The Operating System Handbook

or, Fake Your Way Through Minis and Mainframes

by Bob DuCharme

Introduction

Table of Contents

This Section, the Rest of the Book

This is one part of the book "Operating Systems Handbook (or, Fake Your Way Through Minis and Mainframes)," which was originally published by McGraw-Hill as a $49.60 hardcover. Once they reverted the rights to me after it went out of print, I converted the original XyWrite files to DocBook XML and then used Norm Walsh's stylesheets (see www.nwalsh.com) and the Apache FOP program (see xml.apache.org) to convert that to Acrobat files. The six parts of the book are all available at http://www.snee.com/bob/opsys.html:

- Part 1: Introduction. Note that this part includes a new section explaining why I didn't update the book. I strongly suggest that, no matter how much or how little of the book you can use, you glance through the whole Introduction as well. The "Comments and Suggestions" part is now obsolete; my home page is now http:/www.snee.com/bob and my e-mail address is bob@snee.com.
- Part 2: UNIX. Everything described here should apply to Linux and its relatives.
- Part 3: OpenVMS. Basically, VMS. DEC was calling it "OpenVMS" at the time.
- Part 4: OS/400. The operating system for IBM's AS/400.
- Part 5: VM/CMS. An IBM mainframe operating system.
- Part 6: MVS. Another IBM mainframe operating system.

See www.snee.com/bob for information on books I've written since. In reverse chronological order, they are:

- *XSLT Quickly* is a tutorial and users guide to XSLT designed to get you writing stylesheets as quickly as possible.
- *XML: The Annotated Specification* is a copy of the official W3C XML specification with examples, terminology, and explanations of any SGML and computer science concepts necessary for a complete understanding of the XML spec.
- *SGML CD* is a users guide to free SGML software, most of which can be used with XML as well. The chapter on Emacs and PSGML, which requires no previous knowledge of Emacs, is available on the web site in English, Russian, and Polish.

One more thing: either I couldn't figure out the XSL `keep-together` property or version 0.18.1 of FOP doesn't implement it yet. Either way I apologize that some screen shots get split across page breaks.

A 2001 Preface to a 1994 Book

McGraw-Hill published this book seven years ago, and most of the writing took place eight years ago. I wanted to call this crash course in UNIX, VMS, OS/400, VM and VMS "Fake Your Way Through Minis and Mainframes," but they wanted something more professional-sounding for their Professional Book Division.

The book sold a few thousand copies, and I received several e-mails thanking me for writing it. Once the book went out of print, I had McGraw-Hill revert the rights back to me so that I could legally give away a free Acrobat version of the book to whoever wanted it.

So here it is. I wrote a perl script to convert the original XyWrite files to DocBook XML and then used Norm Walsh's stylesheets and the Apache Project's FOP to create the Apache files. I didn't try to update the content, because I currently don't have access to machines running most of these operating systems and because I would have put it off too long anyway. Of course, parts of the book will look dated—Linux hadn't reached release 1.0 yet when I wrote the Unix chapter and the web was small enough to account for only 1% of Internet traffic. Just as people now often refer to the Internet as "the web" because web browsing is the most popular use of the Internet, in 1993 people referred to it as Usenet, because the discussion newsgroups whose most popular web incarnation were on DejaNews (later bought by the Google search engine folks) went by that name and was the most popular use of the Internet at the time. Chapter 2 mentions Usenet several times.

Enjoy!

Bob DuCharme

http://www.snee.com/bob

Acknowledgements

My thanks to Howard Lune, who explained to me the relationship between VM, CMS, and CP as we killed a bottle of Dewar's at 2 in the morning in a Miami hotel room, and who has since graciously reviewed the mainframe chapters; to Don Bonnice, whom I have never met, but who helped me with the OS/400 chapters through the miracle of e-mail; to Ray Hood, who first taught me UNIX, and, more importantly, tuaght me how to quickly identify and learn the important parts of a software system; to Alex Berson, for taking the time to review the manuscript; to Chet Ensign and Frances Gambino, who taught me to put a book together; th Madeline—may file allocation be as distant a memory to her generation as magnetic core memory is to ours; and most of all, to my wife Jennifer, who has patiently learned more about minis, mainframes, Elvis Presley, Formula 1, Indy cars, and electric guitars than she ever planned to before she met me.

1 Introduction

Today it's fashionable among many personal computer users today to proclaim that minicomputers and mainframes are dinosaurs, and that the meteor that renders them extinct is coming fast. (When I say personal computer, I mean one that's personal, that you have all to yourself, and that you can afford to have at home or on your desk at work. This includes Macintoshes, Amigas, and the Atari ST, not just DOS machines.) They call the big machines primitive, because you don't use a mouse and icons to start up programs.

These personal computer users probably feel frustrated when they look through the want ads and see the many job openings for people who know UNIX, OpenVMS, OS/400, VM/CMS, and MVS. They don't understand that the big rigs aren't dying out; their roles are being redefined to take advantage of their strengths while personal computers take over the jobs that personal computers can do better.

The personal computer's mouse and icons mean that its interface is easier to use and more responsive. Unfortunately, personal computer snobs judge other operating systems by their interfaces. They see the on-line help in MVS scrolling up the screen in all capital letters and snort, "How very primitive!" They don't realize how many sophisticated features, unrelated to the user interface, have always been integral parts of MVS and other large operating systems—features that many in the personal computer business are only now trying to shoehorn into their products. Data safety, proper multitasking, and serious multiuser support still have a way to go for networks of personal computers. And mini and mainframe technology is not standing still; a glance at trade papers shows that they become faster and more powerful every year.

1.1 Why Should You Learn How to Use Minis and Mainframes?

For many, the want ads mentioned above provide sufficient impetus to learn about larger systems. The line "Working knowledge of UNIX, OpenVMS, OS/400, VM/CMS, and MVS" looks great on a resume.

The recent trend toward downsizing means that some of the development jobs for your favorite operating system involve moving (or "migrating") existing applications from larger systems to smaller ones. Many jobs involve moving only part of an application to a different system. Another big trend is distributed, "client/server" systems, in which the application's user interface runs on the smaller system and its data is stored on a larger system, and the two (or more) systems must communicate and cooperate. If you want any of these jobs, you better know the systems at both ends of the job.

An AS/400 running release V2R2 or later of the OS/400 operating system can store a database of up to 2 billion records taking up 248 gigabytes of space. (And remember—that's a minicomputer, not a mainframe!) It will be quite a while before any personal computer-based system can handle a database that large. The practicality of storing larger databases on larger systems brings up another reason for learning how to use minis and mainframes: just as Willie Sutton said that he

robbed banks "because that's where the money is," it's a good idea to get comfortable with large computer systems because that's where the really massive databases are stored. Access to the big systems means access to more information.

Actually, the best reason for learning these systems is this: they're really not that difficult. When people discuss the relative merits of different operating systems, they talk about the advanced features. That's where the difficult parts come in; when it comes to the basics, the tasks that are necessary to get by are remarkably similar from one operating system to another.

BUZZWORD *Downsize* Minicomputers have provided an alternative to mainframes since 1961 and personal computers have provided an alternative to both since 1977. There have always been applications that were too big for personal computers, and many that were too big for minis as well. As all classes of computers become more powerful by an estimated 20% a year, minicomputers and personal computers can more easily handle applications that were formerly considered too big for them.

The process of moving an application from a mini to a personal computer (or to a networked group of personal computers) or from a mainframe to either is known as "downsizing." Some development environments boast of their availability on multiple platforms, making downsizing easier—in other words, if you know how to use the language or development program that was used to create an application, and it's also available on the target system, then you're halfway there. Still, you must be familiar enough with the basic operating system commands on both ends of the project to be able to log in and to read, copy, edit, and transfer files.

A recent variation on "downsizing" is "rightsizing." It essentially means the same thing, but implies that you're considering options besides moving from a bigger system to a smaller system. Its main advantage over the word "downsizing" is that it is more recent, and therefore sounds more up-to-date—always a big plus with buzzwords.

Show me someone who insists that a certain operating system is superior to all others and I'll show you someone who probably only knows one operating system. (Or else a DOS user who has just learned UNIX—sometimes the added power goes to their heads and they forget why personal computers were invented.) Someone once called it the "baby duck syndrome"—these users behave like baby ducks who think that the first thing they see after being born is their mother.

Sometimes, it's fun to be a snob. It's even better, though, to have a broader background and wider perspective than one particular subset of computer nerds. Once you can log in, create and manipulate files, navigate the file system, and send and receive mail on many different operating systems, you gain a perspective on the strengths and weaknesses of each—a perspective that helps you to appreciate these operating systems individually as well as the roles each can play when they must work together. And, people are impressed by someone who appears comfortable with several large systems. It doesn't matter if you only know how to do ten things on each one, as long as they're the right ten things!

1.1.1 What This Book Assumes That You Know

This book is not a beginner's introduction to computers. Although it sticks to basic topics, it contains a series of crash courses, so it moves quickly through these topics.

Presumably, you already know what an operating system is: the supervisory program that runs on a computer at all times, taking your instructions to run other programs or to manipulate and print files, and carrying out these instructions while it coordinates your actions with those of other users on the system. You know what a file is, and you know that a computer that can store thousands of files needs a way to organize them, so that learning a little about the file system is one of the first steps in learning to use a particular operating system. You also know the basic operations that people do with text editors and word processors: creating new files, adding text to them, deleting and editing text, and saving or aborting the edits made in a given session.

Ideally, you should have some experience with a command-line driven operating system—the kind where you type commands and press Enter (or Return) to get results. Some operating systems, like AmigaDOS and the UNIX found on Sun workstations, offer a mouse-driven graphical user interface to handle files and applications, but also offer a window with a command line. They do this because, contrary to the insistence of Macintosh purists, typing out commands is often a more efficient way to accomplish things. For example, typing the word "erase" followed by a filename and then pressing Enter is a simpler—and, yes, more intuitive—way to erase a file than pushing around a hunk of plastic with a rubber ball inside of it.

As this book shows you, mini and mainframe operating systems either erase files with this command or with a slight variation, like using the word "delete" instead of "erase." Some commands are more complicated than this, but remember, learning the basics of an operating system does not require memorizing complicated commands! The keys are remembering simple commands and remembering how to find out the ways to put together more complicated ones when necessary.

For someone only interested in the basics of using an operating system, the easy-to-remember way to accomplish something is always more important than the most efficient way. If you already know one of the operating systems covered in this book, you will find that it doesn't always explain the most efficient techniques. Again, this is because of the crash course approach. Learning the efficient way to do something usually involves learning why it's more efficient; this undoubtedly takes you into areas that the dabbler would rather avoid.

BUZZWORD *Client/Server* This is actually two words, but because it describes a particular relationship between two or more computers, the two are often used together. In a client/server system, smaller computers are hooked up to larger systems, and developers create applications that take advantage of the combination.

Usually, the smaller computer (the "client") presents the user interface and formulates a query or command based on the user's actions. It then sends this query or command to the larger computer, or "server," which carries out the smaller computer's request and returns the results—often a specific subset of the data stored there—to the smaller computer. The user has the advantage of the small computer's better interface, but the data is stored on a system with a large storage capacity, better multi-user support, and built-in safeguards against possible problems ranging from power outages to attempted break-ins by hackers.

In a broader sense, a server is something that provides a service to a client system. It may refer to hardware, as with a file server that stores files for other computers to use. It may refer to software; a database server is a program that mediates requests for data from clients and ensures the integrity of the stored data.

Along with downsizing, increased development of client/server systems is another important reason that mainframe, minicomputer, and microcomputer people must learn more about each others' systems. This way, they can make these different computers work together as efficiently as possible.

1.2 Minicomputers

In 1961, the Digital Equipment Corporation introduced the "Programmed Data Processor 1." The PDP-1 is generally considered to be the first commercially available minicomputer. It was a scaled-down, less expensive (only $120,000!) version of the multi-million dollar behemoths that were synonymous with the word "computer" at the time. To distinguish it from the bigger computers, they called it a "minicomputer." Before the invention of the minicomputer, the word "mainframe" was not even necessary, because people just called them "computers." (Actually, some people called them "IBMs," but that's another story.)

The PDP-1's various successors, especially the $20,000 PDP-8 that DEC introduced in 1965, proved that there was a real market for minicomputers. Other computer makers soon entered the minicomputer market—Hewlett Packard, Wang, Data General, even IBM.

The word "minicomputer," coined in the same era as the word "miniskirt," has become anachronistic. Nowadays, when someone says "computer," they're usually talking about a personal computer or "microcomputer"—something that costs a few thousand dollars and sits on a desk. A "minicomputer" costs five or six figures and can be as large as a refrigerator.

Considering that modern personal computers are much more powerful than the original minicomputers, and that modern minis are much more powerful than the mainframes available when the PDP-1 was first released, what does that make a minicomputer today?

People often use the term "departmental computing" in the same breath as the word "minicomputer." This gives a good clue about the role of minis. If all the people in one department share one computer, it will be bigger than a personal computer, and smaller than a mainframe. It supports from 10 to 100 people. Maintaining it shouldn't be a full-time job for one person. Of course, the use of minicomputers isn't limited to individual departments of large companies or universities; small companies provide one of the minicomputer's biggest markets.

Of the operating systems described in this book, the AS/400's cleverly named OS/400 operating system is the only one that is unequivocally a minicomputer operating system. Because the smallest VAXes sometimes qualify as personal computers, and the largest ones as mainframes, and because versions of UNIX have been developed for everything from IBM PC/ATs to Cray supercomputers, devotees of these two operating systems insist that their versatility ranges beyond the middle of the computing spectrum designated by the word "minicomputer." However, since the majority of the computers running these two operating systems fit the profile of a mini described above, most people think of OpenVMS and UNIX as minicomputer operating systems.

1.3 Mainframes

Reports of the death of mainframes have been greatly exaggerated. Mainframes can store huge

amounts of data and accomplish a lot of work. If the personal computer of your dreams can be compared to an eight-cylinder Ferrari, then a mainframe is an eighteen-wheel truck. They are the big rigs. Some personal computer snobs claim that a properly networked configuration of personal computers can accomplish anything a mainframe can; these people have no idea of the kinds of things that banks, insurance companies, and governments call on mainframes to do: to process *huge* amounts of data, 24 hours a day, in an extremely secure environment.

BUZZWORD *Big iron* Refers to mainframe hardware. For example, "The insurance company's main office had one Amdahl, but most of the big iron was IBM."

That fact that many people who've never used mainframes consider them antiquated has hurt their reputation. In order to appear up-to-date, more people try to avoid using the word "mainframe." In late 1990, an important trade magazine called "Mainframe Journal" changed its name to "Enterprise Systems Journal." The use of the buzzword "enterprise" reflects more than a desire to look current: it shows a change in attitude about mainframes and their role in business computing today.

BUZZWORD *Enterprise* Usually used as part of an adjective, as in "Enterprise-Wide Computing." A fancy way to say "the whole company." This sort of phrase comes up more and more as people try to hook up all the computers in a given company into one cooperative system.

Instead of being The Computer, which is what a mainframe represented at most companies for years, people now consider it to be one of the resources in a computing environment. There are

plenty of former mainframe tasks that that can now be done better and more cheaply on minicomputers, or even on personal computers, but many jobs remain that are more suited to mainframes. Since so many computers are hooked up to other computers, the best possible system is one that distributes the jobs so that each computer does what it's best at. Of course, someone sitting at a terminal—the company president, a secretary, or a consultant pretending to know the system—shouldn't have to worry about which computer is performing which task as long as this user gets what he or she wants.

DOS on a Mainframe?

Computers did not always use disks to store information. When the disk drive was invented, it was considered such a breakthrough that virtually all computers today have at least one disk built in. In 1966, when personal computers were only a dream, IBM released the first operating system that took advantage of disk storage and called it the Disk Operating System, or DOS. It went through several versions with names like DOS-2314, DOS MP, DOS/VS, and DOS/VSE. This last one—Disk Operating System/Virtual Storage Extended—is still used on some smaller mainframes today, although its popularity waned in the mid-seventies with the rise of VM/CMS and MVS.

When IBM released a personal computer operating system that used disks, they called it PC/DOS to distinguish it from the mainframe DOS. Microsoft called their version of PC/DOS "MS-DOS," for "Microsoft Disk Operating System." PC/DOS and MS-DOS are similar enough that people usually don't bother to distinguish between the two; they refer to both as "DOS." Because DOS VSE is the most common version of the mainframe DOS that anyone still uses, people usually refer to the mainframe DOS as "DOS/VSE" to distinguish it from the PC operating system.

Actually, they refer to it less and less with each passing year, because its place in history as a predecessor to MVS and CMS makes it increasingly archaic. What happens to a DOS/VSE installation that's ready to move on? They don't always replace DOS/VSE with a more up-to-date mainframe system. It tends to be used on smaller System/370s, so the increasing power of the AS/400 and its comparative simplicity in use and maintenance make it a popular replacement for DOS/VSE systems.

A system in which personal computers and mainframes work together distributes the jobs that each computer does best. For personal computers, this means the interface, or "front end": windows, colors, fonts, and the use of a mouse. For mainframes, this means coordinating vast amounts of data that many other users might be trying to use simultaneously. (This role has been cleverly named the "back end" of such a cooperative system.) After the person sitting at the personal computer uses the flashy interface to describe the needed data, the personal computer sends a request to the mainframe, which does the sorting and manipulation necessary to pull out the requested data and send it back to the personal computer.

Because of this changing role, organizations who develop mainframe systems no longer bother with attempts to make fancy interfaces to their systems. They realize that the mainframe version of a fancy interface, when used with typical mainframe terminals, pales in comparison to something as common and inexpensive as a hand-held video game. Instead, they concentrate on making the huge data manipulation power of mainframes available to other computers that are connected to mainframes, so that mainframes can play an efficient role in the increasingly popular "distributed systems" made up of various computers connected together.

IBM's 360 Series of Mainframes

In the early days of computers, the question of compatibility, even among machines from the same company, was moot. Different computers were designed to specialize in handling either characters, integers, or decimal numbers. Of the six different computer models offered by IBM in the early 1960's, no two could run the same programs.

It didn't occur to anyone that a program written on one computer should be able to run on another computer. This changed when people started getting new computers and realized how much work would be required to rewrite the programs to work on their new computers.

IBM saw that it would be easier to sell software if it could offer a wide range of hardware to run that software. In 1964, IBM announced the System/360 series. The number came from the number of degrees in a circle, and was supposed to symbolize the computer's ability to be all things to all people. Even better, the different models were compatible, so that a program that ran on one could run unaltered on a more powerful model in the series.

> Eight years later, after introducing VM/CMS, IBM brought out the 370 series. Although there have been many upgrades and improvements, just about all IBM mainframes used today are part of the 370 series.

To sum up, the role of mainframes is changing and evolving faster than it is shrinking. After all, the guys who designed these things came from the generation of engineers who put men on the moon. Give them a little credit.

> **BUZZWORD** *DASD* Because it does have a specific technical meaning, it may not count as an official buzzword—it sounds so impressive, though, when you say "dazzdee" instead of "mainframe hard disk." It stands for "Direct Access Storage Device." (By the way, "device" is practically its own buzzword, meaning "hardware thing.") A DASD unit is really a stack of hard disks, but you can think of it as one, since it functions as a single unit of storage.

1.4 Getting to Know an Operating System

Because the most basic, necessary tasks on any operating system are pretty much the same, each part of this book has a similar outline. Each covers the following subjects:

- History.
- Starting up.

- Filename rules.
- The file system.
- Important commands for dealing with files.
- On-line help.
- Using the text editor.
- Printing.
- Command files.
- How to send and receive electronic mail.
- A sample session.

1.4.1 History and Culture

Knowing an operating system's history is not particularly important to becoming a comfortable user of that system. Usually, only the experts who have used the system for years seem to know or even care about its origins and development. This is precisely why it's great to know a little of the history: if you want to pretend that you're an expert, it's much easier to learn where an operating system came from and why it became popular than to memorize the syntax and usage of dozens of commands. People discussing UNIX at a party or at work will be very impressed when you casually say, "Of course, the fact that the original UNIX license agreement included the source code while excluding any technical support was a complete reversal of standard practice at the time, and a key factor in the eventual priest-like status of the important UNIX gurus." On the other hand, if you know every little switch to the UNIX `ls` command and can make the filenames list out backwards, forwards, and sideways, they'll just think you're a computer nerd.

The culture, or general attitude of an operating system's typical heavy users, is closely tied to the system's strengths and weaknesses. Familiarity with it helps you fake at least a nominal membership in these cultish groups. It certainly helps you communicate with the real devotees, which is crucial to moving beyond beginner status.

1.4.2 Starting Up: Getting to Use the System

When you turn on a personal computer, unless some special security hardware or software has been added, you can use it right away. Multi-user systems have some measure of security to prevent the wrong people from using them. The first concern for you, as a user of one of these systems, is to establish that you are a legitimate user of that system by logging in or logging on.

This generally involves entering your user ID—a name assigned to you as a user of the system—and a password that you theoretically keep secret so that only you can use that user ID.

Logging In/On/Out/Off

Which is more proper, saying that you log in to a computer, then log out, or log on, then log off? The original IBM way was to say "log on." However, when you connect to a UNIX or OpenVMS system, it asks you to "log in," and it's improper to say that you log on to one of these systems. It's interesting to note that AIX, IBM's version of UNIX, asks you to "log in," which acknowledges the influence of the world of UNIX on IBM. Novell networks of personal computers also ask you to "log in." On the other hand, IBM's AS/400 minicomputer skips the whole question by asking you to "sign on."

Before I remotely connect to my UNIX ID on a Sun workstation ID or to my mainframe VM/CMS account, I must first log in to a DEC Server that can route my request to any of several computers. In other words, I must log *in* to the DEC Server before I log *in* to a Sun workstation, but I must log *in* to the DEC Server before I log *on* to an IBM mainframe.

1.4.3 Filenames

You and the operating system distinguish one file from another by its name. But while `joememo.txt` and `JOEMEMO.TXT` would both refer to the same file in CMS or OpenVMS, they would refer to two different files in UNIX, because UNIX is very case-sensitive. (Like many aspects of UNIX, this is considered both an advantage and a disadvantage, depending on who you ask.)

As you'll see in section 1.5, "General Advice," we can apply certain guidelines to the naming of files on all computers, but you must learn the basic idiosyncrasies of file naming rules on a particular system before you create or rename files on it. How many parts does a filename have, and what are their names? What does each part do? How long can each part be? Are there any keyboard characters that are taboo in filenames? What tricks are there for dealing with more than one file at a time?

1.4.4 How Files Are Organized

Another crucial aspect of an operating system is the method it uses to organize files. When you issue the command to list filenames, you don't want to see thousands of them; there should be some way to categorize them into groups, just as you can categorize the files in a file cabinet according to project or client. Also, on a multi-user system, you will probably be assigned your own storage area in which to keep your files. This brings up several new questions: what is the relationship between your files, those of other users, and the program files that make up the system software? Is there a way to check out what's stored outside of your own area?

The file organization system is one of the most important features that distinguish one operating system from another. Until you learn it, you can't get around on a computer to see what's there and to find the programs and data you need.

1.4.5 On-line Help

If you remember only one thing about any computer or program (in addition to starting up), remember how to access and use any available on-line help. If you know this, you can discover all the other necessary information on your own. You can also get beyond the basics covered by this book and become more of a real expert in that system.

1.4.6 Dealing with Files: The Most Important Commands

Each operating system's "Most Important Commands" chapter describes the crucial commands for manipulating and organizing files.

Using a computer is ultimately about the manipulation of files. (Some systems may use a different term instead of "file," but the same principles apply.) These are the units in which we group stored data on a computer, regardless of the data's source, destination, or purpose. We take files called programs that are instructions to the computer, run them, and use them to create, look at, alter, delete, and transfer other files.

This section shows you the most important things that you can do to files:

- List their names, with any pertinent information about them (like their size and the last time

that they were changed).

- Put a text file's contents on the screen where you can see it.
- Copy files.
- Rename files.
- Delete files.

Where applicable, this section also shows how to create and remove subdirectories (subdivisions of the disk space allotted to your user ID) and how to move files in and out of these subdirectories.

Many operating systems provide a way to control other users' access to your files. If not knowing this prevents you from using any of the other essential commands, then the "Using Files" chapter explains how to control access to your files. For example, an OpenVMS system may not give you permission to remove your own subdirectories, so the "Using Files in OpenVMS" chapter explains how to give yourself permission with the `SET PROTECTION` command. When you create a command file in UNIX, you may not have permission to execute it, so the "Using Files in UNIX" chapter explains how to give yourself this permission with the `chmod` command.

This section also prepares you for an operating system's typical error messages. A beginner will encounter many, and knowing which part of these messages to ignore and which parts can help you will speed the process of discovering what you did wrong.

1.4.7 The Text Editor

To create your own text files, or to edit existing ones, you need a text editor. Writing your own command files requires one. Composing mail usually requires one (unless the mail program has a text editor built in). Files moved to one computer from another running a different operating system often have extraneous characters that need to be removed. (For example, although a text file is a text file on any system, older DOS programs indicated the end of a text file with a ^Z character, which may show up as an unwanted character at the end of a file that was moved from a DOS PC to another computer.)

Text Files, Binary Files

A text file is generally made up of keyboard characters. Because the American National Standard Institute (ANSI) standardized a code that specified which bytes represent which keyboard characters (the Ameri-

can Standard Code for Information Interchange, or ASCII), each letter or number in a text file is represented by the same byte on nearly any computer. For example, the uppercase letter "A" is represented by 01000001, or byte number 65. This system makes it easy to move text files from a computer running one operating system to a computer running another.

Most programs that you run on a computer were originally written as text files, then compiled into binary files. In other words, they were translated by a special program known as a compiler into the computer's own language. These files are just collections of bits and will appear as gibberish if you try to look at them. Compiled programs aren't the only binary files—data files stored by popular word processors, spreadsheets, and graphics programs are also binary. They are full of hidden codes and compressed data stored in whatever data structures the programmers who designed them thought would be most efficient.

To do anything with these binary data files, you usually need a copy of the program that created them. (For instance, to edit a file created with Microsoft Word, you need Microsoft Word.) Text files, however, are more universal; any text editor can edit any text file.

This book shows you how to use the text editor that comes with each operating system to do the following:

- Create a new text file.
- Edit an existing file.
- Insert new text into a file.

- Delete existing characters or lines from a file.
- Search for an expression within a file.
- Save your changes.
- Abort your editing session, so that changes are not saved.

1.4.8 Printing Text Files

A computer industry analyst once said "The paperless office is about as useful as the paperless bathroom." Today, people are using their screens more often to read the information that they need, but the vast majority of computing that takes place still leads to a hard copy result. People use word processors to write letters that they will print out and mail; they use spreadsheet and database programs to print out data that they will photocopy and distribute to co-workers.

Sending a file to a printer is usually pretty simple. The use of a multiuser system adds a few wrinkles. Because the system may have more than one printer available, you might have to identify the printer you want to use. Since other users may have print jobs waiting to print on that same printer, this section also shows how to list out the waiting print jobs, if possible, to get an idea of how long you must wait. And, if you send something to the printer and then change your mind about printing it, this section shows you how to cancel the print job.

1.4.9 Command Files

Any computer that lets you issue commands by typing them at a command line will also allow you to create a text file with a series of commands and to then execute those commands by typing in the name of the file where you stored them. These command files provide a way to automate repetitive tasks. In some cases, they provide a complete programming language.

Command files help you pretend to know more about an operating system than you really do for two reasons:

- Beginners tend to believe that only experts write and use command files.
- If you look up the syntax of complex commands and put them into command files with simple names that are easy to remember, you'll have an easy way to execute complex commands without remembering their syntax.

It's not hard to create simple command files if you are reasonably comfortable with a system's text editor. This section shows you how.

1.4.10 Sending and Receiving Mail

The ability to communicate through electronic mail, or "e-mail," is one of the greatest advantages of a multi-user system. It cuts down on phone tag. It allows you to ask people questions when you feel like it, not when they're free, and they can answer you when it's convenient for them. If the subject of your discussion is in a file on the computer, you can send it, or embed part of it in your message. If you can log in to the computer over telephone lines, you can communicate remotely, which by itself has several advantages: you can communicate with people whose hours are different from yours, and you can keep current on issues in the office when you are on the road or staying at home. Many people have accounts on multi-user systems solely to use the electronic mail program.

Because the mail program that comes with a system often leaves something to be desired, many companies purchase and install a better one. Different mail programs on the same system can usually communicate with each other, so even if your company has purchased a new one, you won't be wasting your time if you learn the basics of the one that's included with the operating system. The following are the crucial tasks for dealing with your mail:

- Checking to see if you have mail.
- Reading messages.
- Deleting messages.
- Saving a received message as a file.
- Replying to a message.
- Creating a new message.
- Sending an existing file to someone.

1.4.11 A Sample Session

Each part of the book ends with a brief scenario that demonstrates a typical session on that operating system. Joe User, our hero, works in an office where he is expected to know the basics of the system described. Usually, he logs in and finds a mail message that asks him to do something. As he sets out to fulfill the request, he may encounter problems, but perseverance and the information in this book enable him to work around them. When you read about someone executing the various commands described for a given operating system, it becomes easier to see how some of the pieces fit together in a typical situation.

You will also find references, in the sample sessions and throughout the book, to a mythical

database management program called UpRiteBase. No such program really exists; in describing how an application might fit into a typical installed version of each operating system, I simply chose to make up a sample application program name.

1.5 General Advice

Your life will be easier if you keep certain hints in mind when you attack a new operating system. Trying to distinguish the key differences and similarities between a new one and one that you already know can be very confusing. This section offers some advice on strategies by which you can take advantage of the similarities and minimize the problems caused by the differences.

> **BUZZWORD** *String* This isn't really a buzzword, but a technical term describing a sequence of characters. Quotes are often used to show where a string begins and ends; for example, "here is one" and 'here is another'.

1.5.1 Filenames

The rules for naming files on different computers can seem very different. How many parts a name has, how long these parts can be, and which characters you may use vary from computer to computer. I like to use some guidelines that are based on the common denominator of all the rules I know. They allow me to make up a filename on a particular computer while knowing that the same name would cause little, if any, trouble on another. This is particularly important when using a computer that may be attached, through a network, to other computers; it's a shame when all the technical details of file transfer and file sharing are worked out and automated for you but they don't work because of something as simple as one system's inability to recognize a filename on another.

- *Parts in the name:* UNIX, like the Macintosh and the Amiga, has filenames of only one string of characters; the name of a DOS filename has two parts; CMS, MVS, and OpenVMS filenames have three parts. Two is a nice compromise, and systems that use three-part names usually add a default third part if you don't include one. It's a common practice

(although not a rule) on UNIX machines to use a period as a filename's second, third, or fourth-to-last character, separating the last few characters so that they can give a clue about the type of file it is. `junememo.txt, clean_up.c,` and `budget.wks` would not be unusual filenames. The use of the period on DOS and OpenVMS machines to separate the first two parts of the filenames makes these names just as valid on both of these systems. The use of more than one period in a filename is a good example of something that works on some systems but not on others, and which you should therefore avoid.

- *Length of the parts:* Systems that divide the name up into parts rarely allow any part to be more than eight characters long. A three-character length for the filename's second part is a convention in UNIX and OpenVMS and a rule in DOS, so a maximum of eight for the first part of our two-part compromise and up to three for the second is a guideline that gets you by on many different operating systems.

  The second part of these names usually give a clue about the nature of those files—for instance, `clean_up.c` would be the source code for a program written in the C programming language, `budget.wks` might be a worksheet, and `junememo.txt` is probably a simple text file.

- *Characters to use:* All the operating systems that I know of allow you to use the letters of the alphabet and the ten numeric digits. All but MVS and OS/400 allow you to use the underscore character (_). If you use some other character from the top row of your keyboard, you may find that it works at first but causes a problem later. Maybe the problem will occur when you move the file to another computer; maybe it will occur on the computer where you created the file. The section below entitled "Wildcards" shows examples of characters that lead to trouble if you try to incorporate them into filenames.

  Filenames on any computer are notorious for torturous abbreviations that completely obscure their meaning. This is where the underscore comes in handy on operating systems that allow its use in filenames—if you use it to separate different abbreviations, the name becomes easier to figure out. `aprschd.txt` is a tough one, but `apr_schd.txt` is a little closer to "april schedule."

1.5.1.1 Wildcards

If you want to carry out an operation on seventeen different files, you don't necessarily need to type the command seventeen times. Nearly all operating systems let you refer to more than one file at once by using wildcards to take advantage of common characters in the filenames.

A wildcard is a special character that does for regular characters what its namesake does for regular cards in poker: it can be treated as any other character—or, sometimes, other characters. For example, to delete the files `april1.txt, april8.txt, april15.txt, april15a.txt,` and `april22.txt,` you don't need to type in five commands that each delete a single file. On most computers, you can simply tell the operating system to delete some-

thing like `april*.txt`.

On most systems, an asterisk represents zero or more characters; other special characters are often available to represent other numbers or ranges of characters. (This is why you should stick to letters and numbers when you make up filenames—characters that can represent other characters will cause trouble.)

For example, if one computer's command to list out filenames is the word `list`, then the command

```
list schedule*
```

might list files with the names `schedule.jan`, `schedule.feb`, `schedule.bak`, `schedules`, even a file whose entire name is just `schedule`, because the asterisk can represent zero or more characters. The term `schedule*` is known as the *file specification*, because it is not itself a filename, but a way to specify a file or group of files.

Wildcards are often demonstrated with the command that lists files, but they can usually be used with other commands as well. For example, if the command to copy a file is the word `copy` and the word `accounting` represents a disk, subdirectory or other location where you store accounting files, then the file specification `budget*` in the command

```
copy budget* accounting
```

could copy the files `budget92`, `budget93`, `budget94`, `budget.bak`, and `budget` to the new location.

Notice how both sample commands have the asterisk at the end of the file specification. On some operating systems, it might be possible to list out `aprbud.94`, `maybud.94`, and `junbud.94` by typing this:

```
list *bud.94
```

Here the asterisk represents any letters at the beginning of a filename. This is not as common as using an asterisk at the end of the name or, on an operating system that allows multi-part filenames, at the end of one part of the name. Check the "Wildcards" section for each operating system to make sure.

The file `litebud.txt` would also be listed by the above command, because of the asterisk's flexibility. What if you don't want to be that flexible and only want to list filenames with exactly three letters before the "bud.txt" part? Most operating systems also offer a wildcard to represent individual characters. If it were a question mark, then the command

```
copy part?cal.txt accounting
```

would copy the files `part1cal.txt`, `part2cal.txt`, `part3cal.txt`, and `part4.cal` to the accounting area, but it would not copy `part12cal.txt` or `partcal.txt`.

This character could be repeated to represent a specific number of characters in a filename. This would solve our problem of listing the 1994 budget files without including `litebud.txt`; if each begins with exactly three letters before the "bud.94" that they share, you could enter this:

```
list ???bud.94
```

This brings us to an important point about naming your files: files with a similar purpose should have similar names, so that you can deal with them as a group with a minimum of typing. If the 1994 budget files had been called `aprilbud.94`, `bud94.may`, and `june94.bud`, it would take three separate commands to delete, list, or copy them on many operating systems.

And remember, these aren't the only commands for manipulating files. Any command that can do something to a file—whether it prints it, searches through it, or e-mails it to another user—can usually do it to many files at once, if you know how to use wildcards.

1.5.1.2 Wildcards and File Deletion

When you enter any operating system's command to delete files and use a file specification that includes wildcards, it's a good idea to first use the same file specification with the command that lists files. This way, you see a list of which files you're about to erase.

For example, let's say a given operating system's commands for listing and deleting files are `LIST` and `DELETE`, and the asterisk is used to represent one or more characters. If you want to delete five files that all begin with the letters "JUNE," you should enter

```
LIST JUNE*
```

before you type this:

```
DELETE JUNE*
```

If `LIST JUNE*` lists seven files that fit that pattern, then you'll know that your pattern is too general, and that using the same pattern with the `DELETE` command would have deleted more files than you had intended.

The command that lists file names can be very useful as you learn about other file manipulation commands. Until you are comfortable with a system, always use the file listing command to make sure that the results of your delete, copy, and rename commands had the desired effect.

1.5.2 Mail

The best way to learn about the mail system on any computer is to send mail to your own ID. It's nice if you have a friend willing to put up with messages like

```
Subject:mail^H^Hest?
I    ope thethis woks^[^[rks
```

but you'll spare yourself some embarrassment and get a good idea of what your e-mail looks like to recipients if you send your first messages to yourself.

To be comfortable with a mail program, you have to be comfortable with the text editor first. Make sure you understand the basics of creating and editing text files before you try to get too far with your electronic mail system.

1.5.3 The Text Editor

Don't wait until you have to create important files before practicing with the text editor. At that point, you'll want to concentrate on what you're saying, not on the commands and keystrokes necessary to make the file look the way you want. Create some dummy files, or letters to e-mail to yourself for e-mail practice. Just make sure that they're files that will cause you no pain if you ruin them.

A good opportunity to practice with the editor is to use it to take notes about the operating system as you play with it. Save your work often, keep a backup copy of the file, and print it every now and then in case something unplanned happens to it. If you don't use that computer for a while and forget some aspects of using it, your file full of notes will be handy when you have to use the system again.

1.5.3.1 Line Editors, Full-Screen Editors

Many operating systems provide you with two different editors: a line editor, which works in TTY mode, and a full-screen editor. The line editor will be the older of the two, and will be provided for the convenience of people who have been using that system for a long time. To use a line editor, you don't move your cursor from line to line as you do with a modern word processor or text editor; you issue commands in terms of line numbers. A typical series of editing commands would tell the editing program to carry out instructions like this: add a new line after line number 5, show me lines 3 to 15, change the phrase "file name" in line 12 to "filename," delete line 11, save the file. Unless this sounds like fun, you'll want to use the full-screen editor provided with each operating system.

BUZZWORD *Full screen* The ancestors of modern computer terminals were called teleprinters or teletypewriters. These were essentially printers with typewriter keyboards attached to them. When you typed a command and pressed Return, the terminal printed your command and then, if there were no problems with it, printed the output underneath it.

The first Video Display Terminals (VDTs) to replace these machines sub-

stituted the VDT screen for the long roll of paper necessary on teletypewriters. Commands and output still appeared one line at a time, but this time they appeared on the bottom of the screen, and as more data appeared the earlier data scrolled up the screen. If it scrolled off the screen, it was lost forever; to see it again, you entered the command again.

Eventually, engineers figured out how to make characters appear at specific places on a screen, instead of always at the bottom. They also devised ways to display the cursor at a specific location and still have the computer read whatever the user typed there. This enabled them to create input forms, or on-screen versions of paper forms, where the user could move the cursor from place to place on the screen and enter the appropriate data.

This more sophisticated way of dealing with terminals became known as "full-screen mode." The old way, in which the text perpetual scrolled from the bottom of the screen to the top, became known as teletypewriter mode, or more commonly, TTY mode.

Although this book will show you the most common text editor available for each operating system, you should investigate any alternative editors available on your system. Some sites purchase and install a different text editing program if they consider it superior to the one included with the operating system.

People who are familiar with the editor on one system can often find a version of it for another operating system that they may use—the PC/DOS KEDIT editor is a PC version of the mainframe XEDIT program, and various versions of the UNIX vi editor are also available for PC/DOS and OpenVMS. So, when you move to a new operating system, you may not have to learn a new text editor.

1.5.3.2 The Editing Buffer

On several different platforms, the word "buffer" comes up occasionally in the text editor's status messages and, if available, in its on-line help. It just means "the part of memory set aside for the file you're editing." When you edit a file on any computer, it copies the file from the disk to the computer's memory, and you then edit that copy. Saving your work means copying the edited version in memory back to disk; this is why you lose your work when you lose power in the middle of editing a file on a personal computer.

Sometimes technical talk refers to the copy sitting in memory as the copy in the buffer. For example, when you edit a file with the OpenVMS EVE editor and then quit without first saving your changes, a message tells you "Buffer modifications will not be saved, continue quitting?" This means "The edits that you made to the copy of your file sitting in memory won't be saved, are you sure you want to quit?"

1.5.4 Looking at Text Files

All command-line operating systems have a command to display a text file's contents on your screen. The command might be `TYPE`, `cat`, or `LIST`. (This last one can be confusing, because it displays a text file in MVS, but lists file names in VM/CMS.) Remember that these commands are for looking at *text* files, not binary files. If you try to display a binary file with one of these commands, the system tries to interpret the binary information as text so that it can put it on the screen. At best, it looks like gibberish; at worst, the system interprets some of the information as special codes telling it that you've changed the settings on your terminal or terminal emulation software. In response to this, it starts sending codes to your terminal that have nothing to do with what your terminal expects. If this happens, your terminal may lock up, forcing you to end your session and start all over again.

Moral: don't start using this command to try to look at every file whose name shows up when you list filenames. One part of the filename on each system (usually the second part—it might be called the filetype or extension) gives you a clue as to what kind of file each one is. Get to know which ones represent text files and which represent binary files.

1.5.5 "Printing" on the Screen

Be careful when you come across the word "print" in a command or a command's description. In the days of teletypewriters, every program that showed you any information literally printed it on the paper that scrolled through the terminal. As VDTs proliferated in the nineteen-seventies, they existed side-by-side with the teletypewriters, and it was understood that text that would have printed on a teletypewriter was displayed on a VDT's screen.

Although VDTs have replaced teletypewriters, the terminology still hadn't changed. This means that today a help message that explains that a given command "prints the names of your files" doesn't mean that it sends filenames to the printer; it means that it "prints" it on your terminal.

1.5.6 Reading and Writing

When discussing a computer's operations, the use of the terms "read" and "write" often confuse novices. Reading a file doesn't necessarily involve your seeing it; it's the hardware that reads and writes data. A tape recorder provides the best analogy to understand what computers are doing when they read and write: writing is essentially the act of recording data on your storage medium, and reading is the playback of that data—that is, pulling it off the storage medium so that you can use it.

When you save a file that you created with a word processor or spreadsheet, you are writing it to disk. (Writing it "on" the disk may sound like better English, but we're talking computer talk here—the correct preposition is "to.") When you call up a previously created file into your word processor or spreadsheet, you are reading it from the disk. When you copy a file from one disk to another, your computer reads it from the source disk and writes it to the destination disk; this is like playing a song that has been stored on one tape while you record it on another.

Just as audio cassettes have a little piece of plastic that you can punch out to prevent someone from recording over the information that exists there, floppy diskettes have either a little plastic switch to move or a notch that you cover with a sticker to prevent anyone from "recording" on that diskette, or writing over the information there. This protects the data on that diskette, and is called a "write protect" switch.

One of the most dreaded error messages on any kind of computer is a "read error." This usually means that there's a problem with the disk that the computer is trying to read. Picture an audio cassette that fell into a pond and then, after it was fished out, was left out in the cold so that the moisture inside had a chance to freeze up. Your tape recorder would have a hard time playing this cassette or reading the information stored on its tape. If your hardware can't get the data off the storage medium, the data is lost. The same principle applies to the data on disks. That's why people make backups—with more than one copy of valuable information, they can read from the backup if the primary disk is corrupted (computer talk for "screwed up").

1.5.7 Logging Off (or Out)

When you type the command to end your session on a computer and press Enter, some computers give you a clear indication that you finished your session—for example, a summary that shows the time of day you quit and the amount of time that you were connected. Others don't. You should always make sure that you have properly ended your session, since some sites bill you according to the amount of time that you are connected.

If you don't see such a message, press Enter a couple of times and see what happens. If you return to a login screen or login prompt, you know that you no longer have an active session. If you're still unsure, type a simple command for that operating system and see if anything happens. If anything does, you're still logged in.

What happens at the end of a session can vary from site to site. Even an expert on a particular operating system would not know exactly what to expect at a particular installation. Don't be afraid to ask.

1.5.8 Terminal Emulation and File Transfer

All operating systems have communications software available for doing file transfer to and from other computers. It may be part of the operating system, or it may be purchased from a third party. (Very likely, both hold true on a particular system; as with text editors and mail programs, the one included with the operating system may be so limited that a commercial one is purchased anyway.) On a personal computer or workstation, these programs often must also do terminal emulation, which lets you use your local computer as if it were a terminal for the host computer.

The Kermit file transfer and terminal emulation program is available for virtually all operating systems, and it's free. This is particularly important when doing downsizing work, because downsizing means "moving an application from a larger computer to a smaller one" and you need a program to do the file transfer. Once you learn Kermit's most important commands, you can use it on any operating system where you find it installed.

For personal computers, Kermit is available from nearly any bulletin board catering to your computer. CompuServe has it for DOS computers, the Macintosh, the Amiga, and the Atari-ST (use the file finder forums—IBMFF, MACFF, AMIGAFF, and ATARIFF—to locate the latest versions). On minicomputers and mainframes, it takes a system administrator to install Kermit, but if you're using a computer that is part of an academic computing center, you can bet that Kermit is already installed.

To find out, enter

```
kermit
```

at the operating system's command prompt. If the Kermit prompt appears (this will look different on different systems, but probably be some variant of `Kermit>`), you've found it. Enter

```
help
```

at the Kermit prompt to learn more.

1.5.8.1 Emulated Terminals

There are two terminals whose names come up often, even though you may never see examples of the actual terminals:

- The *VT100* was one of the original members of DEC's VT ("virtual terminal") series of terminals. Although they have come out with more advanced models since (each designated by

a higher number, such as VT220 and VT340), emulation of the VT100 has become a baseline of emulation competence for emulation programs and for the terminals with which a system will work. If the terminals that your system will work with and the terminals that your emulation software can emulate only have one name in common, it will be VT100. If they have other names in common, they will probably be more sophisticated terminals, and more worthwhile for you to use.

- The *3270* was an important IBM terminal, whose descendants (the 3278, 3279, etc.) are also still in use. People refer to the family generically as "3270 terminals."

  3270 keyboards have a couple of keys not found on typical personal computer keyboards (for example, the Reset key, and separate Return and Enter keys) so when you're emulating a 3270 terminal, you must sometimes figure out which personal computer keys are standing in for the 3270 keys not found on a typical personal computer keyboard. In this book, you'll find 3270 issues mentioned in the material on the three IBM systems covered: OS/400, VM/CMS, and MVS.

1.6 Syntax Expressions in this Book

This book shows you the syntax for many commands in the various operating systems covered. Some commands have mandatory parts, optional parts, and default settings. The following conventions show which parts of a command's syntax are what:

`[ ]`	Anything in brackets is optional.
`/`	A slash indicates options from which to choose.
`underline`	If several options are possible, the default setting is underlined.
`Key+Key`	When two keys should be pressed simultaneously, they are written with a plus sign between them. For example, to type an upper-case "S," you would press Shift+S.

The following syntax for the mythical `whatgives` command

```
whatgives [today/yesterday/tomorrow]
```

means that you could type the word `whatgives` by itself, because everything else is in square brackets, and therefore optional. If you did include a parameter, it should be either "today," "yesterday," or "tomorrow" because the slash characters show that these are the only options. If you didn't include any parameter, the `whatgives` command would be executed as if you had put "today" after it; this is the default parameter, as indicated by the underline.

1.7 Comments and Suggestions

Comments and suggestions about this book can be sent to me in care of McGraw-Hill Professional Book Group, 11 West 19th Street, 3rd Floor, New York NY 10011.

To send them more directly to me, you can use electronic mail to send them to BOB-DUCHARME@ACM.ORG or CompuServe 72441,3003.