restricted.

Nevertheless, UNIX' fame continued to spread. The University of California at Berkeley got a contract to port V7 it to the VAX, adding virtual memory and numerous other features in the process. This work led to **4.x BSD**. AT&T itself modified V7 extensively, leading to **System V**. It took a decade before these two versions could be partially reconciled, under the auspices of the IEEE, which led to the **POSIX** standard.

Although UNIX was now more popular, without the source code, it was also a lot less fun, especially for the people whose initial enthusiasm had made it a big success. The time was ripe for a new system.

1.2. HISTORY OF MINIX

Many university professors regretted that UNIX could no longer be taught in operating systems courses, but there appeared to be no choice. One of them, Andrew Tanenbaum, at the Vrije Universiteit in Amsterdam, The Netherlands, went out and bought an IBM PC, and like Ken Thompson a decade earlier, set out to write a new operating system from scratch. Just as Thompson was inspired by MULTICS, but ultimately wrote a new and much smaller operating system, Tanenbaum was inspired by UNIX, but ultimately also wrote a new and much smaller operating system—MINIX—which stands for Mini-UNIX.

Since MINIX contains no AT&T code whatsoever, it falls outside the AT&T licensing restrictions. The source code has been made widely available to universities for study in courses and otherwise. Like UNIX, MINIX quickly acquired an enthusiastic following and began to occupy the same niche that UNIX had filled in its early days—a small operating system available with all the source code that people could study and modify as they wished.

Within a month of its release (Jan. 1987), there was already so much interest in MINIX worldwide, that a news group (comp.os.minix) was set up on USENET, a computer network accessible to most universities and computer companies in North America, Europe, Japan, Australia, and elsewhere. A few months later, the news group had over 10,000 people reading and contributing to it.

The initial release of MINIX was for the IBM PC and PC/AT only. It did not take long before people with other kinds of computers began thinking about porting it to their machines. The first port was to a 68000-based machine, the Atari ST, done

primarily by Johan Stevenson, with assistance from Jost Müller The hard part was making MINIX, which, like UNIX, allows multiple processes to run simultaneously, run on a bare 68000, with no memory management or relocation hardware. Once this problem had been solved and new I/O device drivers were written for the Atari's keyboard, screen, disk, etc., MINIX-ST became a reality. The 1.5 Atari version was prepared by Frans Meulenbroeks.

The port to the Commodore Amiga was done by Raymond Michiels and Steven Reiz. The Macintosh port was done by Joseph Pickert. Unlike all the other ports, the Macintosh version does not replace the manufacturer's operating system and run on the bare machine. Instead, it runs on top of the Macintosh operating system, allowing the facilities of both systems to be used simultaneously. Of course, there is a price to to paid, in terms of both size and performance.

The contribution of USENET cannot be underestimated in the MINIX development. Hundreds of individuals have donated ideas, bug fixes, and software, many of which are included in this release. One person stands out above all the others though, Bruce Evans, who has produced improvements too numerous to count, and all of them in a very professional way. The authors are credited in the code for their contributions, where that is technically feasible.

MINIX is still a vigorous on-going development, with new software, ports to new machines, and many other activities in progress. In this respect, MINIX is very different from most software products. The usual model is that a company gets an idea for a product, writes the software, and then sells executable binaries for customers to use. Users are not able to modify the program, and requests for the source code are rejected out of hand. Worldwide public discussion of the system internals is not welcome.

MINIX, in contrast takes a more open approach. The source code for the entire operating system is included in the basic software package, and users are encouraged to tailor the system to their own specific needs. Thousands of people on USENET have done just that. The internal workings of the system are described in detail in the following book:

Title: Operating Systems: Design and Implementation
Author: Andrew S. Tanenbaum
Publisher: Prentice Hall
ISBN: 0-13-637406-9

However, please note that the book describes a somewhat earlier version of the system. Nevertheless, the basic principles are still intact in this version.

MINIX was originally written to be system call compatible with V7 UNIX, the last of the small UNICES. This is the version described in the book.

Future versions of MINIX will migrate towards at least partial conformance with the ANSI C and IEEE POSIX standards. Complete conformance is unlikely, as conformant systems are necessarily so huge that no one person can possibly understand them. Making MINIX fully conformant would defeat the goal of having a system that

is small enough that people can actually understand it. It would also require much larger and more expensive hardware than is currently the case. MINIX 1.5 is an intermediate form: it still supports the V7 system calls and has a Kernighan and Ritchie C compiler, but virtually all the individual files are ANSI C conformant (and also K&R conformant). Furthermore, all of the header files provided in */usr/include* conform to both the ANSI and POSIX standards. This makes porting programs from conformant systems to MINIX 1.5 and vice versa much easier.

1.3. STRUCTURE OF THIS MANUAL

Chapters 2 through 6 tell how to install MINIX on the IBM PC family, Atari, Amiga, Macintosh, and SPARCSTATION 1, respectively. You should read the one appropriate for your machine. Then skip to Chap. 7 to learn more about how to use MINIX.

For people interested in modifying the operating system itself, Chap. 8 has been provided. It discusses things needed by people who want to recompile the system. If you do not plan to do this, you may safely skip this chapter for the time being.

Chapter 9 contains the manual pages for the commands that come with MINIX. Each entry describes one utility program (or sometimes several closely related ones). Every MINIX user, even though intimately familiar with UNIX, should read Chap. 9 very carefully. Some of the commands have extended manual pages. These are present in Chap. 10.

Chapter 11 discusses the MINIX system calls. The treatment here is brief, since most of the system calls should be familiar to UNIX users.

Chapter 12 contains information about networking in MINIX.

Appendix A contains a (nearly complete) listing of the MINIX 1.5 operating system code. The kernel listing is for the Intel (IBM) line, but the file system and memory manager are identical for all versions, as is the general structure of the kernel. Parts of the kernel, especially the I/O device drivers, are different for each version. Most of these machine-dependent parts (mostly deice drivers) have not been listed here, but since the sources are present on the disks, you can easily make your own listing of the missing pieces. The MINIX program *mref* can be used, for example. Appendix B is a cross reference map of the source code listing.

One final note. There is a large MINIX user group on USENET with over 16,000 people. It is called *comp.os.minix*. Thousands of messages have been posted to this group dealing with bugs, improvements, suggestions, and new software. Unfortunately, USENET is not a public network, so one cannot just join, but over a million people are on it, mostly at universities and companies in the computer industry in the U.S., Canada, Europe, Japan, Australia, and other countries. If you are interested in following all the latest developments on the MINIX front, and there are many in the works (e.g., ANSI and POSIX conformance), it is worthwhile trying to find someone who has access.