Running MS-DOS Version 6.22, 20th Anniversary Edition

Van Wolverton

PUBLISHED BY *Microsoft Press A Division of Microsoft Corporation*
One Microsoft Way ,
Redmond,
Washington 98052-6399

Library of Congress Cataloging-in-Publication Data

Wolverton, Van, 1939-
Running MS-DOS / Van Wolverton.--6th Rev. ed.
p. cm.
Includes index.

ISBN 0-73561-812-7

ISBN 0-7356-1812-7 (20th Anniversary Ed.)

1. MS-DOS (Computer file) I. Title.

QA76.76.O63W65 1993

005.4'469--dc20 93-37565

CIP

Printed and bound in the United States of America.

1 2 3 4 5 6 7 8 9 QWE 8 7 6 5 4 3

Distributed in Canada by H.B. Fenn and Company Ltd.

A CIP catalogue record for this book is available from the British Library.

Microsoft Press books are available through booksellers and distributors worldwide. For further information about international editions, contact your local Microsoft Corporation office or contact Microsoft Press International directly at fax (425) 936-7329. Visit our Web site at www.microsoft.com/mspress. Send comments to <mspinput@microsoft.com>.

The example companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted herein are fictitious. No association with any real company, organization, product, domain name, e-mail address, logo, person, place, or event is intended or should be inferred.

Acquisitions Editor: *Alex Blanton* ***Project Editor:*** *Sandra Haynes Body Part No. X08-92504*

For Jeanne, who makes it all worthwhile

VAN WOLVERTON

A professional writer since 1963, Van Wolverton has had bylines as a newspaper reporter, editorial writer, political columnist, and technical writer. He wrote his first computer program-one that tabulated political polls-for the Idaho State Journal in Pocatello, Idaho, in 1965. His interests in computers and writing have been intertwined ever since. As a computer professional, Wolverton has worked at IBM and Intel and has written software documentation for national software companies, including Microsoft Corporation. He is the author of Hard Disk Management and MS-DOS Commands, both Microsoft Press Quick Reference Guides, and Supercharging MS-DOS. He was also a contributor to The MS-DOS Encyclopedia. Wolverton and his wife, Jeanne, live in a twenty-first-century log cabin near Alberton, Montana.

Running MS-DOS Version 6.22, 20th Anniversary Edition

by Van Wolverton

ISBN:0735618127

Microsoft Press © 2003 (612 pages)

The classic reference to the classic operating system!

Table of Contents

Running MS-DOS Version 6.22, 20th Anniversary Edition

Preface to the 20th Anniversary Edition

Introduction

Part I - Getting to Know MS-DOS

Chapter 1 - What is MS-DOS?
Chapter 2 - Starting MS-DOS
Chapter 3 - Getting Your Bearings
Chapter 4 - A Look at Files and Floppy Disks

Part II - Learning to Use MS-DOS

Chapter 5 - Managing Your Files
Chapter 6 - Managing Your Floppy Disks
Chapter 7 - Managing Your Devices
Chapter 8 - A Tree of Files
Chapter 9 - Managing Your Hard Disk
Chapter 10 - Protecting Your Disks and Files
Chapter 11 - The MS-DOS Shell
Chapter 12 - Creating and Editing Files of Text
Chapter 13 - Taking Control of Your System
Chapter 14 - Creating Your Own Commands
Chapter 15 - Creating Smart Commands

Chapter 16 - Creating More Smart Commands
Chapter 17 - Tailoring Your System
Part III - Appendixes
Appendix A - Installing MS-DOS
Appendix B - Glossary
Appendix C - MS-DOS Command Reference
Index

List of Figures

Running MS-DOS Version 6.22, 20th Anniversary Edition

by Van Wolverton

ISBN:0735618127

Microsoft Press © 2003 (612 pages)

The classic reference to the classic operating system!

Back Cover

Get the book that set the standard for all other MS-DOS books—now celebrating its 20th anniversary! *Running MS-DOS* is the best selling guide to the operating system that changed personal computing history. Featuring Van Wolverton's down-to-earth style and eloquent applications, this one-step reference makes MS-DOS accessible for anyone looking to optimize PC performance. Whether you work in tech support or simply want to keep your classic PC in top form, Van shows you how to master MS-DOS with unparalleled clarity and expertise!

Discover how to put MS-DOS functions and commands to work!

- Tweak your system so it runs more efficiently
- Take control of your disk drives and devices
- Create back ups and rescue deleted work
- Retrieve files faster and manage memory
- Run legacy applications—including classic games
- Write your own batch files and smart commands!

Plus, check out the comprehensive *MS-DOS Command Reference* in the appendix—great for answers and examples on the spot!

About the Author

Van Wolverton, a writer since 1963, has had bylines as a newspaper reporter, editorial writer, political columnist, and technical writer. He wrote his first computer program—one that tabulated political polls—for the *Idaho State Journal* in Pocatello, Idaho, in1965. His interests in computers and writing have been intertwined ever since. As a computer professional, Wolverton has worked at IBM and Intel and has written software documentation for national software companies, including Microsoft Corporation. He is the author of *Hard Disk Management* and *MS-DOS Commands*, both Microsoft Press Quick Reference Guides, and

Supercharging MS-DOS. He was also a contributor to *The MS-DOS Encyclopedia*

.

Running MS-DOS Version 6.22, 20th Anniversary Edition

by Van Wolverton

ISBN:0735618127

Microsoft Press © 2003 (612 pages)

The classic reference to the classic operating system!

Table of Contents | Back Cover | Comments

There are currently no comments on this book.

Preface to the 20th Anniversary Edition

What is 20 years?

To a writer like John McPhee [1], awarded a Pulitzer Prize for documenting the riffs of poetry, the lyrical odes and sonnets gracing the undending hymn of the epic, tumbled among the tuff and breccia and sandstone and mudstone and granite and diorite and all the other rock that makes up a slice through the U.S. roughly along the path of Interstate 80 from New Jersey to California—to a writer like John McPhee, soldiering away at presenting 540 million years (give or take), 20 years doesn't register, unless perhaps as a possible way to estimate the length of some cataclysmic event, a volcanic eruption or sudden draining of many cubic miles of rock—but these events would more likely be measured in 20 days or even 20 hours; that's what made them cataclysmic. No, I would avoid asking John McPhee to ponder the period of 20 years for fear of evoking the same sort of response I got when I asked Derry Odonovan, racing enthusiast and true son of Eire, why the drag racer he built (and drove) didn't have a tachometer (although I said, of course, *tach*): Well, let's see (rolls his eyes a bit), I make it through the quarter in something less than 5 seconds and the engine's turning around 7 grand most of the time, which means that (longer pause, eyes squinched tight shut to let me know that he is calculating...) the beast will turn over maybe 600 times. Now why would you be wantin' me to *count* them? (Although he had long since left the Emerald Isle for California and supported his racing by masquerading as a lineman for PG&E and sounded as though he was born and raised in the Santa Clara Valley, whenever I asked a question that required him to expend more than 15 or 20 words in showing just how little I really knew, he sounded like a leprechaun by the time he was through.)

So, I won't trouble John McPhee.

To a writer like Tracy Kidder, though, the problem is just the opposite. In *Soul of a New Machine* [2], Kidder camps out for months in the offices of a driven team of programmers and engineers at Data General Corporation as they labor to design a killer new computer. (Come to think of it, he got a Pulitzer, too—is some sort of envy creeping into this Preface?) Kidder shows us the effect on each member of this happy band as seemingly inexplicable timing errors—the semiconductor equivalent of a clock every once in a while going *tock* instead of *tick-tock*—keeps their elegantly designed creation from being able to add 2 +2, let alone thrust DG back into contention in the minicomputer market. What made it so much more difficult was that there were many of these errant clocks, each beating millions or billions of times each second. One burnt-out engineer finally disappeared one night, leaving a note on his chair that said simply "I'm going to a commune in Vermont and will deal with no unit of time shorter than a season."

Twenty *years*? I don't think I'll bother Mr. Kidder, either.

A human generation is most often defined as 20 years, so that's one way to look at the period of time. But we're concerned not just with humans, but with their creations, among them the computer, in considering the import of 20 years here. And generation has quite a

different meaning when we talk about computers.

First-generation computers were built in the years following World War II. These behemoths generated enormous amounts of heat because they used vacuum tubes to do their calculations. About 2,500 first-generation computers were built throughout the world between 1946 and (roughly) 1958. Among the most powerful computers of this period were those in the SAGE (Semi-Automated Ground Environment) air defense centers, three-story buildings scattered around the U.S. that contained two computers, each with 55,000 vacuum tubes. A SAGE center required the same amount of electricity as a small city, and had about the same computing power as an early PC (of, say, about 20 years ago).

Second-generation computers used transistors instead of vacuum tubes. They were still big, and still generated a lot of heat, but a great improvement, and ruled the roost from the mid-50s to the mid-60s. Third-generation computers came along in the mid-60s when integrated circuits were developed, putting several transistors on a single chip. Third-generation computers reigned until the early 80s, almost the same length of time as a human generation.

But a stealth technology was creeping into the computer market. Engineers had been pushing the technology, jamming more and more transistors onto a chip, finding more and more ways to increase speed and decrease size. First there were minicomputers, those desk-sized (rather than room-sized) machines Kidder described so well in his book. Then Intel Corporation put all the functions of a simple calculator on a single chip in 1971; the use of "generation" as a term of stability and orderly progress was doomed.

Within a few years, Gordon Moore of Intel postulated that the number of transistors on—and, therefore, the capability of—a single chip would double every 18 months. That eerily prescient forecast meant that there was no point in trying to keep track of "generations" any more. Today's microprocessors have more than 40 million transistors on a single chip. Twenty years ago, the processor that powered most PCs (Intel 80286) had 134,000 transistors. Twenty years before that there were no microprocessors, just individual transistors. Twenty years is a long time.

It was only a matter of time before someone realized that if you could make a calculator with a chip, keyboard, and a display, you ought to be able to make a computer the same way. And so it came to pass. The January 1975 issue of Popular Electronics magazine featured on its cover a computer kit for $395. When a young Harvard student named Bill Gates saw that article, it didn't take him long to decide that the world was about to change forever, and that he was going to be one of the agents of that change. He left Harvard, flew to Albuquerque where the Altair was built, and formed a company called Microsoft to provide software for the Altair and all the other personal computers he knew would follow.

In the midst of all this, lesser folk toiled away. I worked for IBM for 10 years as a technical writer and editor, and then left in 1977 to join a company called Intel. There I learned to use a personal computer and write programs in Basic (*Microsoft* Basic, as it happened) by taking home what Intel called a development system on weekends. It consisted of three

large cabinets that filled a 5-foot-tall equipment rack, plus keyboard and a monitor. (Luckily, we had a van.) My first contact with Microsoft came then, when I spoke on the telephone with Andrea Lewis, the only female among the original founders of Microsoft. She had written the manual for Microsoft Basic; one of the writers who worked for me was rewriting it, and I cleared the project with her (and cleared up some technical questions we had in the process).

After 3 years I left Intel to join a startup so I could become a millionaire when it went public. Here I had my second contact with Microsoft, this one face-to-face on the sixth floor of a bank building in downtown Bellevue, Washington. Our company bought the first Basic (what else?) compiler that Microsoft had done for the Intel 8086 processor. This was 1979; Microsoft had fewer than 30 employees, the average age was somewhere south of 20, and it was mightily depressing for this 40-year-old because I could sense something big was happening here.

Unfortunately, instead of going public my startup went belly-up and I found myself an unemployed entrepreneur. Turning to what I knew, I tried freelance technical writing. I was lucky: the time was right, a lot was happening, and I was able to find work. Within a few weeks, I was writing a manual for a hush-hush machine being built by a company called Florida Computer Systems. I had to sign in at a security desk, get the key to a locked room, and sit in this windowless room learning to use a program called VisiCalc, the first spreadsheet program. The computer used an operating system called DOS, or sometimes PC-DOS. (It was a while before someone told me that DOS was written not by IBM, but by a company called Microsoft.)

Within a year, Microsoft was one of my clients and I was writing manuals for them that described the internal workings of DOS so programmers could write other programs to run on the IBM PC and other computers that used DOS. Because, although IBM called it PC-DOS, Microsoft had retained the rights to sell it to other companies and *they* called it MS-DOS. About a year later, I made one of several trips to Seattle. By now Microsoft had nearly 200 employees and occupied a building just inside the Redmond city limits (next to a Burgermaster drive-in that remains part of Microsoft's Arthurian legend).

Andrea Lewis was one of my contacts at Microsoft; she stopped me in the hall one afternoon, told me that Microsoft was going to establish a book-publishing division, and asked if I'd be interested in writing a book that told people how to use DOS. I said, without much thought, sure. She said she'd be in touch.

A couple of months later I got a call from the publisher of what was now called Microsoft Press asking if I could meet him in Palo Alto to discuss a contract. I did, signed the contract, wrote the book, and began a long, productive, and generally happy relationship with Microsoft Press.

Acknowledgements

Twenty years is a long time. Hardly anyone I know is left at Microsoft Press, let alone at Microsoft, where I used to track down the DOS programmers in their lairs to try to figure out just exactly what was happening under the covers while we users innocently entered commands and fiddled with our files. But it's never too late to thank those who played significant roles in the birth and long, long life of this book that has sold several million copies (no one seems to know just quite *how* many million) in a couple of dozen languages.

The first would have to be Andrea Lewis, of course, for asking me to write the book, and then Nahum Stiskin, the founding publisher of Microsoft Press who established a standard of quality of writing, editing, and design that set the computer-book world on its ear in the early 1980s. Nahum and I sat in the bar of Ricky's Hyatt House in Palo Alto the summer of 1982 for a couple of hours talking about books, trading various lies, and finally agreeing on a contract that required virtually no haggling.

There have been literally dozens at Microsoft Press who have contributed mightily to the quality and success of *Running MS-DOS*. I can't remember most of them now; twenty years is a long time. But they were always there when they were needed, proofreaders and copy editors and technical editors, marketing mavens, PR wizards, designers and illustrators, publishers and assistant publishers who put up with my carping and temperamental artist's frustrations. You done good, all of you.

I'm in serious danger of writing hagiography when it comes to the next thank-you. The first Microsoft Press editor of this book, JoAnne Woodcock, was (and still is, if she's wielding a blue pencil) the finest editor on the face of this earth. Well, I might be persuaded to limit that to the finest editor of computer books in the English language—since that's the extent of my writing experience—but JoAnne, you're simply the best. The book would not be what it is without you.

The last acknowledgement, and the greatest, goes to the one person most responsible for the existence of the book, not to mention its usability, readability, or whatever it is that has moved people to say that *Running MS-DOS* made computers understandable to them. Jeanne Wolverton, my first editor for over 40 years now, mother of my children, grandmother of my grandchildren, she's read drafts, sat at computers for hours on end working the exercises and pointing out when they didn't work (or make any sense), brought me coffee by the Imperial gallon, soothed my fevered brow when it seemed I was doomed not to be able to get this thing done, and generally made sure that (1) I *did* get it done and (2) it was as good as I was capable of doing.

I've had some wonderful memories in the last 20 years associated with Microsoft and this book. Shaking Bill Gates's hand at a couple of very posh parties that Microsoft threw when Microsoft Press was just getting started. A note from Brit Hume saying that the book really worked for him. A letter from the publisher of the Spanish edition telling me that he had presented, at her request, a copy of the book to Queen Sofia of Spain. Michael Dell—a very *young* Michael Dell—grabbing my hand at Comdex and exclaiming "You're *the* Van

Wolverton?" My daughter Kay putting a promotional poster of her dad's book on her dorm room wall. Wow.

With all but a few PCs in the world today running Windows, it's tempting to think of DOS as something that belongs in the annals of a former world, but *au contraire*—within the soul of every new machine that runs Windows is DOS. It's still there, it still works, and it's still the best way to do some things. True to form, Jeanne offers an appropriate example of this: A couple of months ago, she asked me how to print the list of files displayed by Windows Explorer, because there was no Print command on the File menu. So the guru poked around in Windows for a while and couldn't find a way. I went downstairs to try on my machine, and on the way down I thought: DOS! I open a DOS window and typed:

```
dir > fred.txt
copy fred.txt prn
```

Heh heh. See Chapter 13, "Taking Control of Your System."

[1]John McPhee, *Annals of the Former World* (New York: Farrar, Strauss, and Giroux, 2000)

[2]Tracy Kidder, *Soul of a New Machine*, (New York: Little, Brown, 1981)

PREVIOUS NEXT

Introduction

It may be tempting to skip these opening words and "get to the meat of it," but please read this introduction anyway. The information included here is useful and won't take long to read.

You may want to know whether this book applies to you. It does if your computer uses MS-DOS. The book itself was written with an IBM personal computer, but it applies equally well to any machine that uses MS-DOS.

You bought this book—or at least took the time to pick it up and glance through it—despite the manual you got with your copy of MS-DOS. Why? What else can a book like this offer? It can offer simplicity. The MS-DOS manual is thorough and complete. It's your official, comprehensive reference guide to MS-DOS, but its goal is to tell you *about* MS-DOS rather than how to *use* MS-DOS in your everyday work.

This book does not show you how to set up your computer, nor does it describe in detail the pieces of the system, such as the keyboard or the display. These matters should be covered thoroughly in the manuals that came with your computer.

The book assumes neither that you are, nor that you aspire to become, a programmer. It doesn't try to explain how MS-DOS works, and it leaves to the MS-DOS manual the task of explaining some of the more technical features. The book does assume that you have access to an IBM personal computer or one of the many other machines that run MS-DOS, and that you want to put the machine to work. It includes scores of examples, and it is organized by what you want the computer to do, not (as a programmer would expect) by how MS-DOS itself is structured. The examples reflect real-life situations.

You don't have to be a mechanical engineer to drive a car well, but you do need experience. You don't have to be a computer scientist to use MS-DOS well, either, and this book starts you on your way.

What's in the Book and Where

This book covers MS-DOS through version 6.22 as it is used on machines that have a hard disk and either one or two floppy disk drives. Although a hard disk is assumed, the examples are also structured to work on computers that have only two floppy disk drives.

Part 1, Chapters 1 through 4, describes the pieces of the computer system, defines some terms and concepts, and provides hands-on examples that show you the major capabilities of MS-DOS.

Part 2, the bulk of the book, includes Chapters 5 through 17. These chapters show you how to operate your computer system and manage all its parts with the MS-DOS commands.

Chapters 5, 6, and 7 show you how to manage your files, floppy disks, and computer devices such as the printer and the display. Chapter 8 describes the MS-DOS multilevel filing system that allows you to set up a personalized computer file system that matches the way you work. Chapter 9 shows you how to manage the files and directories on a hard disk, and Chapter 10 shows you some ways to protect both the disk and your work from loss or damage.

Chapter 11 takes you through a menu-based program called the Shell (shipped with versions 5 and 6.0 and available on the supplemental disk to versions 6.2, 6.21 and 6.22) that makes your display visually more interesting and your work with the computer easier in many ways. Chapter 12 describes another menu-based program, the MS-DOS Editor (available in versions 5 and later).

Chapters 13 through 17 describe ways to tailor MS-DOS to your own needs. Chapter 13 shows you how to take control of your system with a special set of commands called filter commands; it also shows you how to use a program named Doskey that can save time and work by recording keystrokes and commands. Chapters 14 through 16 show you how to create your own sets of commands and save them in special files called batch files. Chapter 17 shows you several techniques you can use to make MS-DOS immediately useful in its own right, and it describes ways to help MS-DOS run your computer more efficiently.

Finally, in Part 3, Appendix A tells you how to install MS-DOS, Appendix B provides a glossary of commonly used terms, and Appendix C describes the MS-DOS commands, with cross-references to the discussions in the preceding chapters.

If you plan to use your computer for word processing, spreadsheets, database management, games, or any other applications, this book is probably all you'll need. Not only does it show you how to use MS-DOS so you can run your programs, it shows you how to make good use of MS-DOS without additional software.

About the Examples

The best way to learn how to put MS-DOS to work is to use it. This book, therefore, is devoted primarily to examples. Terms and concepts are defined as you need them and are illustrated with hands-on examples that help you see both what you do and why. Because the book covers different versions of MS-DOS and different types of machines, there are variations in some examples; these alternatives are identified. Unless an example states it is for a particular version or computer setup, the MS-DOS displays shown in this book are the MS-DOS version 6 responses on a computer with one hard disk and one floppy disk drive. If you are using a different version of MS-DOS or a different computer setup, the responses you see may vary somewhat. Do not be concerned.

What to Type and When

There's an awkward mismatch between a computer and a book that shows you how to use it. The computer is dynamic: It displays messages, moves data back and forth between disks and memory, prints words and pictures, and chirps now and then to announce completion of another task. When you use the computer, you enter into a dialogue: You type something, the computer responds, you type something else, and so on, back and forth, until your work is done.

A book, however, is static. It can show only snapshots of your dialogue with the system, yet it must describe that dialogue well enough so that you can take part in it. In this book, we have to show what you type and how the computer responds. We have to distinguish parts of this dialogue, such as the names of files and messages displayed on the screen, from the surrounding prose. Here are the conventions we've adopted:

- Hands-on examples are shown in different type, on separate lines, just as you would see them on your display. The characters you type are printed in lowercase colored type; MS-DOS usually doesn't care whether you type in uppercase or lowercase, but lowercase seems to be easier. Here is a sample of the conventions for hands-on examples:

  ```
  C:\>format b:
  Insert new floppy disk for drive B:
  and press ENTER when ready . . .
  ```

- Occasionally, similar information occurs in text. In these instances, the interaction between you and MS-DOS is printed in italics to distinguish it from the surrounding text. For example, you may see "Type *n* when MS-DOS displays *Format another (Y/N)?*"

- Many MS-DOS commands include options, or parameters, that allow you to specify a particular disk drive, file, or piece of equipment, or to use a particular form of the command. Options are shown in angle brackets (<>) when they represent a variable entry, such as the name of a file. When they must be entered exactly, they are

shown in the form you must use. For example, here are some options of the Format command used in the preceding examples (don't worry about understanding the command at this point):

format <drive> /4 /F:<size> /Q

Now it's time to meet MS-DOS. This book was written to be used alongside the system, so put it beside your keyboard, turn to Chapter 1, and get ready to put MS-DOS to work.

PREVIOUS NEXT

Part I: Getting to Know MS-DOS

Chapter List

Chapter 1: What is MS-DOS?

Chapter 2: Starting MS-DOS

Chapter 3: Getting Your Bearings

Chapter 4: A Look at Files and Floppy Disks

Part Overview

Part 1 introduces you to the concept of an operating system: what it is, what it does, and why you need it. It then describes the terms and the basic operating principles of MS-DOS. These four chapters show you how to start MS-DOS and how to control the system with MS-DOS commands. They give you the foundation for using MS-DOS effectively in your daily work with the computer. The information is primarily tutorial, and many examples are included. Later parts of the book contain detailed reference information that describes all the MS-DOS commands and their capabilities.

Chapter 1: What is MS-DOS?

You've got your computer, and you've probably got one or two programs, such as a word processor or a spreadsheet, to use with it. But what is this thing called MS-DOS? Why do you hear so much about it, and why have hundreds of pages of instructions been written for it?

MS-DOS Is a Program

MS-DOS is a program, but it's not just any program. Chances are none of your other programs would work without it because MS-DOS controls every part of the computer system. MS-DOS not only makes it possible for your other programs to work, it also gives you complete control over what your computer does, and how MS-DOS is the link between you and your computer.

To appreciate the role MS-DOS plays, take a quick look at the pieces of your computer system and what they do.

PREVIOUS NEXT

Hardware Makes It Possible

Your computer equipment, called *hardware*, probably includes a keyboard, display, printer, and one or more disk drives. The purposes of the first three are straightforward: You type instructions at the keyboard, and the system responds by displaying or printing messages and results.

The purpose of a disk drive isn't quite so obvious, but it quickly becomes apparent as you use the system: A disk drive records and plays back information, much as a tape deck records and plays back music. The computer's information is recorded in files on disks; you'll find that disk files are as central to your computer work as paper files are to more traditional office work.

PREVIOUS NEXT

Software Makes It Happen

No matter how powerful the hardware, a computer can't do anything without programs, called *software*. Computers use two major types of software: *system programs*, which control the operation of the computer system, and *application programs*, which perform more obviously useful tasks, such as word processing.

Each program uses the hardware. It must be able to receive instructions from the keyboard, display and print results, read and write files from and to a disk, send and receive data through the computer's communications connections, change the colors on a color display, and so on, through all the capabilities of the hardware.

So that each program doesn't have to perform all these functions for itself, a system program called the *operating system* manages the hardware. The operating system allows an application program to concentrate on what it does best, whether it's moving paragraphs about, tracking accounts receivable, or calculating stress in a bridge beam. MS-DOS is an operating system.

PREVIOUS NEXT

MS-DOS Is a Disk Operating System

The most frequently used operating system for IBM and IBM-compatible computers is the Microsoft Disk Operating System—MS-DOS. MS-DOS is called a disk operating system because much of its work involves managing disks and disk files.

What Does an Operating System Do?

An operating system plays a role something like a symphony conductor's. When the score calls for the violins to play, the conductor cues the violins; when the score says the cellos should play more softly, the tympani should stop, or the entire orchestra should pick up the tempo, the conductor so instructs the musicians.

The players in the orchestra and their instruments represent the hardware. The experience and skill of the conductor represent the operating system. The score represents an application program.

When one score is replaced by another—Beethoven's Fifth Symphony is put aside and replaced by Haydn's *Surprise* Symphony, for example—the same musicians use the same instruments, and the same conductor uses the same experience and skills. A different sound, a different mood, perhaps, but the elements are the same.

When one application program is replaced by another—for example, an accounting program is put aside and replaced with a word processor—the same hardware carries out the instructions of the same operating system. A different program, a different purpose, perhaps, but the elements are the same.

MS-DOS coordinates the computer system, just as the conductor coordinates the orchestra. Your application programs run in concert with MS-DOS, trusting it to keep the system humming.

Much of what MS-DOS does, such as how it stores a file on a disk or prints on the printer, is invisible to you. But MS-DOS lets you control the things you care about, such as which program to run, what document to print, or what files to erase. These functions share an important characteristic: They need disks and disk drives.

Disk Drives

Personal computers use two main types of disk: a flexible disk in a protective plastic jacket, called a *floppy disk*, which you can remove from the drive, and a permanently mounted unit called a *hard disk*. There are two types of floppy disks: 5.25 inches square in a flexible plastic jacket, and 3.5 inches square in a rigid plastic shell.

A hard disk holds much more information than a floppy disk—from 15 to 100 times as much, or even more—and is much faster. Most personal computers have one hard disk and one or two floppy disk drives. Machines without a hard disk usually have two floppy disk drives.

To distinguish among the types of disks, this book uses *floppy disk* to mean either type of flexible disk, *hard disk* to mean only a permanently mounted disk, and *disk* to refer to both.

Disk Files

Just as you organize and store your written records in paper files, you organize and store computer information in disk files.

A disk file—usually called a file—is a collection of related information stored on a disk. It could be a letter, an income tax return, or a list of customers. It could also be a program, because the programs you use are stored in files.

Virtually all your computer work revolves around files. Because one of the major functions of MS-DOS is to take care of files, much of this book is devoted to showing you how to create, print, copy, organize, and otherwise manage files.

Where Is MS-DOS?

When your computer is turned off, MS-DOS is stored on disk. Although it's a special type of program, MS-DOS is still a program, and that means it's stored on disk in a set of files like any other collection of computer information.

If your computer has a hard disk, MS-DOS is probably already on it—placed there, perhaps, by your computer dealer, or by someone else who set up your system. If your computer does not have a hard disk, it must use MS-DOS from floppy disks, so it should have come with a copy of MS-DOS on two or more floppy disks.

Different Versions of MS-DOS

MS-DOS has been revised a number of times since its release in 1981; the first version was numbered 1.00. MS-DOS is revised to add more capability, to take advantage of more sophisticated hardware, and to correct errors. When you start up your system, MS-DOS may display the version number you are using.

When a new version of MS-DOS appears, a change in the number following the decimal point—6.0 to 6.2, or even 6.22, for example—marks a minor change that leaves the new version of MS-DOS substantially the same as the previous version. A change in the number preceding the decimal point marks a major change. Version 6.0, for example, adds several new features that weren't available in version 5.0.

Even though newer versions of MS-DOS can do a lot more than earlier versions of MS-DOS, they remain compatible with the earlier versions. Thus, if you start with version 2.1, you can still use all your knowledge and experience, plus your files and disks, when you move to a newer version of MS-DOS.

For simplicity, this book usually refers to MS-DOS by major version number only—for

example, version 5 or version 4, rather than version 5.0 or version 4.01. It also omits references to versions of MS-DOS earlier than version 3, but much of the information applies to these versions too. Remember, version 2 is just as much a part of MS-DOS as version 5. It's simply older and, although it includes many of the features described here, it doesn't provide them all.

What Is Compatibility?

You've no doubt seen the term *IBM-compatible* in an article or an advertisement. What does compatibility actually mean? Compatibility essentially refers to the ability of one computer to use programs and data created for or stored on another computer. In everyday use, the most meaningful measure of compatibility is the extent to which you can use the same programs, data, and floppy disks in computers of different makes or different models:

- If two systems are totally compatible, they can freely use the same programs and floppy disks. This is the type of compatibility exhibited among different models of IBM Personal Computers and the IBM-compatible machines made by manufacturers other than IBM. On these machines, such full compatibility is made possible in part by MS-DOS: Any computer that can run MS-DOS can run programs designed for MS-DOS, and that computer can (given the proper application programs) freely use floppy disks from any other MS-DOS computer.
- Incompatible systems might use different versions of the same program, but they can't use either programs or floppy disks intended for the other computer. This is typically the situation between IBM and Macintosh computers. An IBM machine can, for example, use the IBM version of Microsoft Word, and the Macintosh can use the Macintosh version of Microsoft Word, but neither computer can use the version intended for the other.

This book describes how MS-DOS works on all IBM and IBM-compatible machines.

PREVIOUS NEXT

What Can You Do with MS-DOS?

MS-DOS coordinates the operation of the computer for your application programs. That's valuable—essential, really—but MS-DOS has much more to offer. You can use MS-DOS itself, controlling it with instructions called *commands*, to manage your files, control the work flow, and perform useful tasks that might otherwise require additional software.

For example, MS-DOS includes a program that lets you create and revise files of text. Although it's not a word processor, the MS-DOS Editor is fine for short memos and lists. Using it, you can write short documents in less time than it might take using your word processing program.

You can tailor MS-DOS to your specific needs by creating powerful commands made up of other MS-DOS commands, and you can even create your own small applications. For example, this book shows you how to create a simple file manager—a program that lets you search a file for specific information—using nothing but MS-DOS commands.

MS-DOS versions 4 through 6 also include a separate program, called the *Shell*, that lets you choose commands and files from on-screen lists called *menus*. If you want, you can use the Shell for routine work, dispense with it and work directly with MS-DOS, or move freely between MS-DOS and the Shell as your work requires.

Your knowledge of MS-DOS can range from just enough to use a single application program to mastery of the full range of capabilities in the latest version. But no matter how far you go, you needn't learn to program. It's all MS-DOS, and it's all in this book.

PREVIOUS NEXT

Chapter Summary

This quick tour of MS-DOS may have introduced several new terms and concepts. Here are the key points to remember:

- A working computer system needs both hardware (equipment) and software (programs).
- MS-DOS (the Microsoft Disk Operating System) coordinates the operation of all parts of the computer system.
- A file is a collection of related information stored on a disk. Most of your computer work will involve files.
- Besides running your application programs, MS-DOS is valuable in its own right.

The next chapter starts you off at the keyboard.

PREVIOUS NEXT

Chapter 2: Starting MS-DOS

Now that you have been introduced to a number of the things that MS-DOS does for you, it's time to start your system and do something. Whenever you start your computer, whether it is to use a word processor, an accounting program, or MS-DOS itself, you begin by *loading* MS-DOS into the computer's memory, its workplace. Loading the MS-DOS program and starting it running is sometimes called "booting the system" or "booting the disk." This term is borrowed from the phrase "pulling yourself up by your bootstraps" because MS-DOS essentially pulls itself up by its own bootstraps, loading itself from disk into memory, where it then waits for a command from you.

The examples from here on assume that you have a computer with a hard disk, that your system is set up to use MS-DOS, and that you are familiar with your computer's control switches. If you need to install a more recent version of MS-DOS on your hard disk, refer to Appendix A. (If your computer does not have a hard disk and you're using MS-DOS from floppy disks, have your usual startup disk ready.)

Starting the System

When you use MS-DOS from a hard disk, the MS-DOS program must be copied into the computer's memory from the hard disk (usually known to MS-DOS as drive C). All you need to do before starting the system is make sure the latch on drive A (the floppy disk drive) isn't closed; otherwise, the system will try to load the MS-DOS program into the computer's memory from the floppy disk in drive A.

If you're not using a hard disk, the MS-DOS program must be copied into the computer's memory from the floppy disk in drive A. Open the latch of drive A (usually either the left-hand or the upper floppy disk drive) and put in the floppy disk you use to start MS-DOS—called the *system disk* in this book—with the label up and away from the machine. If you're using a 5.25-inch disk, close the latch.

Turn on the system. The computer seems to do nothing for several seconds, but this is normal. Each time you turn on the power switch, the computer checks its memory and all attached devices to be sure everything is working properly. The system beeps after it has made sure that all is well, the drive lights flash, and the computer begins loading MS-DOS into memory. As soon as the program is loaded, MS-DOS is running and ready to go to work.

A First Look at MS-DOS

Both MS-DOS and the computers it runs on have evolved in the years since MS-DOS and the IBM PC were introduced. The examples in this book are designed to work correctly with your computer and any release of MS-DOS that supports the commands described, but there are variations in the way computers are set up and in the ways MS-DOS can be organized, installed, and presented to the person who uses it.

One highly visible difference is how MS-DOS looks once it is loaded into memory and ready for you to command. The following sections describe the major variations, one of which should explain what you see on your screen when you start the system.

Many computers today are sold with Microsoft Windows already installed on the hard disk. In some cases, the system is configured so that Windows starts automatically each time the computer is turned on. If this is the case, you'll see a screen that looks something like the one shown in Figure 2-1.

Figure 2-1: The opening (Program Manager) screen of Microsoft Windows.

Windows requires MS-DOS to run, so you can leave Windows temporarily to use this book. To leave Windows, press the Alt key, then F, then X; when Windows displays a box in the middle of the screen telling you that this will end your Windows session, press Enter.

The screen clears, and now MS-DOS is in control. It displays the following:

```
C:\>_
```

or it displays

```
C:\WINDOWS>_
```

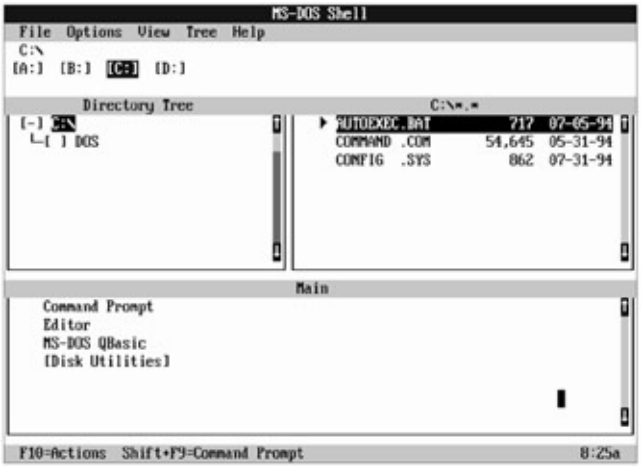
You're ready to start using MS-DOS. Go on to the heading "The System Prompt" on page 16.

Opening with a Startup Menu

If version 6 is installed on your machine, when you start it you might see a screen titled *MS-DOS 6 Startup Menu.* Below the title are two or more numbered choices (such a list of choices is called a *menu*); menu choices might include *Normal* or *Start network.* Below the menu is the line *Enter a choice* followed by the number of the default choice. This screen lets you choose different ways of starting your system; you can't hurt anything no matter how you start it, so press Enter to choose the default menu item. (The default might be chosen for you after a few seconds.) After you press Enter, you'll see one of the startup variations described in the next four sections.

Opening with the MS-DOS Shell

If you're using MS-DOS version 4 or later, your system might be set up to start with the MS-DOS Shell. If so, you'll see a display like this one:



This illustration shows the opening screen of the version 6.0 Shell as it appears in the form called *text mode,* in which the display is made up of letters, numbers, lines, brackets, and other text characters rather than graphic images such as file folders and arrow-shaped mouse pointers. Your Shell screen might differ in some details.

Chapter 11, "The MS-DOS Shell," shows you how to use the Shell. Because the Shell is optional, however, and because you sometimes leave it to use parts of MS-DOS, the remainder of this chapter—indeed, much of this book—takes you outside the Shell. This should help you become comfortable with MS-DOS itself and should enable users of all versions of MS-DOS to benefit from the examples.

If you see the opening Shell screen, leave the Shell for now by pressing the F3 key. MS-DOS responds by clearing the screen and displaying this:

```
C:\>_
```

or this:

```
C:\DOS>_
```

(If your computer doesn't have a hard disk, you see A:\> instead of C:\>.)

Now you have a direct line to MS-DOS. Go on to the section called "The System Prompt" on page 16.

Opening Without Windows or the MS-DOS Shell

On many systems, regardless of the version of MS-DOS you use, MS-DOS starts out by rapidly displaying some brief messages. In version 6.0, for example, the first message you'll see looks something like this:

```
Starting MS-DOS...
```

If you see such a message—or any others that don't require responses from you—you can assume that MS-DOS is settling in properly.

In a startup routine like this, MS-DOS ends by displaying this:

```
C:\>_
```

or this:

```
C:\DOS>_
```

or this (if you don't have a hard disk):

```
A:\>_
```

and waits for your command. If your startup ends like this, MS-DOS is ready to go to work. Go on to the section called "The System Prompt" on page 16.

Opening with Date and Time Requests

Regardless of the version you use, MS-DOS checks for the correct date and time as part of its standard startup routine. If your computer contains a battery-powered internal clock, MS-DOS checks the clock for the information it needs. If your computer does not have a clock, however, the first thing that MS-DOS does is ask you for the date and time. First, it asks for the date:

```
Current date is Tue 01-01-1980
Enter new date (mm-dd-yy): _
```

If you see this message, press the Enter key for now, even though the date isn't Tuesday, January 1, 1980. MS-DOS requests the time next:

```
Current time is 0:01:30.00a
Enter new time: _
```

Again, just press Enter; you'll see how to set or change both the date and time later in this chapter. MS-DOS should now display a message showing the version you're using, and then end with a display like this:

```
C:\>_
```

or this:

```
C:\DOS>_
```

or this (if you don't have a hard disk):

```
A:\>_
```

Go on to the section called "The System Prompt" on this page.

None of the Above

In the years since MS-DOS appeared, many companies have developed shell programs and more sophisticated software that provide alternative ways to use a computer and manage files and applications. The MS-DOS Shell in versions 4 and later is an example of such a program. Microsoft Windows is another.

If none of the preceding descriptions match what you see when you start your computer, check the documentation that came with your computer, or ask the person who set up your system whether a shell or other special program has been installed and, if so, how you can leave it temporarily to use this book.

The System Prompt

The C:\> (or A:\> if you're not using a hard disk) is called the *system prompt* or *command prompt,* because the system program (MS-DOS) is prompting you to type a command. At this point, MS-DOS is at what is often called *command level,* because it's ready and waiting for a command.

The system prompt identifies the *current drive,* the drive where MS-DOS looks for a file. MS-DOS identifies your drives by letter. On a system with one floppy disk drive and one hard disk, the floppy disk drive is identified as both A and B, the hard disk drive as drive C. On a system with two floppy disk drives, the left-hand or upper drive is usually drive A, the right-hand or lower drive is usually drive B.

When MS-DOS is loaded from the hard disk (drive C), MS-DOS assumes drive C is the current drive, and the initial system prompt is C:\> or C:\DOS>. If you're not using a hard disk, MS-DOS is loaded from drive A; MS-DOS assumes that drive A is the current drive, and the initial system prompt on your system is A:\>.

This book contains many examples for you to try. With a few exceptions, the examples show the system prompt as C:\> because that is the normal system prompt on a computer with a hard disk and version 6 of MS-DOS. If you're not using a hard disk, proceed with the examples, but bear in mind that where you see C:\> in the book, you will see A:\> on your screen.

PREVIOUS NEXT

Entering MS-DOS Commands

For the first few commands you enter in this session, you need only the standard typewriter keys on the keyboard. Three of those keys, Enter, Backspace, and Up arrow, are shown on the keyboards in Figures 2-2 and 2-3 and are worth separate mention.

Figure 2-2: The Backspace, Enter, and Up arrow keys on the early PC-compatible keyboard.

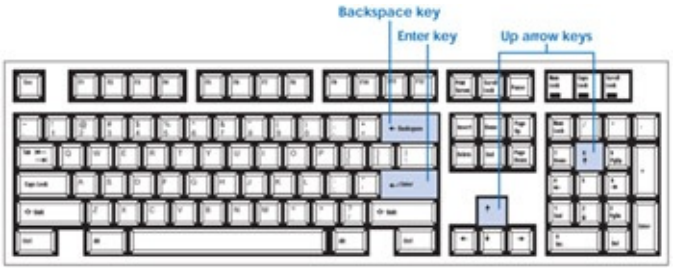


Figure 2-3: The Backspace, Enter, and Up arrow keys on the enhanced PC-compatible keyboard.

The Enter Key

The Enter key is labeled with a bent left arrow (↲), the word *Enter,* or both. Like the return key on a typewriter, it marks the end of a line. In general, MS-DOS doesn't know what you have typed until you press Enter, so remember: End a command by pressing the Enter key.

The Backspace Key

The Backspace key is labeled with a left arrow (←), the word *Backspace,* or both. It erases the last character you typed; use it to correct typing errors.

The Up Arrow Key

The Up arrow key is labeled with an upward-pointing arrow (↑). It is located on the 8 key in the numeric keypad, which resembles a calculator, on all IBM-compatible keyboards and also appears in the set of four "direction" keys to the left of the keypad on enhanced keyboards. The Up arrow key is often used to move a highlight on the screen, but in MS-DOS versions 5 and 6 it also lets you repeat a command, as you'll see in a moment.

Getting Started

At this point, MS-DOS should be displaying the system prompt, followed immediately by a blinking underline. This underline is the *cursor.* It shows where MS-DOS will display whatever you type next. It also tells you that MS-DOS is waiting for you to type something. It's time to put MS-DOS to work.

The MS-DOS commands you'll try in this chapter are easy to use and remember, so no special preparation is needed. If you have version 5 or later of MS-DOS, however, you have an "extra"—a small program named Doskey that you can load into your computer's memory and use with MS-DOS to make some tasks more efficient. You can try Doskey in this chapter, so if you have version 5 or later, type this:

```
C:\>doskey
```

and press the Enter key.

Note If you're using MS-DOS from floppy disks and see the message *Bad command or file name,* don't worry. The floppy disk in drive A doesn't include the part of MS-DOS needed to carry out your command. Just ignore Doskey for now; you'll soon be able to use it without a second thought.

Checking on Your Version of MS-DOS

Some of the examples in this book assume you know which version of MS-DOS you're using. The easiest way to find out about MS-DOS is to ask MS-DOS itself. Whether you know your MS-DOS version or not, try out the Ver (short for version) command. Type this:

```
C:\>ver
```

and press the Enter key.

MS-DOS responds by displaying a message that identifies the version. The exact wording depends on the computer and version of MS-DOS you have. In the Microsoft release of version 6.20, for example, the message you see is

```
MS-DOS Version 6.20
```

Keeping Track of the Date and Time

It's important to know which version of MS-DOS you use, but it's more important to know your computer keeps the correct date and time. The computer has an electronic clock that keeps time to the hundredth of a second. MS-DOS uses this clock to keep track of both the time of day and the date.

In some computers, the clock doesn't run when the system is shut off, so each time you start the system MS-DOS sets the date to January 1, 1980 (01-01-1980) and sets the time to midnight (0:00:00.00 or 12:00:00.00a). That's why, on systems without a battery-powered clock of some type, you see MS-DOS prompting for the correct date and time at startup.

If your system doesn't keep the date and time current, and you just press Enter in response to the date and time prompts when you start the system, MS-DOS assumes that it's midnight on January 1, 1980. Even though you might have been advised (for simplicity) to skip setting the correct date and time earlier in the chapter, it really isn't a good habit to form because MS-DOS marks each file you create or change with the current time and date. Such information is useful, so it's a good idea to set the correct date and time—if necessary—each time you start the system.

Checking or Changing the Date

To check or change the date, you use the MS-DOS Date command. Type this:

```
C:\>date
```

and press Enter. MS-DOS responds like this (you probably see a different date):

```
Current date is Mon 11-21-1994
Enter new date (mm-dd-yy): _
```

The cursor now follows the *Enter new date* request. Such a request is called a *prompt;* MS-DOS frequently prompts you to enter information so that you don't have to memorize operating procedures.

To enter the date, you type the numbers that represent the month, day, and year, separated by hyphens, and then you press the Enter key. You don't have to type the day of the week; MS-DOS figures out the day for you.

For this example, set the date to January 5, 1995, by typing the information you see at the top of the next page. (Be sure to press Enter after the last number.)

```
Current date is Mon 11-21-1994
Enter new date (mm-dd-yy): 01-05-95
```

You can also use a slash(/) or a period to separate the numbers. Whichever you

Note use, if you don't do it exactly right (in other words, in a way that MS-DOS recognizes), MS-DOS displays *Invalid date* and prompts you to try again. If you make a mistake or enter the wrong date, don't be alarmed. As you'll soon see, it's easy to fix such errors.

Check the date again to be sure MS-DOS changed it for you. If you don't have version 5 or later, you repeat your last command by typing it again. Do so now; MS-DOS should respond with its normal date display.

If you have version 5 or later, there's an easier way to repeat a command: Press the Up arrow key once. MS-DOS displays this:

```
C:\>date_
```

There's the Date command you just typed. Remember when you entered a Doskey command a few pages ago? Doskey is a program that keeps track of each command you type in a special area of memory. After you have started Doskey, you can recycle previous commands by pressing the Up arrow key, as you just did.

Press the Enter key, and MS-DOS responds just as if you had typed the Date command:

```
Current date is Thu 01-05-1995
Enter new date (mm-dd-yy): _
```

By pressing just two keys, you have repeated your last MS-DOS command.

You'll correct the date in a moment, but first try the following exercise to see how easily you can fix typing errors.

Backspacing to Correct Typing Errors

Try out the Backspace key. Type some characters, such as the following, at random, but don't press Enter:

```
Current date is Thu 01-05-1995
Enter new date (mm-dd-yy): w710273_
```

This isn't a valid date. If you were to press Enter now, MS-DOS would display the message *Invalid date* and ask you to try again. Correct your typing "error" by pressing the Backspace key until all the characters are erased and the cursor is back to its original position, just to the right of the colon. The screen looks exactly like it did before:

```
Current date is Thu 01-05-1995
Enter new date (mm-dd-yy): _
```

This time, type the correct date and press Enter—for example,

```
Current date is Thu 01-05-1995
Enter new date (mm-dd-yy): 4-19-95
```

for April 19, 1995.

Checking or Changing the Time

Just as you can control the date with the Date command, you can check or change the time with the MS-DOS Time command. If your computer has an internal clock/calendar that keeps the date and time current even when the system is turned off, you probably won't have a lot of use for either Date or Time, but they can still come in handy—when the time changes with daylight saving time, for example, or when you want to know what day of the week a certain date falls on.

Once you've seen the Date command, the Time command looks quite familiar. To try it, type this:

```
C:\>time
```

MS-DOS displays its version of the time and prompts for a new time:

```
Current time is 8:22:33:55a
Enter new time: _
```

If MS-DOS displays the correct time and you don't want to tamper with it, press Enter without typing a response. If the time is incorrect, or if you feel like experimenting, type the time in the appropriate format for your version of MS-DOS, as described below:

MS-DOS Version	Time Format	Examples
1 through 3	24-hour clock	8:30 (before noon) or 20:30 (after noon)
4 and later	12-hour or 24-hour clock	8:30a and 8:30p or 8:30 and 20:30

With versions 1 through 3, for example, you would type

```
Current time is 8:22:33:55a
Enter new time: 13:15
```

to set the time to 1:15 in the afternoon. With versions 4 and later, you could also type this:

```
Current time is 8:22:33:55a
Enter new time: 1:15p
```

Versions 4 and later accept either form.

If you've changed to an incorrect time, reset it before continuing: If you have version 5 or later, press the Up arrow key to recall the last command. If you don't have version 5 or later, type *time.* Now type the correct time and press Enter to carry out the command.

Changing the Current Drive

You can change the current drive simply by typing at the prompt the letter of the new drive, followed by a colon. For example, try changing the current drive to B.

You'll need a floppy disk, so either find one that you have used before or use one of your MS-DOS floppy disks (be careful with it).

If you have one floppy disk drive, insert the floppy disk in the drive with the label up and away from the machine. If you have two floppy disk drives, insert the floppy disk in the drive that does not currently contain your MS-DOS startup disk. Now type this:

```
C:\>b:
```

If you have one floppy disk drive, MS-DOS displays this message:

```
Insert diskette for drive B: and press any key when ready
```

The floppy disk is already in the drive, so press a key. MS-DOS responds:

```
B:\>_
```

Now the system prompt is B:\>, confirming that MS-DOS will look in drive B unless told otherwise.

If you're using a hard disk, change the current drive back to drive C by typing the following:

```
B:\>c:
C:\>_
```

The system prompt returns to C:\>.

If you're not using a hard disk, change the current drive back to drive A by typing the following:

```
B:\>a:
A:\>_
```

The system prompt returns to A:\>.

Printing What's on the Screen

The screen shows you a record of your commands and the responses from MS-DOS; it normally shows a maximum of 25 lines. When all the lines are filled, each additional line causes the entire screen to shift up, or *scroll,* to make room for the new line at the bottom; the top line disappears from view.

Because a copy of what is on the display is often useful, MS-DOS makes it easy to print what's on the screen. Locate the Print Screen key, which is labeled with an abbreviation like PrtSc or Print Scrn (or the key name is spelled out) depending on the keyboard you're using. Make sure your printer is turned on, hold down the Shift key, and press Print Screen. (This combination is referred to in text as Shift-Print Screen.) Each line of the screen is printed.

Note If Shift-Print Screen does not produce a printed copy of what's on the screen, you might need to help your printer understand the instruction. With most Hewlett-Packard LaserJet printers, for example, you press the button on the printer labeled ON LINE, press the FORM FEED button to print the page, and then press ON LINE again to return the printer to its earlier status. If necessary, check the documentation that came with your printer.

PREVIOUS NEXT

Clearing the Screen

Sometimes, when the screen is filled with commands and responses, you might want to clear it before continuing with your work. You can erase everything on the screen with the Clear Screen (Cls) command. Try it by typing this:

```
C:\>cls
```

The screen is cleared, except for the system prompt in the upper left corner.

PREVIOUS NEXT

Turning the System Off

If MS-DOS is displaying the system prompt, all you have to do to shut the system down is turn off the power switch. You can do it anytime, except when the light on a disk drive is on; turning the power off while a drive is in use can cause you to lose the data on the disk. (If you're using an application program and decide to shut the system down, first follow the program's instructions for saving your work and quitting. When the program returns you to the system prompt, you can shut down without risking loss of data.) Some devices attached to your system may have special requirements for shutting down, such as a specific sequence in which they should be turned off. Be sure you know any special instructions for the devices attached to your system.

After you shut the system down, be sure to remove any floppy disks you are using and store them where they will be safe. You can remove your floppy disks before you turn off the power, provided no disk drive is in use.

PREVIOUS NEXT

Chapter Summary

You have completed your first session with MS-DOS. It wasn't very long, but you started the system, entered a few MS-DOS commands, and printed what was on the screen. These are the key points:

- You control MS-DOS by typing commands.
- MS-DOS doesn't know what you have typed until you press the Enter key.
- The Backspace key erases the last character you typed.
- The system prompt tells you that MS-DOS is at the command level, ready to accept a command from you.
- The letter in the system prompt identifies the current drive; you can change the current drive by typing the new drive letter, followed by a colon.
- You can check to see what your version of MS-DOS is with the Ver (for version) command.
- The computer keeps track of the date and time. You can also set them with the Date and Time commands.
- In versions 5 and later, Doskey helps you repeat commands quickly and easily.
- Pressing Shift-Print Screen prints the contents of the screen.
- Typing *cls* clears the screen.

PREVIOUS NEXT

Chapter 3: Getting Your Bearings

Overview

When you venture into new territory—a different part of town, a park, a department store, a building you've never visited—you know where you're starting from, why you're there, and pretty much where you want to go. You took your first steps with MS-DOS in the previous chapter. Now it's time to get your bearings—time to begin learning your way around, moving in the direction of one task or another, and calling a halt if you want or need to.

That's what this chapter is all about. It introduces you to the directory of files that MS-DOS keeps on each disk and shows you how to use the special keys on your keyboard. You use these keys to tell MS-DOS to cancel lines or commands, to freeze the display, and to restart MS-DOS itself. If you have version 5 or 6, you're given a closer look at Doskey, which extends your control over the keyboard and enables you to review and repeat commands you've already used.

To try the examples, start MS-DOS as you normally do, even if your system is already running. Especially with version 6, this will ensure that the examples work as described. If necessary, press F3 to leave the MS-DOS Shell. Don't worry about leaving your computer on while you read the text between examples; MS-DOS is patient.

PREVIOUS NEXT

The Directory

Recall from Chapter 1 that information stored on a disk is stored as a file. For every disk you use, MS-DOS automatically keeps and updates a list of all the files you've saved on the disk. This list is called the *directory*.

If you create and save a new file, MS-DOS adds it to the list for that disk. If you revise an old file, MS-DOS keeps track of that too. The directory eliminates the need for you to keep a separate record of everything you save. You can tell MS-DOS you want to see the directory of a particular disk whenever MS-DOS is displaying the system prompt or, as explained in Chapter 11, when you're using the MS-DOS Shell.

The examples in this chapter give you a look at a directory and show you different ways to display information about your files. But before you begin exploring directories, you should know a little about how MS-DOS saves your files.

Whenever you create a file, you give it a descriptive name, called the *file name*, of up to eight characters. If you want, you can add a suffix, called the *extension*, of up to three more letters. The file name and extension help MS-DOS distinguish one file from another and keep information where it belongs.

Whenever you ask MS-DOS to show you the directory of a disk, MS-DOS lists the files it finds, showing each file name (and extension, if there is one). It also shows you the size of each file, in units called *bytes*, and gives the date and the time the file was created or last changed.

Many people have heard of a byte but aren't quite certain what it is. The easiest way to think of a byte is as the amount of storage required to hold one character in computer memory or on disk. Here are a few familiar items and their sizes, in bytes: the letters *abcd*, 4 bytes, 1 byte per letter; the words *United States*, 13 bytes—spaces count; a double-spaced, typewritten page, 1500 bytes; this book, approximately 1,000,000 bytes.

Depending on the size and type of drives you use (hard disk, floppy disk, or both), a floppy disk can hold from 362,496 to 2,923,620 bytes, and a hard disk can hold anywhere from 20,000,000 to 1,200,000,000 bytes—even more. For convenience, such large quantities are usually given in kilobytes (KB), megabytes (MB), or gigabytes (GB). One kilobyte equals 1024 bytes, one megabyte equals 1024 kilobytes, and one gigabyte equals 1024 megabytes, so disk capacity can range from 360 KB to 1200 MB (1.2 GB) or more.

A Special Kind of Directory

Left to itself, MS-DOS doesn't group files logically as you would, categorizing them by type, content, or any other characteristic. MS-DOS doesn't, for example, keep all program files in one place and all documents in another. To MS-DOS, a file is just a file, and as it keeps track of the files you create, it keeps adding their file names to the disk directory.

The main disk directory cannot simply grow without limit, however, even though a large hard disk can hold thousands of files. To help control growth and give you a way to keep track of similar groups of files, MS-DOS includes commands that let you divide disk storage space into smaller, more manageable areas called *subdirectories*. Subdirectories are the disk equivalent of dividers in a file drawer.

Although you won't be working with subdirectories until later in this book, you do need to know that MS-DOS versions 4 and later normally install themselves automatically, on a hard disk, in a subdirectory named DOS. Earlier versions of MS-DOS, though they did not install themselves, were often placed in a subdirectory by the people who installed them. The tradition of keeping MS-DOS in its own subdirectory is, in fact, so commonplace that this book assumes you have a DOS subdirectory if you use MS-DOS from a hard disk.

If you have a hard disk, you can easily check for a DOS subdirectory. If your system prompt looks like this,

```
C:\DOS>_
```

MS-DOS itself is telling you that you have a DOS subdirectory and it is looking at the subdirectory right now.

If your system prompt looks like this,

```
C:\>_
```

you need to check a little further. Use the Change Directory command, which tells MS-DOS to find and focus its attention on the subdirectory you name. Type this:

```
C:\>cd dos
```

If all goes well, MS-DOS turns to your DOS subdirectory, and your system prompt probably changes to C:\DOS>. If MS-DOS can't find a DOS subdirectory, it responds with the message *Invalid directory* and again displays the system prompt.

Note The preceding two examples cover the majority of hard disks equipped with versions 3 through 6 of MS-DOS. If you received the *Invalid directory* message, you can still try the following examples. They'll work just fine, but you should bear in mind that the file names you see might not be the same as those shown in the book.

Chapter 8, "A Tree of Files," shows you how to create, use, manage, and remove subdirectories. The next part of this chapter shows you how to browse through directories and subdirectories, finding specific files they contain.

Displaying a Directory

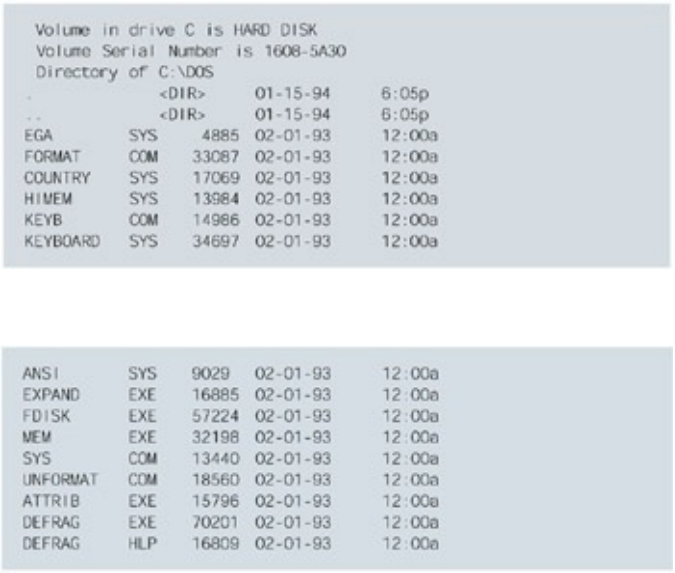
To display the current directory (the directory MS-DOS uses unless you specify otherwise), you simply type *dir*, the name of the Directory command. Type the command and press Enter (remember, if you don't have a hard disk, your system prompt is A:\>):

```
C:\DOS>dir
```

If you have a hard disk and have displayed the DOS directory, the directory probably scrolled off the screen faster than you could read it. There are several ways to handle that, but one of the simplest uses what's called a *command parameter*. As you'll see in the next chapter, a command parameter lets you refine the action of a command. Here, with the Directory command, you can use a parameter typed as */p* to instruct MS-DOS to *pause* after displaying a screenful of directory entries. Try it. If your directory listing was too long to fit on one screen, type this:

```
C:\DOS>dir /p
```

This time, scrolling stops when the screen is full. Press any key to see the remainder of the directory listing, one screenful at a time. Figure 3-1 shows a sampling of the MS-DOS file names you see on a computer with a hard disk and version 6.0 of MS-DOS.



```
 Volume in drive C is HARD DISK
 Volume Serial Number is 1608-5A30
 Directory of C:\DOS
.            <DIR>      01-15-94     6:05p
..           <DIR>      01-15-94     6:05p
EGA      SYS     4885  02-01-93     12:00a
FORMAT   COM    33087  02-01-93     12:00a
COUNTRY  SYS    17069  02-01-93     12:00a
HIMEM    SYS    13984  02-01-93     12:00a
KEYB     COM    14986  02-01-93     12:00a
KEYBOARD SYS    34697  02-01-93     12:00a
```

```
ANSI     SYS     9029  02-01-93     12:00a
EXPAND   EXE    16885  02-01-93     12:00a
FDISK    EXE    57224  02-01-93     12:00a
MEM      EXE    32198  02-01-93     12:00a
SYS      COM    13440  02-01-93     12:00a
UNFORMAT COM    18560  02-01-93     12:00a
ATTRIB   EXE    15796  02-01-93     12:00a
DEFRAG   EXE    70201  02-01-93     12:00a
DEFRAG   HLP    16809  02-01-93     12:00a
```

Figure 3-1: A sample directory display of version 6.0 MS-DOS files.

If you're using floppy disks or a different version of MS-DOS, your directory listing differs—perhaps a little, perhaps a lot. The list might be shorter, for example, or it might show some different names, dates, or times. Regardless, certain names, such as FORMAT, COMMAND, DEBUG, and MODE appear consistently across different versions of MS-DOS. You can check your display for one or two of those names if you want, but the real point of this example is simple: The result of a Directory command is always a list of files stored on the disk.

Note The Directory command is used in examples throughout the book. Unless stated otherwise, the version 6.0 display is shown.

The lines at the top of a directory listing give information about the disk itself and are explained in Chapter 6, "Managing Your Floppy Disks." In versions 5 and later, the last two lines of a directory display show the number of files in the directory, the number of bytes of storage they occupy, and—in the last line—the number of bytes of storage remaining on the disk. (In versions before 5, the directory listing ends with a single line that gives the number of files and the number of bytes available for storage.)

Figure 3-2 shows a sample entry from a directory. The file name is DISKCOPY; note that it is eight letters long, the maximum length of an MS-DOS file name. The next item, COM, is the file's extension. The next item tells you the file's size, and the final two entries give the date and time the file was either created or last changed.

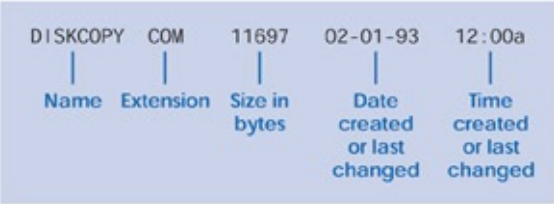


Figure 3-2: A sample directory entry.

PREVIOUS NEXT

Some Important Keys

In the examples in the previous chapter, you used the standard typewriter portion of the keyboard to enter commands. Several other keys have important meanings to MS-DOS. Figures 3-3 and 3-4 show where these keys are located on two common versions of the IBM keyboard. If your keyboard does not have the key used in an example, check your system's documentation for equivalent keys.

Figure 3-3: Special keys on the early IBM PC-compatible keyboard.

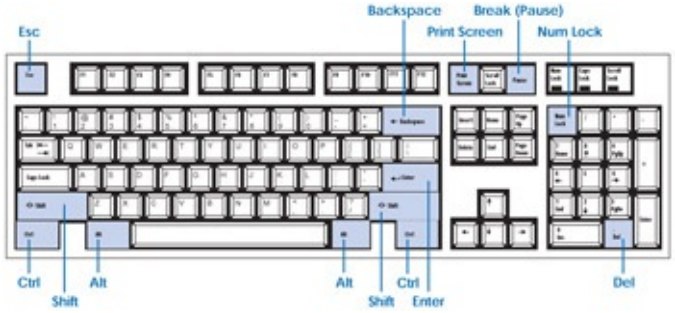


Figure 3-4: Special keys on the enhanced IBM PC-compatible keyboard.

Shift

The Shift keys are labeled with an open arrow, the word *Shift*, or both. Like the Shift keys on a typewriter, they have no effect by themselves; they shift the keyboard to uppercase letters and special characters, such as the dollar sign.

Escape

This key, usually labeled *Esc*, cancels a line you have typed. To see how it works, type several characters (but don't press Enter):

```
C:\DOS>Now is the time
```

To erase this line, you could repeatedly press the Backspace key, but press the Escape key instead:

```
C:\DOS>_
```

If you're using version 5 or later and installed Doskey in Chapter 2, MS-DOS erases the line you typed and moves the cursor back to the system prompt.

If you're using version 4 or earlier, or if you haven't installed Doskey, the result is different:

```
C:\DOS>Now is the time\
```

MS-DOS displays a reverse slash (\) to indicate that the line was canceled and moves the cursor to the next line. MS-DOS doesn't repeat the system prompt, but the cursor indicates it is still ready for you to type a command. Press the Enter key, and MS-DOS displays the system prompt and the cursor on the next line:

```
C:\DOS>_
```

Pressing the Escape key is the quickest way to cancel a line you have typed.

Control

This key, usually labeled *Ctrl*, has no effect by itself, but it is used like the Shift keys to change the effect of pressing another key. The combination of the Control key and some other key is represented in this book by Ctrl- followed by the other key. Ctrl-Break, for example, means "hold down the Control key, and then press and release the Break key." The Control key combinations are described under the heading "Control Key Functions" on page 33.

Numeric Lock

This key, familiarly known as *Num Lock*, does two things. It switches the effect of the keys in the calculator-style numeric pad at the right side of the keyboard back and forth between cursor movement and numbers. On early PC keyboards, it is also used in combination with the Control key to freeze the display. To test the first function, press Num Lock, and then press the 4 key in the numeric pad several times:

```
C:\DOS>444_
```

The keys produce numbers on the screen. Now press Num Lock again and press the same 4 key you pressed before:

```
C:\DOS>444
```

If you're using version 5 or 6 and installed Doskey, MS-DOS moves the cursor to the left one character so that it is under the last number you typed.

If you're using version 4 or earlier, or if you're using version 5 or 6 and Doskey isn't installed, you'll see the following result:

```
C:\DOS>44_
```

Pressing Num Lock a second time switched the keys to their cursor-movement functions. The 4 key is labeled with a left arrow in addition to the number 4; pressing it moves the cursor left, in the direction of the arrow, and (if you're not using Doskey) erases a character just as the Backspace key does. Press Num Lock and the same 4 key again:

```
C:\DOS>444_
```

You switched back to numbers. Press Num Lock one more time to switch back to cursor movement, press Esc to cancel the line, and press the Enter key to return to the system prompt:

```
C:\DOS>444\
C:\DOS>_
```

You won't often use the arrow keys for cursor movement with MS-DOS, but many application programs, such as word processors, require frequent cursor movements. If you have version 5 or 6, you'll also use the arrow keys for giving commands to Doskey.

Break

This key is labeled either *Scroll Lock* and *Break* or *Pause* and *Break*. When labeled *Scroll Lock* and *Break* (as on early PC-compatible keyboards), this key has no effect on MS-DOS by itself, but it is used with the Control key to cancel a command you have entered. If labeled *Pause* and *Break* (as on the enhanced PC-compatible keyboard), this key temporarily halts the display; when used with the Control key, it cancels a command.

Alternate and Delete

The key labeled *Alt* has no effect on MS-DOS by itself, while the *Del* key is used to delete the character above the cursor. Both are used with the Control key to restart MS-DOS.

Print Screen

This key, labeled *Print Screen*, *Prnt Scrn*, *PrtSc*, or some close variation, is used with the Shift or Control key to print the contents of the screen. You used Shift-PrtSc in the previous chapter; you'll use Ctrl-PrtSc and see the difference in a short while. (If you have an IBM PS/2 keyboard, you don't have to press Shift with PrtSc.)

Control Key Functions

Figure 3-5 shows the effects produced by holding down the Control key and pressing another key. You'll probably use these combinations fairly often with MS-DOS, so the next few topics show you examples of each combination. When you are being shown exactly what to type, the names of the keys are separated by hyphens and enclosed in angle brackets to represent pressing a Control key combination. Thus, when you see <Ctrl-Break> in a command, it means "press Ctrl-Break."

Ctrl-Num Lock or Pause	Halts whatever the system is doing until you press another key. Typically used to freeze the display when information is scrolling by too fast or scrolling off the top of the screen. Can also be typed as Ctrl-S (Ctrl plus the letter S).

Ctrl-Break	Cancels whatever the system is doing. Use this when you really don't want the computer to continue what it's doing. Can also be typed as Ctrl-C (Ctrl plus the letter C).
Ctrl-PrtSc	Pressing this key combination once causes MS-DOS to start printing every line as it is displayed; pressing Ctrl-PrtSc a second time stops simultaneous displaying and printing. Can also be typed as Ctrl-P (Ctrl plus the letter P).
Ctrl-Alt-Del	Restarts MS-DOS. This combination is unique; no other keys can be used to do the same thing.

Figure 3-5: Control key combinations.

Before trying the examples, you should also note that MS-DOS displays the Control key as the symbol ^. MS-DOS does not acknowledge all Control key commands on the screen, but when it does, it uses the symbol ^ in combination with a letter. Control-Break, for example, shows on the screen as ^C and can also be typed by holding down the Control key and typing the letter C.

Freezing the Display

As mentioned earlier, MS-DOS lets you temporarily halt the display by pressing the Pause key or, if you have an early PC keyboard, Ctrl-Num Lock. When you do this, the display remains frozen, giving you time to read it. To start the display moving again, you simply press any key. Test this function by typing this to display the directory:

```
C:\DOS>dir
```

When the entries start appearing on the screen, press Pause or Ctrl-Num Lock to freeze the display. Press any key; the display resumes. You can press Pause or Ctrl-Num Lock to stop and start the display as many times as you like, to view displays many screens long.

Canceling a Command

If you enter a command and then change your mind or realize that you meant to enter some other command, you can cancel the command you entered by pressing Ctrl-Break. To test this function, type the Directory command again. This time, however, press Ctrl-Break when MS-DOS begins to display the directory entries.

Here's an example:

```
C:\DOS>dir

 Volume in drive C is HARD DISK
 Volume Serial Number is 1608-5A30
 Directory of C:\DOS
.          <DIR>     01-15-94    6:05p
..         <DIR>     01-15-94    6:05p
```

```
EGA      SYS     4885 02-01-93    12:00a
FORMAT   COM    33087 02-01-93    12:00a
COUNTRY  SYS    17069 02-01-93    12:00a
HIMEM    SYS    13984 02-01-93    12:00a
KEYB     COM    14986 02^C

C:\DOS>_
```

Your display probably would have stopped somewhere else in the directory, but when you press Ctrl-Break, MS-DOS stops what it is doing, displays ^C, and returns to the command level.

Printing and Displaying Simultaneously

In the previous chapter you printed the contents of the screen by pressing Shift-PrtSc. There's another way to print from the screen: Pressing Ctrl-PrtSc tells MS-DOS to start printing everything it displays. MS-DOS continues to print and display simultaneously until you press Ctrl-PrtSc again.

To test this function, make sure your printer is turned on, press Ctrl-PrtSc, and then enter the Directory command:

```
C: \DOS><Ctrl-PrtSc>dir
```

MS-DOS again displays the directory of the system disk, but this time each line is printed as it is displayed. (If you're using a LaserJet or similar printer, you might have to press Ctrl-P instead.)

The directory is displayed more slowly than when you use the Directory command alone because MS-DOS waits until a line is printed before displaying and printing the next line. You can cancel the Directory command before the complete directory is printed by pressing Ctrl-Break. Remember to press Ctrl-PrtSc as well, to end the simultaneous displaying and printing.

If you want to print something without printing the command that creates the display, type the command, press Ctrl-PrtSc, and then press Enter. For example, when you printed the directory in the preceding example, the Directory command was the first line printed. To avoid printing the command, type this:

```
C:\DOS>dir<Ctrl-PrtSc>
```

Now printing begins with the first line of the directory; the Directory command isn't printed. Cancel the command by pressing Ctrl-Break. Be sure to press Ctrl-PrtSc again to stop printing; otherwise, MS-DOS continues to print everything it displays, even if you go on to an entirely different task.

Shift-PrtSc Versus Ctrl-PrtSc

These two methods of printing from the screen work differently and have different uses. Shift-PrtSc prints everything on the screen and stops. Ctrl-PrtSc, as you just saw, alternates displaying and printing, line by line. If everything you want appears on the screen, use Shift-PrtSc; it's faster. But if you want to print something longer than one screenful, use Ctrl-PrtSc.

Ctrl-PrtSc is better for printing long displays because you press it once to tell MS-DOS to start simultaneous displaying and printing, enter a command—such as the Directory command—to create the display, and then press Ctrl-PrtSc again when you want to stop printing. If you use Shift-PrtSc for printing displays more than one screen long, you have to display the first screen, print it, display the second screen, print it, and so forth until everything you want has been printed.

Repeating Commands with Doskey

Versions 5 and later of MS-DOS include the Doskey command you learned about in Chapter 2. Doskey makes your work with MS-DOS easier and more efficient by letting you repeat MS-DOS commands you've used recently without retyping them.

To help you minimize keystrokes (and the chance of mistyping a command), Doskey lets you use several special keys, among them the arrow and function keys shown in Figures 3-6 and 3-7 on the next page.

Figure 3-6: Keys on the early PC-compatible keyboards that have special meaning to Doskey.

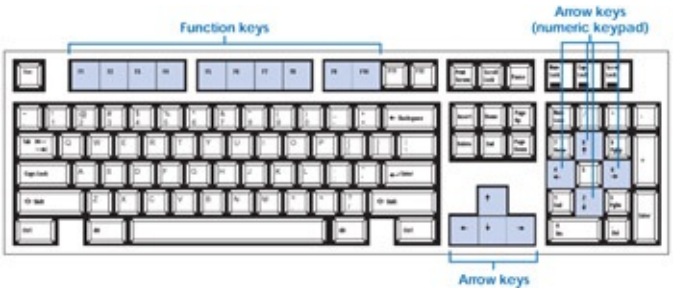


Figure 3-7: Keys on the enhanced PC-compatible keyboard that have special meaning to Doskey.

Note If you're using MS-DOS version 5 or later from floppy disks and saw the message *Bad command or file name* the last time you tried to start Doskey, you need to insert the floppy disk that contains the DOSKEY file. Now that you're familiar with the Directory command, check your Support disk (and others, if necessary) for this

file by typing dir or dir /p. When you see DOSKEY in the directory list, leave the floppy disk containing the file in drive A and retype the doskey command.

The following examples show you some of Doskey's basic features. Recall that you start Doskey simply by typing its name and pressing Enter:

```
C:\DOS>doskey
```

MS-DOS responds:

```
DOSKey installed.
```

As you've seen, Doskey lets you repeat your last MS-DOS command in a simple two-step procedure: Press Up arrow to redisplay the command, and then press Enter to carry it out. But you won't always want to repeat the last command you typed. You might want to repeat one you used several commands ago. Doskey lets you do that, too. To try it, first type the following MS-DOS commands (just press Enter when MS-DOS prompts for the date and time):

```
C:\DOS>date
C:\DOS>time
C:\DOS>ver
C:\DOS>dir
C:\DOS>cls
```

Now you've entered some commands Doskey can help you find and reuse.

Although you probably wouldn't have much trouble remembering five commands, Doskey can help you keep track of dozens. But when you've typed a number of commands, you can't always remember which ones you typed or in what order. Once you've started Doskey, you can simply press the F7 function key to see a list of your previous commands, in sequence. Try it now. Press F7, and Doskey displays this:

```
C:\DOS>
1: date
2: time
3: ver
4: dir
5: cls
C:\DOS>_
```

The commands are numbered and in the same order you typed them. Press the Up arrow key. Doskey displays the last command:

```
C:\DOS>cls_
```

Press the Up arrow key again, and Doskey replaces *cls* with *dir*, the next-to-last command. Whenever you press the Up arrow key, Doskey recalls and displays the previous command.

Now try some other keys. Press the Down arrow key. The display changes back to this:

```
C:\DOS>cls_
```

The Down arrow key does the opposite of the Up arrow key: It tells Doskey to retrieve the next command (as opposed to the previous command) in the list.

You can take bigger leaps through the list, too, with the Page Up and Page Down keys. Press the Page Up key, and the display changes to this:

```
C:\DOS>date_
```

Page Up tells Doskey to recall the first command in the current list. And, predictably, when you press the Page Down key, the display changes to this:

```
C:\DOS>cls_
```

which is the last command in the current list.

You can also request a specific command by entering its line number. Press the F9 key. Doskey displays this:

```
C:\DOS>Line number: _
```

This time, request the Ver command, which is number 3 in the list. Type this:

```
C:\DOS>Line number: 3
```

and press Enter, and this appears:

```
C:\DOS>ver_
```

Press Enter. MS-DOS carries out the Ver command and displays the system prompt.

Everybody from beginners to advanced users of MS-DOS can find many uses for Doskey. These examples showed a few quick ways to use it; later chapters show you how to use Doskey to tailor MS-DOS to the way you work.

PREVIOUS NEXT

Restarting the System

Suppose you find yourself in a situation where your computer is not responding as you think it should, or it complains about something you don't know how to handle, or you decide it would be best to scrap what you're doing and start over from the beginning. You don't have to turn the power switch off and on to restart your system; you can do it by pressing Ctrl-Alt-Del.

Try it. If you're using MS-DOS from floppy disks, check that your normal Startup disk is in drive A. If you have a hard disk, be sure drive A is open.

Now hold down both Ctrl and Alt and press Del.

The screen clears, the drive lights blink, the system beeps, and MS-DOS is loaded just as it was when you turned the power on. Restarting with Ctrl-Alt-Del takes less time, though, because the computer doesn't test all its devices and memory as it does whenever you switch the power off and on. If the MS-DOS Shell appears, press F3 to leave it.

PREVIOUS NEXT

A Short Diversion

The system prompt is an economical way for MS-DOS to show you the current drive and directory and to let you know that you can enter a command. But the combination of the current drive and directory and the greater-than sign (>) is only one possible system prompt. The MS-DOS Prompt command lets you change the system prompt to almost anything you want.

For example, you might prefer a more courteous machine. Type the following and press Enter (<space> means press the Spacebar):

```
C:\DOS>prompt May I help you?<space><Enter>
```

Now the system prompt isn't quite so cryptic:

```
May I help you? _
```

Each time MS-DOS returns to the command level, it displays this polite phrase. Try it by pressing the Enter key once or twice to cause MS-DOS to display the system prompt again. Although your new prompt looks significantly different from C:\> or C:\DOS> (and actually conveys less information), the meaning is the same: MS-DOS is at the command level, ready for you to enter a command.

To see how much you can cram into the system prompt, type the following example as a single line (as before, <space> means press the Spacebar). Although the example is shown on two separate lines and won't fit on one line on screen anyway, don't press the Enter key until you come to <Enter> at the end of the second line:

```
May I help you? prompt The time is<space>$t$_The date is<space>
$d$_The current disk is<space>$n$_Your command:<space><Enter>
```

Now the system prompt is three lines of data followed by a request for a command:

```
The time is 16:26:03.54
The date is Thu 01-05-1995
The current disk is C
Your command: _
```

You would probably quickly tire of all this, but the exercise shows how much flexibility MS-DOS gives you. You don't have to take advantage of it all, but the possibilities are there if you want them.

To return the system prompt to a more normal form, simply type the following Prompt command:

```
The time is 16:26:03.54
The date is Thu 01-05-1995
The current disk is C
Your command: prompt $p$g
```

Your system prompt changes to the familiar C:\> or C:\DOS>.

Chapter Summary

This chapter has informed you of the following:

- Each disk has a directory that lists the name, extension, and size of each file, and the date and time the file was created or last changed.
- You can see the directory by typing *dir* and pressing Enter.
- The Escape key cancels a line you have typed.
- Pause or Ctrl-Num Lock freezes the display. Ctrl-S has the same effect.
- Ctrl-Break cancels a command. Ctrl-C has the same effect.
- Ctrl-PrtSc turns simultaneous displaying and printing on and off. Ctrl-P has the same effect.
- Ctrl-Alt-Del restarts MS-DOS.
- The Doskey command helps you display, choose from, and repeat commands you've already used.

Now that you're more familiar with the keyboard, the next chapter gives you a closer look at floppy disks and files.

PREVIOUS NEXT

Chapter 4: A Look at Files and Floppy Disks

The computer's memory is temporary storage; it is cleared each time you turn off the computer. The only way you can save data permanently is to store it in a file on a disk. When MS-DOS needs data that is stored in a file, it reads the data from the disk into memory. If you change the data and want to keep the changed version, you must store the revised version on disk before turning off the system.

Types of Files

In general, a file contains either a program or data. A *program* is a set of instructions for the computer. *Data* is the text and numbers, such as a project proposal, a table of tax rates, or a list of customers, that the program needs to do your work.

Two types of files are important to your work: text files and command files. They are quite different, so it's important to look more closely at the kind of information these files contain and at how the files are used.

Text Files

Text files are data files that contain characters you can read (everyday letters, numbers, and symbols). Word processing programs store their documents in text files, as does the MS-DOS Editor. Many files you use in your work with the computer—and all the files that you will create and use in this book—are text files.

The definition of a text file may seem self-evident at first, but it actually introduces you to an important characteristic of computer information storage. Your computer keeps information in two very different forms: One is text, the characters contained in text files; the other is machine-readable code, which looks meaningless to most people but is quite meaningful to computers.

Command Files

Command files contain the instructions MS-DOS needs to carry out commands. MS-DOS command files can be programs, such as DISKCOPY.COM, or, as you will see in Chapter 14, "Creating Your Own Commands," they can be a series of commands that you put together to perform a specific task and store in a file.

Not all MS-DOS commands are stored in separate command files, however. Some commands, such as the Directory command, are built into the main body of MS-DOS. When you load MS-DOS into memory, you load these commands with it. When you want to use these commands, MS-DOS has them on tap for immediate use—it does not need to look up a separate command file to carry them out.

These built-in commands are called *permanent*, or *internal*, commands. In contrast, the commands that are kept in command files until they are requested by you are called *temporary*, or *external*, commands. When you use a permanent command, you simply request the command, and MS-DOS carries it out. When you use a temporary command, MS-DOS must load the command file from disk into memory before it can carry out the command.

An application program, such as a word processor, is also stored in a command file; it stores your work, such as documents, in data files.

How Files Are Named

No matter the type of file, each file must have a file name. Recall that a file name can be up to eight characters long. You can use almost any character on the keyboard when you name your files, but it's a good idea to give your files names, such as BUDGET or SALESRPT, that describe their contents.

To identify a file more completely, a three-character suffix called the file *extension* can be added to the file name; this suffix is separated from the file name by a period. So that you and MS-DOS can tell your files apart, each file on a disk must have either a different name or a different extension; REPORT.JAN and REPORT.FEB, for example, are different files to MS-DOS, even though their file names are the same.

Specifying the Drive

When you name a file in a command, MS-DOS needs to know which drive contains the disk with the file on it. If you don't specify a drive, MS-DOS looks on the disk in your current drive (the drive letter shown in the system prompt). If the disk containing the file is not in the current drive, you can precede the file name with the letter of the drive and a colon. For example, if you specify the file as *b:report.doc*, MS-DOS looks for it on the disk in drive B.

PREVIOUS NEXT

Preparing for the Examples

The following pages show a number of examples to help you become more comfortable with files and floppy disks. With MS-DOS, as with most other computer programs, doing is often the easiest and most effective way of learning.

If you have a hard disk, make sure the latch on drive A is open, turn on or restart the computer, and go through the startup routine until you see the system prompt (C:\> or C:\DOS>). If the MS-DOS Shell starts automatically, startup instructions from here on assume that you press F3 to leave the Shell. You'll be working with files in the DOS subdirectory, so change to it if necessary by typing this:

```
C:\>cd \dos
```

Go on to the next heading.

If your system doesn't have a hard disk, start or restart your computer and go through the startup routine until you see the A:\> prompt. The examples use several external MS-DOS commands, beginning with DISKCOPYCOM. Use the Directory command to check your MS-DOS floppy disks for this file, and start with the floppy disk containing DISKCOPY.COM in drive A. Use the Directory command whenever MS-DOS responds *Bad command or file name* because it cannot find the command file it needs for a particular example.

Don't Worry About Memorizing

You'll use several commands in this chapter, but you needn't remember exactly how to use each one; all the commands are described in more detail in the remaining chapters of the book. The purpose of this chapter is to introduce you to files and floppy disks.

PREVIOUS NEXT

Qualifying a Command

Up to now, most commands you have entered have consisted of a single word or abbreviation, such as *time* or *dir*. Many commands, however, let you add one or more qualifiers to make the action more specific. These qualifiers are called *parameters*.

Some commands require parameters; others allow you to add parameters if you want. The Directory command, for example, does not require parameters, but it lets you tailor a command with specifications such as the name of a particular file you want to see. You'll use some parameters in the following examples; descriptions of commands in later chapters show the commands' parameters, both required and optional.

PREVIOUS NEXT

Displaying Specific Directory Entries

In the previous chapter you used the Directory command to display the directory entries of the files in your DOS subdirectory or on your startup disk. You can display the directory entry of a single file or the directory entries of a set of files by adding a parameter to the Directory command.

Displaying the Directory Entry of a Single File

To display the directory entry of a specific file, you simply type the file name (and its extension, if there is one) after the command name. For example, the command to copy the contents of one floppy disk to another is called Diskcopy. Its command file is DISKCOPY.COM. To display the directory entry for only DISKCOPYCOM, type the following command. (If you're not using a hard disk, the floppy disk with DISKCOPY.COM should be in drive A; remember that your system prompt is A:\>, not C:\DOS>.)

```
C:\DOS>dir  diskcopy.com
```

MS-DOS displays only the directory entry of the file you specified (don't worry if you see a different size, date, or time):

```
Volume in drive C is HARD DISK
Volume Serial Number is 1608-5A30
Directory of C:\DOS
DISKCOPY COM      11879 02-01-93  12:00a
        1 file(s)      11879 bytes
                    36333568 bytes free
C:\DOS>
```

If the file you name isn't on the disk, or if you don't type the file name exactly as it is stored, MS-DOS responds *File not found*.

Displaying the Directory Entries of a Set of Files

What if you remember most of a file name, or the file name but not the extension? MS-DOS helps you out by giving you two wildcard characters, * and ?, that you can substitute for actual characters in a file name. Like wild cards in a poker game, the wildcard characters can represent any other character. They differ only in that ? can substitute for one character, whereas * can substitute for more than one character.

Suppose you remember only that a file's name begins with the letter F. It takes only a moment to check all the files that begin with F.

Use the MS-DOS directory as an example. Type the following command:

```
C:\DOS>dir f*
```

MS-DOS displays the directory entries of all file names that begin with F:

```
Volume in drive C is HARD DISK
Volume Serial Number is 1608-5A30
Directory of C:\DOS

FORMAT   COM    33087 02-01-93   12:00a
FDISK    EXE    57224 02-01-93   12:00a
FASTOPEN EXE    12050 02-01-93   12:00a
FC       EXE    18650 02-01-93   12:00a
FIND     EXE     6770 02-01-93   12:00a
       5 file(s)    127781 bytes
                 36333568 bytes free

C:\DOS>
```

(This listing shows the names of all the MS-DOS files that begin with F. If you're using MS-DOS from floppy disks, your list might differ, but you should see FORMAT.COM or FORMAT.EXE.)

Wildcard characters can simplify the task of keeping track of your files. Chapter 5, "Managing Your Files," includes several examples of using wildcard characters. Now it's time to stop practicing and to create some files of your own.

PREVIOUS NEXT

Preparing a Floppy Disk for Use

Before MS-DOS can store a file on a new floppy disk, it must prepare the floppy disk for use. This preparation, in which MS-DOS writes certain information for its own use on the floppy disk, is called *formatting*. You tell MS-DOS to do this formatting with the Format command. You'll need two formatted floppy disks for the examples in this book. Now is a good time to format them, so get out two blank floppy disks and two blank labels before proceeding. If you have bought floppy disks that have been formatted already, there's really no need to format them again. But go ahead and practice formatting the floppy disks. You'll most likely have to format one sometime.

Type the following:

```
C:\DOS>format b:
```

This command tells MS-DOS to format the floppy disk in drive B.

Formatting a floppy disk effectively deletes any files that may be stored on it, so MS-DOS gives you a chance to make sure you haven't put the wrong floppy disk in the specified drive by displaying a message and then waiting for you to type something:

```
Insert new diskette for drive B:
and press ENTER when ready..._
```

If you discover that you put in the wrong floppy disk, no problem: Just take out the wrong one and put in the right one before you press the Enter key.

If you can't find a floppy disk to format, and you want to cancel the command, no problem again: Don't turn the system off; just press Ctrl-Break.

But you do want to format the floppy disk now. If you have one floppy disk drive, place a blank floppy disk in drive A and close the drive latch if necessary. If you have two floppy disk drives, place a blank floppy disk in drive B and then close the latch.

Press Enter. If you have version 5 or later, MS-DOS begins by displaying one or more messages telling you it is checking the disk to see whether it was previously formatted. After these steps, MS-DOS displays a constantly changing message that shows you the progress of formatting the disk:

```
Checking existing disk format.
Saving UNFORMAT information.
Verifying 1.44M

Formatting 1.44M
  8 percent completed.
```

Version 4 of MS-DOS tells you x *percent of disk formatted*. In earlier versions, the message might look like this:

```
Head:   0 Cylinder:    1
```

or it might simply read *Formatting....* In any case, the light on the drive goes on, and MS-DOS begins writing on the floppy disk. When MS-DOS is finished, it tells you *Format complete*. If you have version 4 or later, it then displays the message:

```
Volume label (11 characters, ENTER for none)? _
```

A volume label is a name you give a formatted disk to help identify it and the files it contains. The name can be up to 11 characters long, including blanks, but cannot include certain characters MS-DOS reserves for special uses—characters such as a period (used between a file name and an extension), an asterisk or a question mark (used as wildcard characters), or a forward slash (used when typing command parameters).

You don't have to assign a volume label to a disk, but because MS-DOS displays the volume label at the beginning of any directory listing you request, the few seconds you spend thinking up and typing a volume label when you format a floppy disk can, in the long run, save you time by helping to identify what the disk contains. (If you don't have version 4 or later, you can assign a volume label after formatting with the /V parameter of the Format command; you'll find out about this in Chapter 6, "Managing Your Floppy Disks.")

If MS-DOS is requesting a volume label, assign one to the disk you just formatted. Type a simple but descriptive name, such as this:

```
Volume label (11 characters, ENTER for none)? examples 1
```

and press Enter.

MS-DOS then displays some information about the floppy disk, followed by a final message:

```
1,457,664 bytes total disk space
   1,457,664 bytes available on disk

       512 bytes in each allocation unit
     2,847 allocation units available on disk

Volume Serial Number is 1A2C-13F5

Format another (Y/N)?_
```

The numbers shown are for a 1.44-MB floppy disk. Depending on the type of floppy disk drives you have and the version of MS-DOS you're using, the total disk space in your report might differ—for example, 1,213,952 or 730,112.

The messages about allocation units and the volume serial number are provided by MS-DOS beginning with version 4. Allocation units are groups of bytes used by MS-DOS in storing information; the volume serial number is assigned as part of the formatting procedure. Neither is likely to be significant in your day-to-day use of MS-DOS.

The final message, *Format another (Y/N)?*, means that MS-DOS is now waiting for you to say whether you want to format another floppy disk. Type *y* and press Enter. The message asking you to put the floppy disk in drive B and press Enter is repeated, so go through the same process to format the second floppy disk; name it *examples 2*. When MS-DOS finishes, it asks you again whether you want to format another.

Now type *n* and press Enter. MS-DOS displays the system prompt (C:\DOS>), telling you that the Format command is complete and that MS-DOS is waiting for you to type another command.

You now have two formatted floppy disks. It's time to put one of them to use by creating a file; if you removed the floppy disk you just formatted, put it in your floppy disk drive (drive B if you have two floppy disk drives).

PREVIOUS NEXT

Creating a Text File

An easy way to create a text file is by using the MS-DOS Copy command. As you might guess from its name, the Copy command can be used to make a copy of a file. It can also be used to copy characters from the keyboard into a file.

MS-DOS refers to the parts of your computer, such as the keyboard, display, and printer, as *devices*. To MS-DOS, devices, like files, have names. For example, the keyboard is known to MS-DOS as CON (for CONsole).

You are going to create a file by telling MS-DOS to copy what you type from the keyboard onto the blank floppy disk in drive B.

To create a file named NOTE.DOC on the floppy disk in drive B, type the following example. End each line by pressing Enter; where you see a blank line, press Enter to tell MS-DOS to insert an extra line:

```
C:\DOS>copy con b:note.doc
January 5, 1995

Dear Fred,
Just a note to remind you
that our meeting is at 9.

Jack
```

That's the end of the file. To tell MS-DOS that it's the end of the file, press Ctrl-Z (hold down the Control key and press Z), then press Enter:

```
<Ctrl-Z><Enter>
```

When you press Ctrl-Z, MS-DOS displays ^Z (the ^, remember, represents the Control key). After you press Enter, MS-DOS acknowledges that it copied a file:

```
1 file(s) copied

C:\DOS>_
```

To verify that the file is there, display the directory of the floppy disk in drive B:

```
C:\DOS>dir b:
```

Sure enough, NOTE.DOC is on the floppy disk:

```
Volume in drive B is EXAMPLES 2
 Volume Serial Number is 1839-10EE
 Directory of B:\

NOTE     DOC        94 01-05-95   2:03p
```

```
       1 file(s)         94 bytes
                 1,457,152 bytes free

C:\DOS>_
```

(This directory listing was generated with MS-DOS version 6.2, which sets off values greater than 999 with thousands separators, when the data comes from the Dir, Mem, Chkdsk, or Format command).

This method of creating a text file is quick and convenient and is used in examples throughout the book.

Displaying a Text File

Because you can read the characters in text files, you'll often want to display a text file on the screen. It's even easier to display one than it is to create it. Just use the MS-DOS Type command. To display your file, type the following:

```
C:\DOS>type b:note.doc
```

MS-DOS displays each line and returns to the command level:

```
January 5, 1995

Dear Fred,
Just a note to  remind you
that our meeting is at 9.

Jack

C:\DOS>_
```

This is the quickest way to see what's in a file; you'll probably use the Type command often. But displaying a file isn't always helpful because not all files are text files; they don't all contain readable characters. See for yourself. (If you're using MS-DOS from floppy disks, place the floppy disk with MORE.COM in drive A.) Type this:

```
C:\DOS>type more.com
```

Yes, the display is correct. It's hard to tell from that jumble what is in the file because the file contains a program stored in machine code, not text. Don't worry about the beeps you hear.

Printing a Text File

One of the main reasons you write documents, of course, is to have a printed copy. You can print your file by copying it to the printer. You've already copied from the keyboard to a disk. Now, copy from the disk to the printer. The printer is known to MS-DOS as PRN. Make

certain the printer is turned on, and type the following:

```
C:\DOS>copy b:note.doc prn
```

The file is printed. When you print a file, you'll probably want to position the paper by hand before you enter the command so that the printing will begin where you want it to on the page.

There's an easier way, however, to print a file: the Print command.

Note If you're using MS-DOS from floppy disks, you need the floppy disk containing the file PRINT.COM. Remember, if MS-DOS cannot find a file, it displays the message *Bad command or file name* and waits for you to try again. When necessary, remember to check your MS-DOS disks for appropriate command files. Examples from here on assume that MS-DOS can find the files it needs to carry out your commands.

To print your file with the Print command, type the following:

```
C:\DOS>print b:note.doc
```

If MS-DOS responds this way,

```
Name of list device [PRN]: _
```

press the Enter key. MS-DOS displays one or both of the following messages and prints the file you specified:

```
Resident part of PRINT installed

  B:\NOTE.DOC is currently being printed

C:\DOS>_
```

These messages are explained in more detail in Chapter 5, "Managing Your Files."

The Print command makes most printers advance to the next page after printing. Although this file is too short to show it, you can continue to use the system to do other work while the Print command is printing a file.

Copying a Text File

The Copy command is one of the more versatile MS-DOS commands. You have already used it to create and print a text file. The Copy command also duplicates files.

To copy the file named NOTE.DOC into another file named LETTER.DOC, type the following:

```
C:\DOS>copy b:note.doc b:letter.doc
```

When you press Enter, MS-DOS copies the file; then it acknowledges that it did so:

```
1 file(s) copied

C:\DOS>_
```

Display the directory of the floppy disk in drive B again to verify the copy:

```
C:\DOS>dir b:
```

Now you have two text files:

```
Volume in drive B is EXAMPLES 2
 Volume Serial Number is 1839-10EE
 Directory of B:\

NOTE     DOC         94 01-05-95  2:03p
LETTER   DOC         94 01-05-95  2:03p
       2 file(s)        188 bytes
                  1,456,640 bytes free

C:\DOS>_
```

If you wanted, you could make changes to one file and continue to have a copy of the original version on disk. You'll find the Copy command quite useful when you need several files that differ only slightly or when you have several small files that can be combined in different ways to create other files: often-used paragraphs, for example, that can be recombined in different letters, contracts, or other documents.

Deleting a Text File

Just as you get rid of paper files, you can get rid of disk files. To delete NOTE.DOC from the floppy disk in drive B, type this:

```
C:\DOS>del b:note.doc

C:\DOS>_
```

Now check the directory one more time as it appears at the top of the next page:

```
C:\DOS>dir b:

 Volume in drive B is EXAMPLES 2
 Volume Serial Number is 1839-10EE
 Directory of B:\

LETTER   DOC         94 01-05-95  2:03p
       1 file(s)         94 bytes
                  1,457,152 bytes free

C:\DOS>_
```

It's gone.

Some Advanced Features

Several commands and features give you much greater control over the way MS-DOS does its work. You can do all this:

- Sort lines of data—for example, sort alphabetically the list of directory entries produced by the Directory command.
- View a long display one screenfull at a time, without having to freeze the display by pressing Pause or Ctrl-Num Lock.
- Tell MS-DOS to send the results, or *output*, of a command to the printer instead of to the display, simply by adding a few characters to the command.
- Search lines of data for a series of characters.

  Now you'll get a glimpse of these features, described in detail in later chapters.

Sorting Lines of Data

You have probably arranged card files or lists in some sequence, such as alphabetic or numeric order. The Sort command sorts, or arranges, lines of data such as a list of names. To see how this works, sort the lines of the text file LETTER.DOC.

Type the following:

```
C:\DOS>sort < b:letter.doc
```

The less-than symbol (<) tells MS-DOS to send a copy of the file LETTER.DOC to the Sort command, which then displays the lines of the file after rearranging (sorting) them into alphabetic order:

```
Dear Fred,
Jack
January 5, 1995
Just a note to remind you
that our meeting is at 9.

C:\DOS>_
```

Although you probably don't want to sort the lines of your letters, you can put whatever you like in a text file—for example, a list of customers or employees. The Sort command is a powerful addition to your kit of computer tools.

Viewing a Long Display One Screenful at a Time

When you displayed a directory in Chapter 3, lines might have scrolled off the top of the screen because the display was too long to fit. You saw that you can freeze the display by

pressing Pause or Ctrl-Num Lock or by using the /P parameter of the Directory command. There's another way to stop scrolling: The More command displays one screenful, with --*More*-- at the bottom of the screen, then waits for you to press any key to continue to the next screenful. Display the current directory again, this time using the More command. Type the vertical bar by holding down a Shift key and pressing the key labeled with the | character:

```
C:\DOS>dir œ more
```

MS-DOS displays the first screenful, but the last lines aren't displayed yet (notice the --*More*-- in the last line):

```
Volume in drive C is HARD DISK
 Volume Serial Number is 1608-5A30
 Directory of C:\DOS
.           <DIR>     01-15-94     6:05p
..          <DIR>     01-15-94     6:05p
EGA      SYS      4885 02-01-93    12:00a
FORMAT   COM     33087 02-01-93    12:00a
NLSFUNC  EXE      7052 02-01-93    12:00a
COUNTRY  SYS     17069 02-01-93    12:00a
EGA      CPI     58873 02-01-93    12:00a
HIMEM    SYS     13984 02-01-93    12:00a
KEYB     COM     14986 02-01-93    12:00a
KEYBOARD SYS     34697 02-01-93    12:00a
ANSI     SYS      9029 02-01-93    12:00a
DEBUG    EXE     20634 02-01-93    12:00a
EXPAND   EXE     16885 02-01-93    12:00a
FDISK    EXE     57224 02-01-93    12:00a
SYS      COM     13440 02-01-93    12:00a
NFORMAT  COM     18560 02-01-93    12:00a
ATTRIB   EXE     15796 02-01-93    12:00a
CHOICE   COM      1733 02-01-93    12:00a
- More -
```

To see the rest of the directory, press any key. If MS-DOS once again displays --*More*-- at the bottom of the screen, press Ctrl-Break to end the command.

The More command displays long output one screenful at a time, giving you a chance to view it all at your convenience.

Note When you use More and certain other commands that require MS-DOS to manipulate files, you might see directory entries for files with odd names, such as ALCJDEAO, 1106002B, or %PIPE1.$$$. These are temporary files that MS-DOS creates and then erases when it no longer needs them.

Sending Command Output to the Printer

In earlier examples, you printed the output of the Directory command by using the Print Screen key. There's a more direct way to print the output of a command: Simply follow the command with a greater-than symbol (>) and the name of the printer, PRN. To print the current directory, make sure the printer is turned on and type this:

```
C:\DOS>dir > prn
```

If you don't want to wait for the whole directory to be printed, cancel the printing by pressing Ctrl-Break. This same technique can be used to send the output of a command to some other device or to a file, by substituting the device name or file name for PRN.

Note If you're using MS-DOS from floppy disks, try using this command to print a list of the command files on each floppy disk. Such lists are useful for reference.

Finding a Series of Characters in a File

How many times have you searched through a pile of letters or notes, looking for a particular item or reference? If you have to look through MS-DOS files or the output of MS-DOS commands, the Find command will do the looking for you. For example, suppose you want to see the directory entries of all MS-DOS files with SK in their names. Try the following (the quotation marks tell MS-DOS which letters—known technically as a *character string*, or just *string*—to look for).

(If you don't have a hard disk, check to see which of your MS-DOS floppy disks contains command files with the letters SK in their names.)

Type this (and be sure to use uppercase characters when you type "SK"):

```
C:\DOS>dir œ find "SK"
```

MS-DOS displays only the entries with SK in their names (your list might differ):

```
FDISK    EXE     57224 02-01-93  12:00a
CHKDSK   EXE     16216 02-01-93  12:00a
DISKCOMP COM     10636 02-01-93  12:00a
DISKCOPY COM     11879 02-01-93  12:00a
DOSKEY   COM      5883 02-01-93  12:00a

C:\DOS>_
```

The Find command is even more useful when you search for a series of characters in a text file. If a file contains a list of names and telephone numbers, for example, you can quickly display one particular entry or all entries that contain a particular series of characters (such as an area code). You can even display all entries that *don't* contain a particular series of characters. Chapter 13, "Taking Control of Your System," shows you how to create such an automated index of names and telephone numbers with nothing but MS-DOS commands.

Combining Features

These advanced features of MS-DOS can also be used together in a single command, giving you even more flexibility in controlling MS-DOS. Combining these features makes it possible to do a great deal with just one command. For example, suppose you want to print the directory entries of all files in the current directory whose names include the letter U; further, you want the entries sorted alphabetically. Type the following:

```
C:\DOS>dir œ find "U" œ sort > prn
```

The whole command translates easily into this: Look at the directory, find all files with the letter U in their names, sort those files alphabetically, and send the results to the printer. MS-DOS does as it is told, as you see here:

```
COUNTRY  SYS     17069 02-01-93  12:00a
DEBUG    EXE     20634 02-01-93  12:00a
NLSFUNC  EXE      7052 02-01-93  12:00a
SUBST    EXE     18478 02-01-93  12:00a
UNDELETE EXE     25916 02-01-93  12:00a
UNFORMAT COM     18560 02-01-93  12:00a
VIRULIST CPS     35520 02-01-93  12:00a
WNUNDEL  EXE    129792 02-01-93  12:00a
WNUNDEL  HLP     32329 02-01-93  12:00a
```

(Your list might differ, and you might see the name of a temporary file whose size is 0, but the files should still be sorted alphabetically.)

You may rarely search your directories this carefully, but such combinations make MS-DOS a powerful tool for handling text files.

PREVIOUS NEXT

Chapter Summary

This chapter concludes the portion of the book designed to give you a feel for running MS-DOS, including some of its advanced features. The key points to remember include these:

- The computer's memory is cleared each time you turn the system off. To save your work permanently, you must store it in a file on a disk.
- A text file contains ordinary characters you can read.
- A command file contains instructions that MS-DOS uses to carry out a command.
- A file name can be up to eight characters long; you can add an extension of up to three characters, separated from the file name by a period.
- Each file on a disk must have a different name or a different extension.

The remainder of the book shows you how to use MS-DOS to manage your files, disks, and devices; use the MS-DOS text editors; and create your own commands.

PREVIOUS NEXT

Part II: Learning to Use MS-DOS

Chapter List

Chapter 5: Managing Your Files

Chapter 6: Managing Your Floppy Disks

Chapter 7: Managing Your Devices

Chapter 8: A Tree of Files

Chapter 9: Managing Your Hard Disk

Chapter 10: Protecting Your Disks and Files

Chapter 11: The MS-DOS Shell

Chapter 12: Creating and Editing Files of Text

Chapter 13: Taking Control of Your System

Chapter 14: Creating Your Own Commands

Chapter 15: Creating Smart Commands

Chapter 16: Creating More Smart Commands

Chapter 17: Tailoring Your System

Part Overview

Part 2 shows you how to use MS-DOS to manage your work with the computer. The chapters in Part 2 include extensive examples that use real-life situations to illustrate each MS-DOS command, but the information is organized so that you can quickly find a particular topic to refresh your memory.

These chapters present many different features, and much of the material is relevant to all versions of MS-DOS. Chapters 5, 6, and 7 describe managing your files, floppy disks, and devices. Chapter 8 explains a tree-structured file system. Chapter 9 covers the use of a hard disk in detail. Chapter 10 shows you how to safeguard your disks and files against loss or damage. Chapter 11 introduces the MS-DOS Shell. Chapter 12 describes the MS-DOS text editor, and Chapters 13 through 17 show you how to use advanced features.

Chapter 5: Managing Your Files

The previous chapters defined a file as a named collection of related information stored on a disk and showed you several ways to create, copy, display, print, and otherwise work with your computer files. This chapter describes the MS-DOS filing system in detail, showing you more about how files are named and how you can use MS-DOS to manage your computer files.

Note A few of the examples in the remaining chapters of this book may look familiar because they repeat some of the examples in Chapters 2, 3, and 4. This repetition is intentional so that Chapters 5 through 17 present a complete guide to MS-DOS commands. You won't have to refer back to any of the earlier chapters for command descriptions.

The MS-DOS File Commands

To be useful, a filing system—whether it contains disk files or paper files—must be kept orderly and up to date. Using the MS-DOS file commands, you can manage your disk files much as you manage your paper files. This chapter covers the MS-DOS commands you use most often on a day-to-day basis. It shows you how to do the following:

- Display directory entries in different ways with the Directory command
- Display a file with the Type command
- Copy and combine files with the Copy command
- Send a copy of a file to a device with the Copy command
- Remove a file from a disk with the Delete command
- Change the name of a file with the Rename command
- Compare two files with the Compare command
- Print a file with the Print command

PREVIOUS NEXT

File Names and Extensions

As Chapter 4, "A Look at Files and Floppy Disks," pointed out, files are named so that you (and MS-DOS) can tell them apart; each file on a disk must have a different name. You know that a file name can be up to eight characters long, made up of any letters or numbers; you can also use the following symbols:

` ' ~ ! @ # $ % ^ & ( ); _ - { } \´

You can add a suffix—called an *extension*—to the file name to describe its contents more precisely. The extension can be up to three characters long and can include any of the characters that are valid for the file name. It must be separated from the name by a period. The extension distinguishes one file from another just as the name does: REPORT and REPORT.JAN, for example, are two different files, as are REPORT.JAN and REPORT.FEB. Figure 5-1 shows some valid and invalid file names.

These File Names Are Valid	These Are Invalid...	Because
B	1994BUDGET	Name too long
94BUDGET	BUDGET.1994	Extension too long
BUDGET.94	.94	No file name
BUDGET.95	SALES 94.DAT	Space not allowed
BDGT(95)	$1,300.45	Comma not allowed

Figure 5-1: Some valid and invalid file names.

Try to make file names and extensions as descriptive as possible. A short name might be easy to type, but you can have difficulty remembering what the file contains if you haven't used it for a while. The more descriptive the name, the more easily you can identify the contents of the file.

Special Names and File Name Extensions

Some names and file name extensions have special meanings to MS-DOS. As you'll see in Chapter 7, "Managing Your Devices," MS-DOS refers to the parts of your computer system by certain reserved names known as *device names*. The keyboard and screen, for example, are named CON, and the system clock is called CLOCK$. You cannot use any of these device names as file names.

Similarly, certain file name extensions have special meanings to MS-DOS. These extensions either are created by MS-DOS or cause MS-DOS to assume that the file contains a particular type of program or data. You should avoid giving files any of these extensions. A number of the most important ones are listed in Figure 5-2.

Name	Meaning to MS-DOS
BAS	Short for *Basic*. Contains a program written in the Basic programming language. You run the program while using Basic.
BAT	Short for *Batch.* Identifies a text file you create that contains a set of MS-DOS commands that are run when you type the name of the file.
COM	Short for *Command.* Identifies a command file that contains a program MS-DOS runs when you type the file name.
EXE	Short for *Executable*. Like COM, identifies a command file that contains a program MS-DOS runs when you type the file name.
HLP	Short for *Help*. Contains a file of help text displayed by certain programs, including the MS-DOS Shell and the versions 5 and 6 Editor and Basic programs.
OVL	Short for *Overlay*. Identifies a command file that contains part of a large program.
SYS	Short for *System*. Identifies a file that can be used only by MS-DOS.

Figure 5-2: Some special MS-DOS file name extensions.

Application programs also usually recognize special extensions. For example, Microsoft Word, the Microsoft word processor, uses DOC to identify a document, BAK to identify a backup version of a document, and STY to identify a file that contains a style sheet of print specifications. Again, avoid using extensions that have special meaning for your application programs; these extensions are usually listed in the documentation that comes with each program.

Specifying the Drives

You can tell MS-DOS to look for a file in a specific drive by typing the drive letter and a colon before the file name. If you specify a file as *a:report*, for example, MS-DOS searches in drive A for a file named REPORT; if instead you specify the file as *report*, MS-DOS looks for the file in the current drive.

PREVIOUS NEXT

Preparing for the Examples

If your system isn't running, start it. Place a blank, formatted floppy disk in drive A. Then type the following to change the current drive to A:

```
C:\>a:
```

MS-DOS acknowledges by changing the system prompt:

```
A:\>_
```

Now you're ready to create a set of sample files on this floppy disk. Type the following; where you see ^Z, press F6 or hold down the Ctrl key and press Z, then press Enter:

```
A:\>copy con report.doc
This is a dummy file.
^Z
```

MS-DOS responds:

```
1 file(s) copied
A:\>_
```

Now that you've created this file, you can use it and the MS-DOS Copy command to create some more. Type the following Copy commands (described in detail later in this chapter) to create some other sample files.

```
A:\>copy report.doc report.bak
       1 file(s) copied
A:\>copy report.doc bank.doc
       1 file(s) copied
A:\>copy report.doc budget.jan
       1 file(s) copied
A:\>copy report.doc budget.feb
       1 file(s) copied
A:\>copy report.doc budget.mar
       1 file(s) copied
```

Now check the directory.

```
A:\>dir
```

It should list six files:

```
Volume in drive A is EXAMPLES 1
 Volume Serial Number is 1A2C-13F5
 Directory of A:\
REPORT   DOC        23 01-05-95   9:16a
REPORT   BAK        23 01-05-95   9:16a
BANK     DOC        23 01-05-95   9:16a
```

```
BUDGET   JAN       23 01-05-95   9:16a
BUDGET   FEB       23 01-05-95   9:16a
BUDGET   MAR       23 01-05-95   9:16a
       6 file(s)      138 bytes
                   1454592 bytes free
```

Remember, the time and date in your list will be different, but the file names and sizes (23 bytes) should be the same.

PREVIOUS NEXT

Wildcard Characters

To make it easier to manage your disk files, most file commands let you use wildcard characters to handle several files at once. That way, when you want to do the same thing to several files—change their names, perhaps, or erase them—you don't have to enter a separate command for each file. You can use wildcard characters to tell MS-DOS you mean a set of files with similar names or extensions. Just as a wild card in a poker game can represent any other card in the deck, a wildcard character can represent any other character in a file name or extension.

There are two wildcard characters, the asterisk (*) and the question mark (?). The following examples use the Directory command to illustrate ways you can use wildcard characters to specify groups of files.

Using the Asterisk Wildcard Character: *

The asterisk makes it easy to carry out commands on sets of files with similar names or extensions; it can represent up to all eight characters in a file name or up to all three characters in an extension. If you use the asterisk to represent the entire name or extension, you are specifying all file names or all extensions.

The following examples illustrate several ways to use the asterisk character to find selected directory entries. You can use the asterisk in the same ways with other MS-DOS commands as well.

To specify all files named BUDGET, regardless of extension, type the following:

```
A:\>dir budget.*
```

MS-DOS displays the directory entry of each sample file named BUDGET, regardless of its extension:

```
Volume in drive A is EXAMPLES 1
 Volume Serial Number is 1A2C-13F5
 Directory of A:\

BUDGET   JAN       23 01-05-95   9:16a
BUDGET   FEB       23 01-05-95   9:16a
BUDGET   MAR       23 01-05-95   9:16a
       3 file(s)        69 bytes
                  1454592 bytes free
```

To specify all file names beginning with B, type the following:

```
A:\>dir b*
```

If you don't specify an extension, the Directory command displays the entry for each file that

matches the name, regardless of extension. (It's the equivalent of specifying the extension as *.) There are four such files:

```
Volume in drive A is EXAMPLES 1
 Volume Serial Number is 1A2C-13F5
 Directory of A:\
BANK     DOC        23 01-05-95   9:16a
BUDGET   JAN        23 01-05-95   9:16a
BUDGET   FEB        23 01-05-95   9:16a
BUDGET   MAR        23 01-05-95   9:16a
       4 file(s)       92 bytes
                 1454592 bytes free
```

To specify all files with the same extension, regardless of name, you replace the name with *. For example, to specify each file with the extension DOC, type the following:

```
A:\>dir *.doc
```

MS-DOS displays just those entries:

```
Volume in drive A is EXAMPLES 1
 Volume Serial Number is 1A2C-13F5
 Directory of A:\

REPORT   DOC        23 01-05-95   9:16a
BANK     DOC        23 01-05-95   9:16a
         2 file(s)       46 bytes
                    1454592 bytes free
```

Using the Question Mark Wildcard Character: ?

The question mark replaces only one character in a file name or extension. You'll probably use the asterisk more frequently, using the question mark only when one or two characters vary in the middle of a name or extension.

To see how the question mark works, type the following:

```
A:\>dir budget.?a?
```

This command specifies all files named BUDGET that have extensions beginning with any character, followed by the letter *a*, and ending with any character. MS-DOS displays two entries:

```
Volume in drive A is EXAMPLES 1
 Volume Serial Number is 1A2C-13F5
 Directory of A:\

BUDGET   JAN        23 01-05-95   9:16a
BUDGET   MAR        23 01-05-95   9:16a
```

```
2 file(s)         46 bytes
          1454592 bytes free
```

A Warning About Wildcard Characters

Be careful using wildcard characters with commands that can change files. Suppose you spent several days entering a year's worth of budget data into 12 files named BUDGET.JAN, BUDGET.FEB, BUDGET.MAR, and so on. On the same disk you also have three files that you don't need, named BUDGET.OLD, BUDGET.TST, and BUDGET.BAD. The disk is getting full, so you decide to delete the three unneeded files. It's 2 A.M., you're tired, and you're in a hurry, so you quickly type *del budget.** and press Enter. You have told MS-DOS to do more than you wanted.

You may realize immediately what you have done, or it may not dawn on you until you try to use one of the 12 good budget files and MS-DOS replies *File not found*. You display the directory; there isn't a single file named BUDGET, because you told MS-DOS to delete them all.

With commands that can change or delete a file, use wildcard characters with extreme caution.

This warning applies even if you have version 5 or later of MS-DOS, which includes a command named Undelete that can help you recover deleted files. Bear this in mind: *Undelete cannot always recover files completely*. As you'll see later in this book, Undelete can be valuable. But don't let it lull you into a false sense of security.

PREVIOUS NEXT

Help When You Need It

Note If you don't have version 5 or later of MS-DOS, your version of MS-DOS doesn't include the help feature described here. Skip to the section called "Displaying Directory Entries."

As you work with MS-DOS, you'll find that some commands become so familiar you type them without thinking. Others, which you need less often, will linger in your mind, but you may have to refresh your memory to use them correctly. If you have version 5 or later of MS-DOS, you have a feature called *online help* that you can call on whenever you want help with a command. Online help isn't intended to teach you how to use MS-DOS, but it is a handy guide that can remind you of the form of a command you haven't used in a while.

To see a list of the commands for which you can request help, type *help* and press Enter:

```
A:\>help
```

Version 6 of MS-DOS responds by displaying three columns of topics, each topic enclosed in angle brackets, as you can see at the top of the next page.

You can scroll through this list one row at a time with the Up and Down arrow keys, or a screen at a time with the PgUp and PgDn keys. Press Tab to move from one column to the next, and press a letter to move the cursor to the first command that begins with that letter. To display information about any of the commands listed, move the cursor to the command name and press Enter.

For example, you used the Date command in a previous chapter. *Date* is the third entry below *Copy* in the first column; press the Down arrow key until the cursor is under *Date* and press Enter. Now MS-DOS displays the help information about the Date command:

In addition to this description of the form of the command (MS-DOS calls this the command's *syntax*), online help also includes notes about how you can use the command. Notice that the cursor is on the item *Notes;* press Enter. Now you see the notes about the Date command, and the cursor is on the item *Syntax*. Press Enter again, and once again MS-DOS displays the syntax of the command. Some commands also include examples that you can select just as you selected *Notes* and *Syntax*.

Press the Esc key until you return to the list of commands (the *Contents*). To leave online help, press Alt-F-X. That is, press the Alt key, then press F to display the File menu, and then press X to choose Exit. MS-DOS displays the system prompt.

You can display the help for a specific command directly by typing *help* followed by the name of the command. This has the same effect as typing *help* and then choosing the command from the list. Try it by typing the following:

```
A:\>help date
```

Now MS-DOS displays the syntax screen for the Date command. When you're through with the command information, you can either display the contents list by pressing Alt-C or return to MS-DOS by pressing Alt-F-X. Press Alt-F-X now to return to MS-DOS.

You can display a different sort of help information for a command by typing the name of the command followed by a space and /?. Type the following to see this kind of help information for the Date command:

```
A:\>date /?
```

MS-DOS simply displays information about the command you specified, followed by the system prompt; there are no items to choose, nor is there a list of commands:

```
Displays or sets the date.

DATE [mm-dd-yy]

mm-dd-yy    Sets the date you specify.

Type DATE without parameters to display the current date setting
```

and a prompt for a new one. Press ENTER to keep the same date.

If you're using version 5 and you type *help,* MS-DOS displays a list of the commands for which you can request help, along with a brief description of what each command does. The list is several screens long; press any key to move to the next screen. If you're using version 5 and you type *help* followed by a command's name, or a command's name followed by /?, MS-DOS displays the same information as does version 6 when you type the name of the command followed by /?, as shown above.

Now on to the MS-DOS file-management commands.

PREVIOUS NEXT

Displaying Directory Entries

As you have seen, the Directory (dir) command displays entries from the directory that MS-DOS keeps on each disk. Each entry includes the name and extension of the file, its size in bytes, and the date and time it was created or last updated. You can use the Directory command to display all entries or just the entries of selected files.

In the descriptions of the commands here and throughout the remaining chapters, you are shown the general form of the command—the name of the command and most or all of its parameters—before you try the examples. If a parameter has an exact form, such as /W, the form is shown. If a parameter is something you specify, such as a file name, you'll see it named and shown enclosed in angle brackets—for example, <filename>.

The Directory Command and Its Parameters

Depending on your version of MS-DOS, the Directory command has either three parameters or more. All versions allow you to specify a file name and two parameters typed as /W and /P. Versions 5 and later have more parameters, among them one that allows you to display only the file name and extension of the directory entries and another that arranges directory entries by name, extension, size, or date and time.

Written out, the format of the Directory command and these parameters looks like this:

dir <filename> /W /P /O<sortorder> /B /S

(Other parameters of the Directory command are described later, and a complete summary appears in Appendix C, "MS-DOS Command Reference.")

If you include <filename>, MS-DOS searches the current directory for the file you specify. You can also do the following:

- Precede <filename> with a drive letter. For example, *dir a:report.doc* tells MS-DOS to look on drive A rather than on the current drive.
- Use wildcard characters to specify a group of files. For example, *dir budget.** tells MS-DOS to display the entries for all files named *budget* regardless of extension.
- Omit <filename> to tell MS-DOS to display the entries for all files in the current directory. For example, *dir* displays all entries for the current directory of the current drive; *dir b:* displays all entries for the current directory of drive B.

The /W (Wide) parameter tells MS-DOS to display only the file names and extensions in five columns across the screen. This display contains less information than does a complete directory listing because it omits file sizes, dates, and times, but it is useful when a directory listing is quite long and you only want to see what files are in the directory.

The /P (Pause) parameter tells MS-DOS to display the entries one screenful at a time; a message at the bottom of the screen tells you to press any key to continue. The /P

parameter provides complete file information, including size, date, and time, so it is useful when you want a detailed look at a long directory listing.

The /O<sortorder> parameter, in versions 5 and later, tells MS-DOS to sort (arrange) a directory listing by name, extension, size, or date and time:

- Typing *dir /on* sorts the files alphabetically by file name.
- Typing *dir /oe* sorts the files alphabetically by extension.
- Typing *dir /os* sorts the files by size, smallest to largest.
- Typing *dir /od* sorts the files by date (earliest to latest) and, within the same date, sorts by the time (morning to evening) the file was created or last changed. You cannot sort by time only; that is, you cannot type *dir /ot* to sort by time but not by date.

The /B (Bare) parameter, in versions 5 and later, tells MS-DOS to display only the names and extensions of the files you specify.

The /S (Subdirectory) parameter, in versions 5 and later, tells MS-DOS to display the files in the directory you specify and in all subdirectories of the directory.

Examples of Displaying Directory Entries

Because you have already used the Directory command several times, only the options are shown here. First, use the /W parameter to see a wide directory display showing the files on your sample floppy disk. Type the following:

```
A:\>dir /w
```

MS-DOS arranges just the name and extension of each file in five columns across the screen. If you have version 5 or later, the display looks something like this (only three of the five columns are shown):

```
Volume in drive A is EXAMPLES 1
 Volume Serial Number is 1A2C-13F5
 Directory of A:\

REPORT.DOC      REPORT.BAK      BANK.DOC  ...
BUDGET.MAR
       6 file(s)        138 bytes
                   1454592 bytes free
```

If you have version 4 or earlier, your display is similar but doesn't include quite as much detail. Such a display doesn't contain as much information as the standard directory display, but it packs a lot of entries onto the screen. The wide format is particularly handy when all you want is a quick look at the names of the files on a crowded disk.

Pausing the Directory Display

To display the directory of a disk one screenful at a time, you use the /P option. MS-DOS displays the first screenful of entries, followed by either *Press any key to continue...* or *Strike a key when ready...*, depending on your version of MS-DOS. To see the next screenful, press any key. This option lets you view the entire directory without using Pause or Ctrl-Num Lock to freeze the display periodically.

Your sample disk contains less than one screenful of file names, but you can see the effect by using the /P parameter to list the MS-DOS files on your hard disk. Type the following:

```
A:\>dir c:\dos /p
```

MS-DOS responds by displaying the first screenful of its files and waiting for you to press a key:

```
Volume in drive C is HARD DISK
 Volume Serial Number is 1608-5A30
 Directory of C:\DOS
.           <DIR>     01-15-94   6:05p
..          <DIR>     01-15-94   6:05p
EGA      SYS      4885 02-01-93  12:00a
FORMAT   COM     33087 02-01-93  12:00a
NLSFUNC  EXE      7052 02-01-93  12:00a
COUNTRY  SYS     17069 02-01-93  12:00a
EGA      CPI     58873 02-01-93  12:00a
HIMEM    SYS     13984 02-01-93  12:00a
KEYB     COM     14986 02-01-93  12:00a
KEYBOARD SYS     34697 02-01-93  12:00a
ANSI     SYS      9029 02-01-93  12:00a
DEBUG    EXE     20634 02-01-93  12:00a
EXPAND   EXE     16885 02-01-93  12:00a
FDISK    EXE     57224 02-01-93  12:00a
SYS      COM     13440 02-01-93  12:00a
UNFORMAT COM     18560 02-01-93  12:00a
ATTRIB   EXE     15796 02-01-93  12:00a
CHOICE   COM      1733 02-01-93   12:00a
CHKDSK   EXE     12267 02-01-93  12:00a
Press any key to continue . . .
```

Press any key to see the next screenful. Continue pressing the key until MS-DOS displays the system prompt, or press Ctrl-Break to cancel the command.

Sorting the Directory Display

As described earlier, starting with version 5, MS-DOS has added new ways for you to display a directory. Suppose you have a floppy disk full of files with different extensions, and

you want to group them to get a better idea of what the floppy disk contains.

The /O parameter of the Directory command does the job. If you have version 5 or later, try it. Type the following:

```
A:\>dir /oe
```

MS-DOS responds:

```
Volume in drive A is EXAMPLES 1
 Volume Serial Number is 1A2C-13F5
 Directory of A:\

REPORT   BAK        23 01-05-95   9:16a
REPORT   DOC        23 01-05-95   9:16a
BANK     DOC        23 01-05-95   9:16a
BUDGET   FEB        23 01-05-95   9:16a
BUDGET   JAN        23 01-05-95   9:16a
BUDGET   MAR        23 01-05-95   9:16a
       6 file(s)        138 bytes
                   1454592 bytes free
```

The files are arranged alphabetically by extension. Remember, you can also group files by file name, by size, or by date and time. The last is particularly useful when you have more than one version of a file and you want to look at the versions in chronological order.

Listing Only Files

If you simply want to see what files are on a disk, the /B parameter gives you a bare-bones look at its contents. With the sample files, for instance, typing

```
A:\>dir /b
```

produces this display:

```
REPORT.DOC
REPORT.BAK
BANK.DOC
BUDGET.JAN
BUDGET.FEB
BUDGET.MAR
```

Finally—and this applies to any MS-DOS command—remember that in MS-DOS versions 5 and later you can type *help* or use the universal /? parameter to refresh your memory about a command and its parameters.

Displaying a File

Many of the files you use are text files, and there will be times when you want to check the contents of a file but don't need a printed copy. MS-DOS gives you a quick way to see what's in a file: the Type command. (The name Type is a carryover from the days when most computers had only consoles that looked like typewriters.)

When you use the Type command, MS-DOS displays the file without stopping; if the file is longer than one screenful and you want to read the entire file, freeze the display by pressing Pause or Ctrl-Num Lock, or by using the More command.

The Type command has one parameter:

type <filename>

<filename> is the name of the file to be displayed. The Type command displays just one file at a time, so you can't use wildcard characters in the file name. If you do use a wildcard character, MS-DOS displays *Invalid filename or file not found* and returns to the command level. If the file you name doesn't exist, MS-DOS displays *File not found* and, again, returns to the command level.

An Example of Displaying a File

To display the file named REPORT.DOC on the floppy disk in the current drive, verify that your sample disk is in drive A and type this:

```
A:\>type report.doc
```

MS-DOS displays the file:

```
This is a dummy file.
```

You'll probably use the Type command frequently to check your text files.

PREVIOUS NEXT

Making Copies of Files

Just as you sometimes make copies of your paper files, you'll find yourself needing copies of your disk files.

You may want to share a file with a colleague who has a computer, you may want to alter the copy slightly to produce a different version, or you may want to copy a file from your hard disk to a floppy disk you can carry with you. The Copy command can make a copy of a file on the same disk (with a different file name) or on a different disk (with any valid file name).

When used to make copies of files, the Copy command has two major parameters, <file1> and <file2>. The format of the Copy command is this:

copy <file1> <file2>

<file1> is the name of the file to be copied (the *source* file), and <file2> is the name of the copy to be made (the *target* file). You can use wildcard characters to copy a set of files.

Note Four additional, seldom-used parameters (/A, /B, /V, and /Y) are described in the Copy command entries in Appendix C, "MS-DOS Command Reference."

When you copy files, pay attention to these conditions:

- If you specify a <file1> (including the drive letter) that is not on the disk in the current drive and omit <file2>, MS-DOS copies <file1> to the disk in the current drive and gives the copy the same name as the original. Example (if the current drive is C): *copy a:report.mar*.
- If you specify only a drive letter as <file2>, the file is copied to the disk in the drive you specify and MS-DOS gives it the same name as <file1>. Example: *copy report.feb a:.*
- If you specify a <file1> that doesn't exist, MS-DOS responds *File not found,* followed by the name you typed for <file1> and *0 file(s) copied,* and it returns to command level.
- If <file2> doesn't exist, MS-DOS creates it as a copy of <file1>.
- If you specify a <file2> that does exist and you're using version 6.0 or earlier, MS-DOS replaces its contents with <file1>. This is the same as deleting the existing target file, so be careful not to give the target the same name as an existing file you want to keep. Versions 6.2 and later will warn you with the message *Overwrite <file2> (Yes/No/All)?* if <file2> already exists in the target location, giving you the opportunity to cancel the copy.

The following practice session illustrates different ways to copy files; this section also describes the type of situation in which you might want to use each form of the command.

Examples of Copying Files

You want to change a document you already have on disk, but you want to keep the original as well as the changed version. For example, to make a copy of the file REPORT.DOC on the same floppy disk and to name the copy RESULTS, type this:

```
A:\>copy report.doc results
```

MS-DOS acknowledges *1 file(s) copied*.

To verify that both files, REPORT.DOC and RESULTS, are on the floppy disk, display the directory by typing this:

```
A:\>dir
```

MS-DOS now shows seven files on the floppy disk:

```
Volume in drive A is EXAMPLES 1
 Volume Serial Number is 1A2C-13F5
 Directory of A:\

REPORT   DOC       23 01-05-95   9:16a
REPORT   BAK       23 01-05-95   9:16a
BANK     DOC       23 01-05-95   9:16a
BUDGET   JAN       23 01-05-95   9:16a
BUDGET   FEB       23 01-05-95   9:16a
BUDGET   MAR       23 01-05-95   9:16a
RESULTS            23 01-05-95   9:16a
       7 file(s)      161 bytes
                 1454080 bytes free
```

Any time you want to verify the results of an example, use the Directory command to see what files are on the floppy disk.

Suppose you want to copy a file from another disk and store it, under the same file name, on the disk in the current drive. For example, your current drive is drive A. To copy a file from a different drive to the floppy disk in drive A, all you need to specify is the drive letter and name of the source file because MS-DOS assumes you want to copy the source file to the disk in the current drive and give the target file the same name unless you specify otherwise.

Try it. One of the files in the DOS subdirectory on your hard disk is named README.TXT. To copy it from drive C (the hard disk) to the floppy disk in drive A (the current drive), type the following:

```
A:\>copy c:\dos\readme.txt
```

MS-DOS responds *1 file(s) copied*.

Next, suppose you want to change a file and store the new version under the same file name and on the same disk as the original, but you want to be able to distinguish between the two versions. Simply make a copy of the file on the same disk, with the same file name but a different extension. You can use the asterisk wildcard character to tell MS-DOS to use the same file name. For example, to make a copy of BUDGET.MAR and call it BUDGET.APR, type the following:

```
A:\>copy budget.mar *.apr
```

MS-DOS acknowledges that it copied one file.

You have several files named REPORT stored on disk. Suppose you want to keep the originals but make copies of them all for a new project; to avoid confusion, you want to give the copies a new file name but keep the same extension. For example, to make a copy of each file named REPORT, giving each copy the name FORECAST, type the following:

```
A:\>copy report.* forecast.*
```

MS-DOS displays the name of each source file as it makes the copies:

```
REPORT.DOC

REPORT.BAK
        2 file(s) copied
```

You can copy all the files on a floppy disk by specifying the source file as *.* and specifying the target as just a drive letter. This procedure is not the same as copying the floppy disk with the Diskcopy command; the difference is explained in the section called "Comparing Two Floppy Disks" that begins on page 110 in Chapter 6, "Managing Your Floppy Disks."

In addition to the Copy command, MS-DOS versions 3.2 and later include two additional commands for copying files selectively: Replace, which lets you copy only files that already exist on the target disk, and Xcopy, which lets you copy entire subdirectories in addition to files in a directory. Both of these commands are described in Chapter 9, "Managing Your Hard Disk."

PREVIOUS NEXT

Sending Files to Devices

In Chapter 4, "A Look at Files and Floppy Disks," you printed a file by using the Copy command to send a copy of the file to the printer. You can also send a copy of a file to any other output device. If, for example, you copy a file to a communications connection, or *port*, on the computer, the file goes to whatever device is attached to the port—such as a telecommunications line to another computer.

When it is used to send a copy of a file to an output device, the Copy command has two parameters:

copy <filename> <device>

<filename> is the name of the file to be sent; <device> is the name of the device to which the file is to be sent.

Be sure <device> exists. If you try to send a file to a device that doesn't exist or isn't ready, MS-DOS might stop running. You won't hurt anything, but you'll have to restart the system.

An Example of Sending Files to a Device

To send a copy of each sample file with the extension DOC to the printer, make certain your printer is turned on and type this:

```
A:\>copy *.doc prn
```

MS-DOS displays the name of each file as it sends the file to the printer:

```
REPORT.DOC
BANK.DOC
FORECAST.DOC
        1 file(s) copied
```

The files are printed with no separation between them. MS-DOS reports only one file copied because, in effect, only one output file was created: the printed copy of the three files.

Combining Files

Sometimes, it's useful to combine several files. Perhaps you have several short documents, and you decide it would be easier and more convenient to work with a single document that includes all the shorter ones. If you have several sets of files with similar names or extensions, you can combine each set into a new file. The Copy command allows you to copy several files into a new file without destroying the original versions.

When it is used to combine files, the Copy command has two parameters:

copy <source> <target>

<source> represents the files to be combined. You can use wildcard characters to name the source files to be combined, or you can list several file names, separating each from the next with a plus sign (+). If any file in the list doesn't exist, MS-DOS goes on to the next name without telling you the file doesn't exist.

<target> represents the file that results from combining the source files. If you specify a target, MS-DOS combines the source files into the target file. If you don't specify a target but do specify individual source files, MS-DOS combines all the source files into the first file in the <source> list, changing its contents. Before the copy is actually performed, MS-DOS versions 6.2 and later will warn you about overwriting the first file.

Examples of Combining Files

Suppose you have two files you want to use as the basis for a single new file, but you want to keep the originals intact. For example, to combine the files BANK.DOC and REPORT.DOC into a new file named BANKRPT.DOC, type this:

```
A:\>copy bank.doc+report.doc bankrpt.doc
```

MS-DOS displays the names of the source files as it copies them:

```
BANK.DOC
REPORT.DOC
        1 file(s) copied
```

Again, MS-DOS reports one file copied because the command created only one file.

You can also copy several files into an already existing file. To combine BUDGET.JAN, BUDGET.FEB, and BUDGET.MAR into the first file, BUDGET.JAN, type this:

```
A:\>copy budget.jan+budget.feb+budget.mar
```

MS-DOS displays the name of each source file as it copies:

```
BUDGET.JAN
BUDGET.FEB
BUDGET.MAR
        1 file(s) copied
```

Now, suppose you've been keeping monthly budget files. It's the end of the year. You still need separate monthly files for comparison with next year's figures, but right now you want to work with all the files together. To combine all the files named BUDGET into a file named ANNUAL.BGT, type this:

```
A:\>copy budget.* annuaI.bgt
```

MS-DOS responds:

```
BUDGET.JAN
BUDGET.FEB
BUDGET.MAR
BUDGET.APR
        1 file(s) copied
```

Or suppose you want to combine pairs of files with the same file names but different extensions. You can combine them under the same file names, with new extensions, and end up with both the original and combined versions. For example, if you have entered all the examples, among the files on the floppy disk in drive A are REPORT.DOC and REPORT.BAK, FORECAST.DOC and FORECAST.BAK. To combine each pair of files with the same name and the extensions DOC and BAK into a single file with the same name and the extension MIX, type this:

```
A:\>copy *.bak+*.doc *.mix
```

MS-DOS displays the names of the files as it copies them:

```
REPORT.BAK
REPORT.DOC
FORECAST.BAK
FORECAST.DOC
        2 file(s) copied
```

This time MS-DOS reports two files copied because the command created two files: REPORT.MIX and FORECAST.MIX.

PREVIOUS NEXT

Moving Files

The Move command, new to version 6.0, moves a file from one place to another. Move works much like the Copy command except that it doesn't leave a copy of the file in the original location. Because the Move command is most useful in managing directories of files, it is described in Chapter 8, "A Tree of Files," under the heading "Moving Files from One Directory to Another."

PREVIOUS NEXT

Deleting Files

Just as you have to clean out a file drawer once in a while, you'll occasionally have to clear your disks of files you no longer need. The Delete command (you can type it either as *del* or as *erase*) deletes one or more files from a disk.

The Delete command has two parameters:

del <filename> /P

<filename> is the name of the file to be deleted. If you use wildcard characters, MS-DOS deletes all files that match <filename>. If the file doesn't exist, MS-DOS displays *File not found* and returns to command level.

/P, in versions 4 and later, tells MS-DOS to prompt you for verification before deleting the file. If you use wildcard characters to delete more than one file, MS-DOS prompts you to verify the deletion of each file.

Safeguards

When you delete files, MS-DOS assumes you know what you're doing—that you know exactly which files you're eliminating and that you really don't want them anymore. But being human, you sometimes make mistakes, so you need ways to protect your files.

One good safeguard is the Copy command. MS-DOS doesn't limit the number of times you can copy a file, so you can think of the Copy command as the computer-based equivalent of your photocopy machine. Copy a file, preferably onto a different disk, whenever its value outweighs the minimal time and effort required to copy it.

When you delete files, use the /P parameter, especially if you use wildcard characters to specify a set of files. If you don't use this parameter to tell MS-DOS to prompt for confirmation, the only time it hesitates before carrying out a Delete command is when you type *del* *.*. Because this form of the Delete command removes all files in a directory, MS-DOS prompts you *All files in directory will be deleted! Are you sure (Y/N)?*

If your version of MS-DOS doesn't include the /P parameter, double-check the command on the screen before pressing Enter. Verify the drive letter (if necessary), the file name, and the extension you typed. If you used wildcard characters, either be certain you know exactly which files will be deleted or consider using this safety check: Press Esc to cancel the Delete command, and then use the same wildcard characters with a Directory command. The Directory command lists all the files that match the wildcard characters. If the file names you see match the ones you expected to see, you can reenter the original Delete command with confidence.

Another safeguard, particularly helpful for those using versions of MS-DOS that do not support the /P parameter, involves setting the Read-Only attribute for files you want to keep. (See page 178, "Changing the Attributes of a File or a Directory".) Be aware that if

wildcards are used to identify a group of files for deletion, any candidate files that have the Read-Only attribute set will not be deleted, but no message is provided to this effect.

Examples of Deleting Files

Cautionary statements aside, Delete itself is one of the easiest MS-DOS commands to use. For example, to delete the file named BUDGET.APR on the floppy disk in the current drive, type this:

```
A:\>del budget.apr
```

Press the Enter key, and the file's gone.

Similarly, type this to delete the MS-DOS file you copied from the hard disk:

```
A:\>del readme.txt
```

The next example deletes all files on the sample floppy disk whose extension is BAK. If you have version 4 or later, use the /P parameter to ask MS-DOS to prompt you for verification. Type the following command (omit the /P if you're using an earlier version):

```
A:\>del * . bak /p
```

With the /P parameter, MS-DOS responds by displaying the name of the first file whose extension is BAK and asking you whether to delete it:

```
A:\REPORT.BAK,     Delete (Y/N)?_
```

Type *y* to delete it. Now MS-DOS shows you the name of the next file and again asks:

```
A:\FORECAST.BAK,     Delete (Y/N)?_
```

Type *y* again to delete the file. If you had typed *n* in response to either prompt, MS-DOS would have left the file alone and moved on to the next file (if any) that matched the file name you typed as part of the Delete command. The /P option is thus handy when you want to delete several—but not all—files that have similar names or extensions. The prompt lets you avoid the work of entering a separate command for each file.

But what if your version of MS-DOS doesn't have the /P option? If you typed the Delete command as *del *.bak*, your disk drive became active for a brief time after you pressed the Enter key, and then the system prompt returned to the screen. In that short time, MS-DOS deleted the two sample files whose extension was BAK without telling you the names of the files. This example underscores the need for you to be sure you have typed the correct file name and the correct drive letter or extension (if necessary) whenever you use wildcard characters with the Delete command.

Whether or not you use the /P parameter, be particularly careful when you use the question mark in a file extension. To see why, create two additional test files by typing the following:

```
A:\>copy results *.1
```

```
        1 file(s) copied

A:\>copy results *.12
        1 file(s) copied
```

This gives you three files with the same name but different extensions: RESULTS, RESULTS.1, and RESULTS.12. Now type a Directory command, using a single question mark as the extension. Before you press Enter, decide which files you think MS-DOS will list:

```
A:\>dir results.?
```

MS-DOS lists two files:

```
Volume in drive A is EXAMPLES 1
 Volume Serial Number is 1A2C-13F5
 Directory of A:\

RESULTS 1          23 01-05-95   9:46a
RESULTS            23 01-05-95   9:46a
        2 file(s)         46 bytes
                    1451008 bytes free
```

If you expected to see only RESULTS.1, the response to this Directory command is a surprise. If you had typed a Delete command instead of a Directory command, the files you see listed are the files MS-DOS would have erased. An unexpected directory listing can be momentarily confusing, but unintentionally deleting files is definitely an unpleasant surprise.

To delete both of the files you just created but leave your original RESULTS file untouched, type the following command:

```
A:\>del results.1?
```

Typing *1* as the first character of the extension ensures that the file RESULTS isn't affected. Note, however, that even though you included a question mark (which takes the place of a single character) after the 1, MS-DOS deleted RESULTS.1 as well as RESULTS.12. Be careful whenever you use wildcard characters with commands that change or delete files.

Changing File Names

There will be times when you want to change the name of a file. You might simply change your mind, or perhaps you'll have changed the contents of a file so much that you want to give it a name that more closely describes its new contents. For example, if your word processing program automatically makes a backup copy whenever you edit and save a file, the Rename command can be valuable when (as happens) your working copy is inadvertently lost or damaged—perhaps the power fails, you have a problem with the program itself, or you've mangled the file with editing changes and want to start all over from scratch.

The Rename command changes a file's name or extension, or both. You can use wildcard characters to rename a set of files.

You can abbreviate the Rename command as *ren*. It has two parameters:

rename <oldname> <newname>

<oldname> is the name of an existing file. If the file doesn't exist, MS-DOS displays *Duplicate file name or file not found* and returns to command level.

<newname> is the name you want to give to the file specified by <oldname>. If there is already a file with the new name, MS-DOS displays *Duplicate file name or file not found* and returns to command level. Two files on the same disk can't have the same name, and MS-DOS would have to erase the existing file to carry out the command, so this built-in safeguard keeps you from inadvertently erasing one file in the process of renaming another.

The Rename command simply changes the name of a file; it doesn't copy a file to a different disk. Both the old name and the new name must refer to a file on the same drive. If you specify a different drive letter with the new file name, MS-DOS responds *Invalid parameter*.

Examples of Changing File Names

To change the name of the file ANNUAL.BGT to FINAL on the disk in the current drive, type this:

```
A:\>ren annual.bgt final
```

MS-DOS changes the name and displays the system prompt.

To change the extension of the file BUDGET.MAR from MAR to 003 on the disk in the current drive, you can use the * wildcard character for the new file name. Type the following:

```
A:\>ren budget.mar *.003
```

The file is now named BUDGET.003.

To change the extension DOC to TXT for all files on the disk in the current drive, use the * for both the old and new file names. Type the following:

```
A:\>ren *.doc *.txt
```

Verify this change with the Directory command by typing this:

```
A:\>dir *.txt
```

MS-DOS shows four files, all of which used to have the extension DOC:

```
Volume in drive A is EXAMPLES 1
 Volume Serial Number is 1A2C-13F5
 Directory of A:\

REPORT   TXT        23 01-05-95   9:16a
BANK     TXT        23 01-05-95   9:16a
FORECAST TXT        23 01-05-95   9:16a
BANKRPT  TXT        47 01-05-95  10:51a
       4 file(s)       116 bytes
                   1452032 bytes free
```

If you use the Directory command now to display the entries of all files with the extension DOC, MS-DOS responds *File not found*.

PREVIOUS NEXT

Comparing Files

Sometimes you'll want to know whether two files are exactly the same. Suppose you have two files named BUDGET on different disks. They're the same length—but are they different budgets or two copies of the same one? You could display or print both files and compare them, but that could take quite a while, and you still might miss some small difference. It's quicker and more accurate to use the File Compare (fc) command.

File Compare has two main parameters. See Appendix C, "MS-DOS Command Reference," for a listing of all the parameters.

fc <file1> <file2>

<file1> and <file2> are the file names of the files to be compared.

File Compare displays different results depending on the types of files you are comparing. If both files contain only text, File Compare compares the lines of the first file with the lines of the second file. If it finds differences, it displays the last line in each file that matched, the lines that don't match, and the first lines in each file that once again match. File Compare then continues to compare the file until it comes across another unmatching portion, whereupon it displays another set of lines. If File Compare checks 100 lines without finding a match, it displays the message *Resync failed. Files are too different*. (Some versions of MS-DOS display the message ****Files are different****.)

If one or both of the files you want to compare are nontext files (such as program files) that have the extension EXE, COM, SYS, OBJ, LIB, or BIN, File Compare performs a byte-by-byte comparison. File Compare compares the two files from beginning to end, always checking to see whether the bytes in corresponding positions in the files match. It does not display mismatched lines but lists the characters that differ and how far each is from the beginning of the file. If one file is shorter than the other, File Compare displays the message *FC: FILEx longer than FILEy* at the end of the comparison.

Note MS-DOS versions 3.3 through 5 also include the Compare command. See the Compare entry in Appendix C for more information.

Examples of Comparing Files

The following examples show the MS-DOS prompts and responses as they appear in version 6. If you have an earlier version of MS-DOS, your messages may differ, but the command works as described. Type the commands as shown, but bear in mind that you may not see exactly what is shown here.

To compare REPORT.TXT with BUDGET.FEB, type this:

```
A:\>fc report.txt budget.feb
```

The files are identical, so MS-DOS replies as follows:

```
Comparing files REPORT.TXT and BUDGET.FEB
FC: no differences encountered
```

Because all the sample files contain the same words, you'll have to create two different files to see how File Compare notifies you of differences. To create a file named COMPARE1.TXT, type the following:

```
A:\>copy con compare1.txt
This is line 1.
This is line 2.
This is line 3.
This is line 4.
^Z
    1 file(s) copied
```

Now create a file named COMPARE2.TXT that skips line 3 and adds some lines at the end. Type the following:

```
A:\>copy con compare2.txt
This is line 1.
This is line 2.
This is line 4.
This is line 5.
This is line 6.
^Z
    1 file(s) copied
```

Now compare COMPARE1.TXT with COMPARE2.TXT. Type this:

```
A:\>fc compare1.txt compare2.txt
```

The File Compare command compares the files and then reports the differences:

```
Comparing files COMPARE1.TXT and COMPARE2.TXT
***** COMPARE1.TXT
This is line 2.
This is line 3.
This is line 4.
***** COMPARE2.TXT
This is line 2.
This is line 4.
*****

***** COMPARE1.TXT
***** COMPARE2.TXT
This is line 5.
This is line 6.
*****
```

As you can see, File Compare found the two mismatching portions of the files. The first section of the display points out the absence of line 3 in COMPARE2.TXT. The second section lists the two added lines. Because you won't need these two files again, delete them now by typing this:

```
A:\>del compare?.txt
```

To see the difference between comparing text files and comparing nontext files, compare two nontext files found in the DOS subdirectory on your hard disk. To compare the files EDIT.COM and HELP.COM, type the following:

```
A:\>fc c:\dos\edit.com c:\dos\help.com
```

MS-DOS quickly displays the differences it finds:

```
Comparing files C:\DOS\EDIT.COM and C:\DOS\HELP.COM
00000196: 45 51
00000197: 44 48
00000198: 43 45
00000199: 4F 4C
0000019A: 4D 50
```

Each line gives the location in the files of the mismatching characters and the values representing the characters. The numbers are combinations of the digits 0 through 9 and the letters A through F. These characters are from the base-16 number system, usually called hexadecimal, in which A through F are used to represent the decimal numbers 10 through 15. If you must know what the differing characters are, or if you must calculate the exact locations, you need a chart of the American Standard Code for Information Interchange (ASCII), which shows how characters are encoded, and you need a guide to hexadecimal arithmetic. The manuals that came with your computer might contain both.

You can use wildcard characters to compare two sets of files with one command. To compare all files with the extension TXT against all files with the same file name but the extension MIX, type the following:

```
A:\>fc *.txt *.mix
```

There are now four files with the extension TXT, but only two (REPORT.MIX and FORECAST.MIX) with the extension MIX. MS-DOS tells you which files it can find and which ones it is unable to compare:

```
Comparing files REPORT.TXT and REPORT.MIX
***** REPORT.TXT
***** REPORT.MIX
This is a dummy file.
*****

Comparing files BANK.TXT and BANK.MIX
FC: cannot open BANK.MIX - No such file or directory
```

```
Comparing files FORECAST.TXT and FORECAST.MIX
***** FORECAST.TXT
***** FORECAST.MIX
This is a dummy file.
*****

Comparing files BANKRPT.TXT and BANKRPT.MIX
FC: cannot open BANKRPT.MIX - No such file or directory
```


PREVIOUS NEXT

Printing Files

You can print files at the same time you're using the computer to do other things. The Print command keeps a list—called the *print queue*—of files to be printed, and it prints the files in the order in which they appear in the queue. The print queue normally can hold up to 10 files.

In addition to printing files, the Print command lets you change some characteristics of its operation—notably, the size of the print queue and the printer that MS-DOS uses. For a description of these uses of the Print command, see "Changing Operation of the Print Command" later in this chapter.

Because the computer can really do only one thing at a time, MS-DOS prints when nothing else is happening, such as when you pause to think between keystrokes. You'll notice that printing slows and sometimes even stops when something else is going on—especially when MS-DOS is using a disk drive.

You use the Print command to add a file to the print queue, delete a file from the queue, cancel all printing, and display the names of the files in the queue. When used to print a file, the Print command has four main parameters:

print <filename> /P /C /T

<filename> is the name of the file to be added to or deleted from the print queue. You can enter more than one file name with a Print command; just type the list of file names, separating each from the next with a blank.

/P (Print) tells MS-DOS to add <filename> to the print queue. MS-DOS assumes this parameter if all you specify is <filename>.

/C (Cancel) tells MS-DOS to remove <filename> from the print queue. If the file is being printed, printing stops.

/T (Terminate) stops all printing. If a file is being printed, printing stops and all files are removed from the print queue.

If you enter the Print command with no parameters, MS-DOS displays the list of files in the print queue.

Examples of Printing a File

The following examples work best with a dot-matrix printer equipped with a tractor feed that uses fanfold paper. If you don't have such a printer, or if you use a network printer, you might prefer to read through these examples rather than try them for yourself. On cut-sheet printers and network printers some examples might not work as described, and several can cause a fast printer to use many sheets of paper.

MS-DOS advances the paper to the next page each time it prints a new file, so check what

each example does before you try it. You can then be ready to stop the printing process, and you'll save both time and paper. First, a bit of preparation.

Most of the sample files you created in this chapter consist of a single line and would print too quickly for you to try using all the Print parameters in the following examples. Copy the README.TXT file from the DOS directory on your hard drive to your floppy disk:

```
A:\>copy c:\dos\readme.txt
```

This file is significantly larger than the others on your floppy disk and will certainly take plenty of time to print.

To print the file README.TXT, type this:

```
A:\>print readme.txt
```

The first time you enter the Print command after starting the system, MS-DOS might prompt you for the name of the printer to use:

```
Name of list device [PRN]: _
```

The brackets around PRN mean that MS-DOS will use the device named PRN if you press the Enter key. Unless you have more than one printer attached to your system, or you are using a printer with a serial interface, just press the Enter key. If you have never printed with your printer, press the Enter key.

When you respond to the prompt, MS-DOS loads the Print command file into memory and keeps it there until you either turn the system off or restart MS-DOS. MS-DOS reports that the program is loaded:

```
Resident part of PRINT installed
```

When MS-DOS begins to carry out your Print command, it displays the names and print status of the files in the print queue:

```
A:\README.TXT is currently being printed
```

There is one file in the print queue (README.TXT), and it's now being printed.

When you printed a file in Chapter 4 by copying it to the printer, MS-DOS didn't display the system prompt—and you couldn't use the system—until the file had been printed. This time the system prompt returned as soon as printing started. As soon as MS-DOS starts printing a file with the Print command, MS-DOS is ready to accept another command from you.

If you decide you don't want to print a file after all, you can remove it from the print queue with the /C parameter; type the command you see at the top of the next page while README.TXT is being printed:

```
A:\>print readme.txt /c
```

MS-DOS displays a terse acknowledgment of your command:

```
PRINT queue is empty
```

and stops sending lines to your printer, even though your printer will probably continue to operate until it finishes printing the characters it has already received. At that point, the printer stops and advances the paper to a new page. If you check the last line of the printout, you might see a message like this:

```
NOTES ON MS-DOS VERSION 6
=========================

This file provides important information not included in the
Microsoft MS-DOS User's Guide or in MS-
File A:\README.TXT canceled by operator
```

You can put more than one file into the print queue with a single Print command. To tell MS-DOS to print both README.TXT and BUDGET.JAN, type this:

```
A:\>print readme.txt budget.jan
```

MS-DOS starts printing README.TXT and displays the print queue:

```
A:\README.TXT is currently being printed
      A:\BUDGET.JAN is in queue
```

If you want to stop printing, you can remove all files from the print queue with the /T parameter. Type the following, again while README.TXT is being printed:

```
A:\>print /t
```

Again MS-DOS displays the acknowledgment *PRINT queue is empty*. This time it also prints the message *All files canceled by operator* at the point where it stopped printing and, again, advances the paper and removes all remaining files from the queue.

You can also put several files in the print queue at once by using wildcard characters. To print all the files whose extension is TXT, type this:

```
A:\>print *.txt
```

Now MS-DOS tells you that there are five files in the queue (the order might vary), as you see here:

```
A:\REPORT.TXT is currently being printed
      A:\BANK.TXT is in queue
      A:\README.TXT is in queue
      A:\FORECAST.TXT is in queue
      A:\BANKRPT.TXT is in queue
```

MS-DOS prints the files in the order shown. Stop all printing again by typing this:

```
A:\>print /t
```

Again MS-DOS stops printing the current file, prints the cancellation message, advances the paper, removes all remaining files from the queue, and acknowledges on the screen *PRINT queue is empty*.

The next example uses several sheets of paper. If you're not using continuous-form paper or an automatic sheet feeder, you shouldn't try this example because it could easily print on the platen of your printer; skip to the section called "Changing Operation of the Print Command."

The print queue normally holds up to 10 files. To fill it, tell MS-DOS to print all the files on the floppy disk in drive A; there are 12 text files, of which MS-DOS puts the first 10 in the queue. Type the following:

```
A:\>print *.*
```

MS-DOS tells you the queue is full and displays the list of files in the queue (again, the order might vary):

```
PRINT queue is full

    A:\REPORT.TXT is currently being printed
    A:\BANK.TXT is in queue
    A:\BUDGET.JAN is in queue
    A:\BUDGET.FEB is in queue
    A:\BUDGET.003 is in queue
    A:\RESULTS is in queue
    A:\README.TXT is in queue
    A:\FORECAST.TXT is in queue
    A:\BANKRPT.TXT is in queue
    A:\FINAL is in queue
```

You really don't need to print all these files. Stop all printing by typing this:

```
A:\>print /t
```

MS-DOS empties the queue and alerts you as before:

```
PRINT queue is empty
```

Changing Operation of the Print Command

MS-DOS initially limits the print queue to 10 files, but you can increase the size of the queue and can also tell MS-DOS to use a printer other than the standard printer, PRN. (More details on printers are in Chapter 7, "Managing Your Devices.")

You can change Print command operations only the first time you use the Print command during a session at your computer; if you try to use these options again before restarting MS-DOS or turning the computer off, MS-DOS displays the message *Invalid switch* and ignores the command.

When used to change the size of the print queue or the name of the printer, the Print command has two parameters:

print /D:<printer> /Q:<size>

/D:<printer> tells MS-DOS to use the printer named <printer>. If you omit /D:<printer>, MS-DOS uses the standard printer, named PRN.

/Q:<size> tells MS-DOS the number of files the print queue can hold; the minimum number is 4, and the maximum number is 32. If you omit /Q:<size>, the print queue holds 10 files.

If you wanted to increase the size of the print queue to 15 files, you would type *print /q:15* the first time you used the Print command. If you wanted to tell MS-DOS to use the printer named LPT2, you would type *print /d:lpt2*. You can combine these parameters in the same Print command, but you cannot combine them with other parameters unless you are entering the Print command for the first time since starting MS-DOS.

The Print command lets you print text files without losing the use of your system during printing; it can make both you and your system more productive.

Be aware that when you're using the Print command to print one or more files stored on a floppy disk, MS-DOS will continue to read the floppy disk even though the system prompt has been displayed. Do not remove the floppy disk until the printing task has been completed. If MS-DOS attempts to read the floppy disk, but the disk has been removed from the drive, no message is displayed on the screen (another program could now be using the screen) but the printed output will contain this message:

Not ready error reading file
a:\<filename.ext>

Several infrequently used parameters that give you more precise control over how the print program interacts with MS-DOS are described in Appendix C, "MS-DOS Command Reference."

PREVIOUS NEXT

Chapter 6: Managing Your Floppy Disks

Overview

Disks are the computer's filing cabinets. Managing your computer filing system includes not only keeping track of your files, as described in the previous chapter, but also taking care of your floppy disks. There are many ways to prepare and store information on floppy disks; the concepts underlying floppy disk handling, however, apply to all microcomputers.

Several MS-DOS commands deal with entire floppy disks, not with individual files. For example, you must prepare a new floppy disk for use; this process is called *formatting* (or, less commonly, *initializing*). Or, should you need an exact copy of a floppy disk, you don't copy each file separately; you copy the entire floppy disk with one command.

This chapter suggests ways to handle your floppy disks, briefly describes how MS-DOS stores files on floppy disks, and shows you how to do the following:

- Prepare a floppy disk for use with the Format command
- Duplicate a floppy disk with the Diskcopy command
- Compare the contents of two floppy disks with the Diskcomp command
- Analyze and report on the use of floppy disk storage space with the Check Disk command
- Assign, change, or delete the volume label (identifying name) of a disk with the Label command
- Display the volume label of a disk with the Volume command

The Diskcopy and Diskcomp commands are for floppy disks only. A number of other MS-DOS commands, such as Format, Check Disk, Label, and Volume, are used with both floppy disks and hard disks. Additional commands, such as Backup and Restore, are used exclusively or primarily with hard disks. Depending on which chapter is most appropriate, these disk-management commands are described in this chapter, in Chapter 9, "Managing Your Hard Disk," or in Chapter 10, "Protecting Your Disks and Files."

PREVIOUS NEXT

Handling Floppy Disks

Floppy disks are remarkably durable, especially the 3.5-inch type in a hard plastic shell. The useful life of a floppy disk depends on how often you use it, of course, but even more important is how you treat it. Handle your floppy disks with the same care you use when handling valuable recording tapes or photographs:

- Avoid touching the floppy disk surfaces that show through the openings in the protective jacket. Dirt, fingerprints, or dust can shorten the life of a floppy disk and can damage or destroy the data.
- Keep floppy disks away from magnets and other sources of magnetic influence, such as telephones, electric motors, and televisions.
- Keep food and drinks away from floppy disks. The same goes for cigarettes, cigars, pipes, and ashtrays.
- Don't fold, spindle, or mutilate floppy disks. Don't pile any other objects on them.
- Don't write on 5.25-inch floppy disk labels with a pencil, ballpoint pen, or other sharp instrument; use a felt-tipped marker.
- Store your floppy disks in a safe place when you're not using them. Protect them from extreme heat and cold, humidity, and contact with other objects.

Many products are available for storing floppy disks, including plastic boxes, vinyl pockets that fit a three-ring binder, and hanging file folders. All offer good protection; they aren't necessary, but they can make it easier for you to organize your floppy disks and store them safely, rather than leaving them scattered around your desk.

Although an office is a mild environment compared to a factory or a shop floor, data on a floppy disk can be damaged by such innocuous objects as a paper clip that has been stored in a magnetic paper-clip holder, a magnetized letter opener, an electric pencil sharpener, or a telephone-answering machine. If you put a letter down on top of a floppy disk lying on your desk, it's all too easy to put a hot coffee cup or a heavy object on the letter without realizing that the floppy disk is underneath.

The safest places for a floppy disk are in the computer and in protective storage. Information and time are two of your most valuable assets. A damaged floppy disk can cost you both, so protect your floppy disks accordingly.

Backing Up Your Floppy Disks

Even though you treat your floppy disks with care, they can still be mislaid or damaged by accident. Files can be inadvertently changed or erased, and eventually floppy disks simply wear out. Making backup copies of your floppy disks limits the amount of information and time you lose if something goes wrong. The time it takes to make these copies could be one of your better investments.

Unless a floppy disk containing a program is copy protected so that you cannot duplicate the original, make a copy of the program before you ever use it—even if you plan to install the program on your hard disk. Store the original floppy disk in a safe place, and use the copy. If something happens to the copy, make another copy from the original. Always keep the original stored safely.

Your collection of data files will grow as you use application programs such as a word processor or a spreadsheet. Back up a floppy disk containing data whenever the value of the information it contains—or the time it would take to re-create it—is greater than the value of a blank floppy disk and the few minutes it takes to make a copy. Keep your backup copies in a safe place, and use your computer with the comforting thought that, should something unforeseen happen, you're protected.

PREVIOUS NEXT

How Information Is Stored on a Floppy Disk

Information is stored on a floppy disk much as music or video is recorded on tape. A description of how MS-DOS uses a floppy disk helps you understand the commands you use to manage your floppy disks.

What Is a Floppy Disk?

What we call a floppy disk actually consists of two parts: a disk of thin plastic coated with magnetic material and a protective plastic jacket or hard shell. Figure 6-1 shows a 5.25-inch floppy disk in its flexible jacket, and Figure 6-2 on the next page shows both the front and back of a 3.5-inch floppy disk in its hard shell.

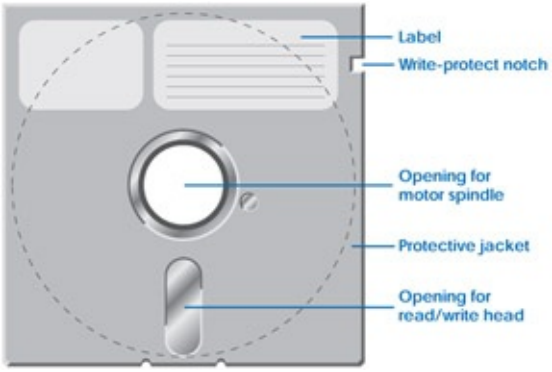


Figure 6-1: A 5.25-inch floppy disk.

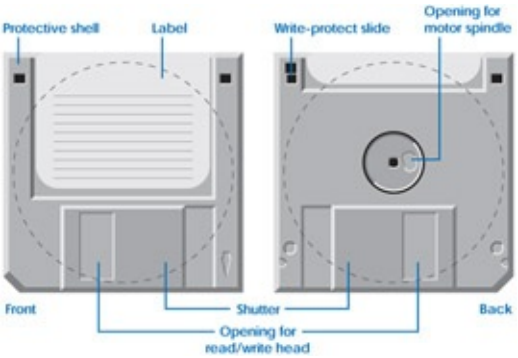


Figure 6-2: A 3.5-inch floppy disk.

The dashed lines in Figures 6-1 and 6-2 show how the coated disk lies inside the protective jacket. The magnetic coating itself is visible through the openings in the jacket of a 5.25-inch floppy disk (the dark areas in Figure 6-1). The spring-loaded shutter that normally covers the opening in a 3.5-inch floppy disk is moved aside by the floppy disk drive to provide access to the magnetic coating. The hole in the center of the disk goes around the drive motor, which spins the disk so that data can be written (recorded) or read (played back).

The write-protect notch or slide lets you protect all the files on a floppy disk from being erased or changed. To protect a 5.25-inch floppy disk, cover the write-protect notch with one of the small tabs of tape included with the box of floppy disks. To protect a 3.5-inch floppy disk, move the write-protect slide up (toward the edge of the floppy disk) until the

hole through the floppy disk shell is open. To permit files on the floppy disk to be changed or erased, remove the tape or move the slide down until the hole is closed. Protect your MS-DOS floppy disks in this way; protect each of your application program floppy disks, too, unless the application program manual tells you otherwise.

How Does MS-DOS Keep Track of Files?

Information is recorded on a floppy disk in narrow concentric circles called *tracks*; there are 40 such tracks on a low-density 360-kilobyte (360 KB) floppy disk and 80 tracks on the high-density 5.25-inch and all 3.5-inch floppy disks. A track is divided into smaller areas called *sectors,* each of which can hold 512 bytes. Figure 6-3 shows how tracks and sectors are laid out on a floppy disk. For simplicity, the illustration shows only four tracks.

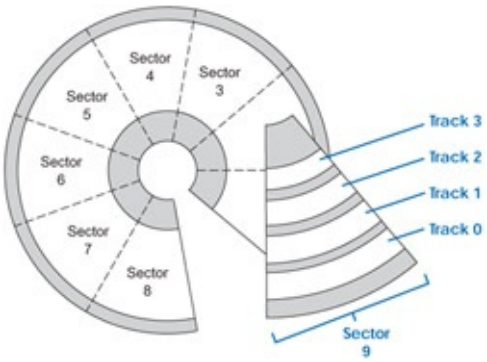


Figure 6-3: Tracks and sectors on a floppy disk.

The side, track, and sector numbers of the beginning of a file are stored as part of its directory entry. You don't see this information when you use the Directory command, but MS-DOS can find any sector on a floppy disk by its side, track, and sector numbers, just as you can find any seat in a theater by its section, row, and seat numbers.

Floppy Disk Capacity

Tracks on a low-density 5.25-inch floppy disk are numbered 0 through 39 (making 40 in all); sectors are numbered 1 through 9, for a total of 360 sectors (40 tracks times 9 sectors per track) on each side. Each sector holds 512 bytes, so the two sides of a low-density floppy disk have an actual capacity of 368,640 bytes (though not all of it is available for files).

A high-density 5.25-inch floppy disk has 80 tracks (numbered 0 through 79), each of which has 15 sectors, for a total of 1200 sectors per side. A sector still stores 512 bytes, so the two sides of a high-density floppy disk can hold 1,228,800 bytes (1.2 MB).

A 3.5-inch floppy disk has 80 tracks per side. Depending on its capacity, however, it has 9, 18, or 36 sectors per track—still with 512 bytes per sector. The 9-sector floppy disks can hold 737,280 bytes (720 KB); the 18-sector floppy disks can hold 1,474,560 bytes (1.44 MB). The 36-sector floppy disks can hold 2,949,120 bytes, which equals 2.88 MB.

Volume Label

Any disk can be assigned a name, or *volume label,* to identify its contents. The volume label can be up to 11 characters long, and you can use the same characters allowed in a file name, plus a space. You can assign a volume label when you format a disk or at any time later on. MS-DOS stores the volume label on the disk and displays it when you use the Directory, Check Disk, Label, or Volume command. The volume label is for identification only; you can't use it in a command to specify a disk.

PREVIOUS NEXT

Preparing for the Examples

The examples in this chapter require one floppy disk. If you have version 5 or later of MS-DOS, the Format command is more sophisticated than in earlier versions, so use a new floppy disk that's never been formatted. If you have an earlier version of MS-DOS, use any floppy disk that doesn't contain any files you want to keep (the examples erase any data on the floppy disk). Put the floppy disk in drive A.

PREVIOUS NEXT

Preparing a Floppy Disk for Use

As you've seen, the Format command prepares a floppy disk for use. The floppy disk can be either new or previously formatted. However, formatting means you can no longer use existing files, so don't format a floppy disk that contains files you need.

In carrying out the Format command, MS-DOS normally checks for flaws on the recording surface of the floppy disk and marks any bad sectors so they won't be used. After formatting, MS-DOS displays a message that tells you the maximum number of bytes the floppy disk can hold, how many bytes (if any) are defective, and how many bytes are available for storing files. Beginning with version 4, MS-DOS also tells you how much storage space the floppy disk has in terms of *allocation units* (a term for packages of bytes MS-DOS uses instead of reading from or writing to disk one byte at a time), and it displays the floppy disk's *volume serial number* (a disk identifier MS-DOS assigns during formatting). As mentioned earlier, this information is unlikely to be of any concern to you.

MS-DOS knows the type of floppy disk drive you have and formats the floppy disk accordingly. That is, it can tell whether the drive uses 3.5-inch or 5.25-inch, low-density or high-density floppy disks, and it formats the floppy disk to match the drive's maximum capabilities unless you specify otherwise.

This does not mean, however, that you can simply put any floppy disk in any drive and leave all the details to MS-DOS. It is up to you to provide the appropriate floppy disk for formatting. You cannot expect a half-gallon bucket to hold a gallon of water, nor can you expect MS-DOS to format a 360-KB or 720-KB floppy disk to reliably hold 1.2 MB or 1.44 MB of information.

You can, however, format and use low-density floppy disks in a high-density drive. For example, if you have a high-density 3.5-inch drive and run out of 1.44-MB floppy disks, you can format 720-KB floppy disks in the drive and use them reliably in a 720-KB or a 1.44-MB drive. If you have a 5.25-inch drive, you can format, read, and write 360-KB floppy disks in it. In this case, though, you should expect to use your 360-KB floppy disks only in another high-density drive because a standard 360-KB drive cannot read these floppy disks reliably.

The Format command reserves space on the floppy disk for the directory, thus reducing the amount of storage available for files. Because the size of the directory varies, depending on whether the floppy disk is low or high density, the storage capacity of your floppy disks depends on the type of floppy disk you use. Figure 6-4 shows floppy disk sizes, tracks per side, number of sectors per track, and total disk capacity for standard two-sided floppy disk drives.

Size	Tracks per Side	Sectors per Track	Total Capacity in Bytes	First Used in MS-DOS Version
3.5-inch	80	36	2,949,120 (2.88 MB)	5.0

3.5-inch	80	18	1,474,560 (1.44 MB)	3.3
3.5-inch	80	9	737,280 (720 KB)	3.2
5.25-inch	80	15	1,228,800 (1.2 MB)	3.0
5.25-inch	40	9	368,640 (360 KB)	2.0

Figure 6-4: Storage capacity of different floppy disks.

Format Command Parameters

When used to prepare a floppy disk, the Format command has the following six main parameters:

format <drive> /4 /F:<size> /V:<label> /Q /U

<drive> is the letter, followed by a colon, of the drive that contains the floppy disk to be formatted (such as *b:*). With version 3.1 or earlier, if you omit <drive>, MS-DOS formats the floppy disk in the current drive. If you omit <drive> with later versions, MS-DOS responds either *Drive letter must be specified* or *Required parameter missing* and returns to the system prompt.

/4 formats a 360-KB floppy disk in a high-density 5.25-inch drive.

/F:<size>, in versions 4 and later, formats a floppy disk for the capacity indicated by <size>. You can specify any appropriate floppy disk capacity from 160 KB to 2.88 MB, and you can type <size> in any of the forms shown in Figure 6-5. If you are using version 3.2 or 3.3, two alternative parameters, /N:<sectors> and /T:<tracks>, let you specify floppy disk capacity by giving MS-DOS the number of sectors and tracks to format; examples are given in Appendix C, "MS-DOS Command Reference."

Disk Capacity	Type in Any of the Following Forms (No Space Preceding K, KB, M, MB)					
360 KB	360	360K	360KB			
720 KB	720	720K	720KB			
1.2 MB	1200	1200K	1200KB	1.2	1.2M	1.2MB
1.44 MB	1440	1440K	1440KB	1.44	1.44M	1.44MB
2.88 MB[a]	2880	2880K	2880KB	2.88	2.88M	2.88MB

[a] Versions 5 and later.

Figure 6-5: Ways to type <size> as part of the /F:<size> parameter of the Format command in versions 4 and later.

/V:<label>, in versions 4 and later, assigns the floppy disk the volume label <label>. This parameter speeds formatting by letting you specify the volume label ahead of time and thus skip the usual request for a volume label that MS-DOS otherwise displays at the end of the format procedure. If you have an earlier version of MS-DOS, /V (without <label>) works in reverse, telling MS-DOS to request a volume label at the end of the format procedure.

/Q, in versions 5 and later, tells MS-DOS to perform a quick format. You can perform a quick format only on a previously formatted disk. MS-DOS clears the records on the disk that tell it where files are stored, but it doesn't prepare the disk for use in any other way, nor does it check the disk for bad sectors. A quick format is very fast, and even though it isn't as thorough as a normal format, it does make all of the usable disk space available for storing new files.

/U, also in versions 5 and later, instructs MS-DOS to perform an unconditional format—one that clears the disk completely and cannot be reversed with the Unformat command.

Warning

If you're not using version 3.2 or later and you type a Format command without specifying a drive letter, MS-DOS formats the disk in the current drive. If the current drive is a floppy disk drive that contains your system disk and you haven't covered the write-protect notch, MS-DOS erases every file on your system floppy disk.

If the current drive is a hard disk, formatting erases every file on it—not just the MS-DOS files but every program and data file you have stored. Be certain you know which disk is going to be formatted before you press the Enter key after typing in a Format command.

Examples of Preparing a Floppy Disk

Format the floppy disk in drive A, and give it a volume label. If you have version 4 or later, type the following:

```
C:\>format a: /v:dosdisk
```

If you have an earlier version, type this:

```
C:\>format a: /v
```

MS-DOS asks you to put the floppy disk in drive A:

```
Insert new diskette for drive A:
and press ENTER when ready..._
```

Make sure the right floppy disk is in the drive, and then press Enter.

If you're using version 5 or later, MS-DOS first displays a message telling you it's checking the existing format (if any) of the disk. Then it begins the formatting and tells the size of the disk it's formatting. Versions 4 and later then report on their progress with a message that *xx percent* of the disk has been formatted.

If you're using version 3.1 or an earlier version, MS-DOS displays *Formatting...* while it formats the disk. If you're using version 3.2 or 3.3, MS-DOS displays *Head: 0 Cylinder: 0* and changes the head and cylinder numbers to show you its progress (the term *cylinder* is another way of referring to a track).

When MS-DOS has formatted the floppy disk, it displays *Format complete.* If you have version 4 or later, this message is followed by a report on available storage space and, on a separate line, the volume serial number assigned by MS-DOS (which, if you're curious, is based on the current date and time). As you'll see in a moment, MS-DOS has also used the parameter */v:dosdisk* to name the floppy disk DOSDISK. At the end of the report, MS-DOS asks if you want to format another floppy disk:

```
Format another (Y/N)?_
```

Type *n* and press Enter.

If you're using an earlier version of MS-DOS, the *Format complete* message is followed by a prompt for the volume label:

```
Volume label (11 characters, ENTER for none)? _
```

Name the floppy disk DOSDISK by typing the following:

```
dosdisk
```

MS-DOS displays the report of available storage on the floppy disk and asks if you want to format another. Reply *n.*

Now display the directory of the disk: It's empty, but you can see the volume label on the first line.

```
C:\>dir a:

 Volume in drive A is DOSDISK
 Volume Serial Number is 2F49-1AFF
 Directory of A:\

File not found
```

(If you don't have version 4 or later, you don't see the volume serial number.)

Copying a Complete Floppy Disk

The Diskcopy command makes an exact duplicate of any floppy disk. If the target floppy disk isn't formatted, later versions of MS-DOS format it before copying. Diskcopy in earlier versions requires formatted floppy disks. Diskcopy works only with floppy disks of the same size and capacity. For example, you cannot use it to copy from a 360-KB floppy disk to a formatted 1.2-MB, 5.25-inch floppy disk, nor can you use it to copy from a hard disk to any floppy disk.

The Diskcopy command has three main parameters:

diskcopy <source> <target> /V

<source> is the letter, followed by a colon, of the drive that contains the floppy disk to be copied (such as *a:*).

<target> is the letter, followed by a colon, of the drive that contains the floppy disk that is to receive the copy (such as *b:*). If you have only one floppy disk drive, <source> and <target> can be the same letter (for example, *diskcopy a: a:*).

/V (*verify*) ensures that the floppy disk is copied correctly.

If you omit <target>, MS-DOS copies from the floppy disk in <source> to the floppy disk in the current drive; if you omit <target> and you specify the current drive as <source>, MS-DOS assumes you want to use only the current drive and prompts you to switch floppy disks during the copy. If you don't specify either <source> or <target>, MS-DOS assumes you want to use only the current drive and prompts you to switch floppy disks during the copy.

Examples of Copying a Floppy Disk

This example copies an existing floppy disk, duplicating information from the source floppy disk to the target floppy disk. Use one of your MS-DOS floppy disks as the source. If your MS-DOS floppy disks are not close at hand, use a floppy disk for one of your application programs or use a data floppy disk—the one you used for the examples in Chapter 5, "Managing Your Files," will do. Put the source floppy disk in drive A.

Before proceeding, be sure that your target floppy disk is either unformatted or formatted for the same capacity as the source floppy disk. Remember, Diskcopy works only if the target floppy disk is or can be formatted for the same capacity as the source.

If You Have One Floppy Disk Drive

Because you have only one floppy disk drive, MS-DOS must use it for both the source and the target floppy disks, prompting you to exchange floppy disks as required. Check that the source floppy disk is in the floppy disk drive.

To copy the floppy disk, type this:

```
C:\>diskcopy a: a:
```

MS-DOS prompts you to put in the source floppy disk:

```
Insert SOURCE diskette in drive A:
Press any key to continue . . .
```

You put the floppy disk in already, so press any key. MS-DOS tells you how many tracks, sectors, and sides it's copying, then prompts you to put the target floppy disk in the drive:

```
Insert TARGET diskette in drive A:
Press any key to continue . . .
```

Remove the source floppy disk, put in the target floppy disk, and press a key. If the floppy disk is unformatted and your version of MS-DOS formats as it copies, a message tells you what is happening. If you're using version 6.0 or earlier, MS-DOS will also prompt you to exchange floppy disks until it has finished copying the original. When MS-DOS has finished, it asks if you want to copy another:

```
Copy another diskette (Y/N)? _
```

Reply *n.*

MS-DOS versions 6.2 and later can read the entire source floppy disk (using the hard disk for temporary storage), then write the entire target floppy disk, eliminating the numerous floppy disk changes required in previous versions.

If You Have Two Floppy Disk Drives

If your system has two disk drives of different types (a 3.5-inch and a 5.25-inch), you can't use both drives because the Diskcopy command only works with identical floppy disks. You can, however, copy a disk by using just one drive; follow the instructions under the preceding heading, "If You Have One Floppy Disk Drive."

If your system has two identical floppy disk drives, your source floppy disk should be in drive A. To copy the source floppy disk in drive A to the target floppy disk in drive B, type this:

```
A:\>diskcopy a: b:
```

MS-DOS prompts you to put in the floppy disks. Put the target floppy disk in drive B and press a key. MS-DOS tells you how many tracks, sectors, and sides it's copying, then asks if you want to copy another; reply *n.*

PREVIOUS NEXT

Comparing Two Floppy Disks

Occasionally you'll want to know whether two floppy disks are identical. The Diskcomp command compares two floppy disks track by track. Diskcomp can be used only with floppy disks of the same size and capacity; you cannot use it to compare a hard disk with a floppy disk.

Note Just because two floppy disks contain the same files, they're not necessarily identical—the files might be stored in different sectors. If you want to compare all the files on two floppy disks, rather than the floppy disks themselves, use the File Compare (fc) command described in Chapter 5, "Managing Your Files," and specify all files (*.*).

The Diskcomp command has two main parameters:

diskcomp <drive1> <drive2>

<drive1> and <drive2> are the drive letters, each followed by a colon, of the drives containing the floppy disks to be compared (such as *a:* and *b:*).

If MS-DOS finds any differences, it displays the side and track of each—as you see here:

```
Compare error on
side 0, track 33
```

Examples of Comparing Two Floppy Disks

Follow the instructions under the heading that describes your system.

If You Have One Floppy Disk Drive

To compare the source floppy disk to the copy you just made, type this:

```
C:\>diskcomp a: a:
```

MS-DOS prompts you to put in the first floppy disk:

```
Insert FIRST diskette in drive A:
Press any key to continue . . .
```

Make sure the copy you just made is in the drive, then press a key. You will see something like this:

```
Comparing 80 tracks

18 sectors per track, 2 side(s)
```

MS-DOS prompts you to put in the second floppy disk:

```
Insert SECOND diskette in drive A:
```

```
Press any key to continue . . .
```

Remove the floppy disk, put in the source floppy disk you used for the Diskcopy example, and then press a key. If necessary, MS-DOS continues to prompt you to exchange floppy disks until it finally tells you it's done and asks if you want to compare more floppy disks:

```
Compare OK

Compare another diskette (Y/N) ?_
```

Reply *n.*

If You Have Two Floppy Disk Drives

If your system has two disk drives of different types (a 3.5-inch and a 5.25-inch), you can't use both drives because the Diskcomp command works only with identical floppy disks. You can, however, compare disks by using just one drive; follow the instructions under the preceding heading, "If You Have One Floppy Disk Drive."

If your system has two identical floppy disk drives, you can compare the floppy disk in drive B (the copy you just made) to the floppy disk in drive A (the floppy disk you used as your source) by typing this:

```
C:\>diskcomp a: b:
```

MS-DOS prompts you to put in the floppy disks:

```
Insert FIRST diskette in drive A:
Insert SECOND diskette in drive B:
Press any key to continue . . .
```

The floppy disks are already in the drives, so press a key. MS-DOS reports how many tracks, sectors, and sides it is comparing, then reports the results of the comparison and asks if you want to compare more floppy disks:

```
Comparing 80 tracks
18 sectors per track, 2 side(s)
Compare OK
Compare another diskette (Y/N) ?_
```

Reply *n.*

Checking the Condition of a Disk

Computers aren't infallible; malfunctions can produce errors in the directory of a disk. Such errors are rare, but the Check Disk (chkdsk) command helps by making sure that all files are recorded properly.

Note If you're using version 6.2 or later, the ScanDisk program is a better tool for detecting and correcting disk errors. ScanDisk is described in Chapter 9, "Managing Your Hard Disk," in the section called "Checking a Disk for Errors."

Check Disk analyzes the directory on a disk, comparing the directory entries with the locations and lengths of the files, and reports any errors it finds. The Check Disk report includes the following items:

- The total amount of space on the disk
- The number of files and directories and how much space they use
- How much space on the disk remains available for files
- In versions 4 and later, the size of each allocation unit, the number of such units on the disk, and the number available for storage
- The size of the computer's memory (up to 640 KB) and how many bytes remain free for use

You can also ask the command to display the name of each file on the disk and to check whether any files are stored inefficiently.

If possible, MS-DOS stores files in adjacent, or *contiguous,* sectors. As files are deleted and new files are stored, however, they can become fragmented (stored in nonadjacent sectors). A fragmented file isn't a cause for worry; the worst that can happen is that MS-DOS will take slightly longer to read the file. If several files on a floppy disk are fragmented, you can restore them to contiguous sectors by copying all the files to an empty, formatted floppy disk with the Copy command. (Remember, don't use the Diskcopy command because it makes a faithful sector-by-sector copy of the floppy disk, storing the files in exactly the same nonadjacent sectors in which they are stored on the original floppy disk.)

The Check Disk command has four parameters:

chkdsk <drive><filename> /V /F

<drive> is the letter, followed by a colon, of the drive that contains the disk to be checked. If you omit <drive>, MS-DOS checks the disk in the current drive.

<filename> is the name of the file whose storage you want MS-DOS to check. MS-DOS displays a message if the file is stored in noncontiguous sectors. You can use wildcard characters to check a set of files.

/V displays the name of each directory and file on the disk.

/F tells MS-DOS to correct any errors it finds in the directory if you so specify when the error is found.

Examples of Checking a Disk

Check the floppy disk in drive A by typing this:

```
C:\>chkdsk a:
```

Press any key if MS-DOS prompts you to insert a floppy disk in drive A.

MS-DOS displays its report. (The report you see might be different; this example is for a 360-KB floppy disk.)

```
Volume DOSDISK      created on 01-05-1995 12:00a
Volume Serial Number  is 1BC6-425F

    362496 bytes total disk space
    350208 bytes in 10 user files
     12288 bytes available on disk

      1024 bytes in each allocation unit
       354 total allocation units on disk
        12 available allocation units on disk

    655360 total bytes memory
    562384 bytes free
```

Note The Check Disk report for MS-DOS 6.2 and later includes a paragraph at the end of the report advising you to use the Scan Disk command instead of Check Disk. See Chapter 9 for more information on Scan Disk.

To check the floppy disk in drive A, and to check whether all files on it are stored in contiguous sectors, type this:

```
C:\>chkdsk a:*.*
```

MS-DOS displays the same report as the preceding but adds the following message:

```
All specified file(s) are contiguous
```

If any files were stored in noncontiguous sectors, MS-DOS would display their names and the number of noncontiguous blocks of storage in place of this message.

To check the floppy disk in drive A and at the same time display the name of each file on it, type this:

```
C:\>chkdsk a: /v
```

MS-DOS displays the name of each file on the floppy disk and then adds its usual report of disk space and memory available. If the list of files scrolls off the top of the screen, remember that, to view it all, you can freeze the display by pressing the Pause key or Ctrl-Num Lock, or use the More command.

Note If the files on the floppy disk are organized into directories, the /V parameter of the Check Disk command lists the directories, as well as the files they contain. Directories are described in Chapter 8, "A Tree of Files."

You can combine the Check Disk parameters in one command; for example, *chkdsk a:*.* /v* would check the floppy disk in drive A, check all files on it for fragmentation, and display the names of all files.

If the Check Disk command finds an error in the directory, it displays a message such as *xx lost allocation units found in yy chains. Convert lost chains to files (Y/N)?* Although this message might look confusing, it simply means that Check Disk has found some storage units on the disk that have been used but, because of a program or system problem, aren't linked with any particular files. If you type the Check Disk command with the /F parameter and then type *y* when the *Convert lost chains...* message appears, MS-DOS turns these "lost" units into files with the file name and extension FILE*nnnn*.CHK (where *nnnn* is a number such as 0001). Depending on the type of error that caused these units to be lost, MS-DOS may or may not be able to recover the data in them. When the Check Disk command is complete, you can look at the contents of these files with the MS-DOS Editor and decide whether or not they contain information you want to save. If you don't want to save the information, delete the files to make the storage space available for new files.

PREVIOUS NEXT

Assigning or Changing a Disk's Volume Label

The Label command assigns, changes, or deletes the volume label of a floppy disk or a hard disk. It has two parameters:

label <drive><newlabel>

<drive> is the letter, followed by a colon, of the drive (such as *a:*) that contains the disk whose volume label is to be changed.

<newlabel> is the volume label to be assigned to the disk in <drive>.

If you omit <drive>, MS-DOS assumes you want to change the label of the disk that's in the current drive. If you omit <newlabel>, MS-DOS prompts you to enter the new label.

Examples of Changing a Disk's Volume Label

At the beginning of this chapter, you used the /V option of the Format command to assign the volume label DOSDISK to a floppy disk. Sometimes you'll want to change a volume label. You can do so with the Label command. Put your sample floppy disk in drive A, and then type the Label command:

```
C:\>label a:
```

MS-DOS responds with something like this:

```
Volume in drive A is DOSDISK
Volume Serial Number is 2F49-1AFF
Volume label (11 characters, ENTER for none)? _
```

The sample disk isn't really an MS-DOS disk, so change its name to MYDISK by typing this:

```
mydisk
```

and press Enter.

If the floppy disk had a volume label and you wanted to delete it, you would reply to the prompt by pressing the Enter key without typing a name. MS-DOS would next ask you *Delete current volume label (Y/N)?* You would then reply *y* to delete the volume label.

Displaying a Disk's Volume Label

The Volume (vol) command also displays the volume label and volume serial number of a hard disk or a floppy disk but doesn't ask for a new label as the Label command does. If you assign descriptive volume labels to your floppy disks when you format them, you can use the Volume command to make sure that you're using the correct floppy disks. It's faster and easier than checking the directory.

The Volume command has one parameter:

vol <drive>

<drive> is the letter, followed by a colon, of the drive (such as *a:*) that contains the floppy disk whose volume label is to be displayed. If you omit <drive>, MS-DOS displays the volume label of the disk in the current drive.

To display the volume label of the floppy disk in drive A, type this:

```
C:\>vol a:
```

MS-DOS displays the volume label:

```
Volume in drive A is MYDISK
Volume Serial Number is 2F49-1AFF
```

If the floppy disk had no volume label, MS-DOS would respond *Volume in drive A has no label,* followed by the volume serial number.

PREVIOUS NEXT

Chapter 7: Managing Your Devices

Overview

Data flows into and out of a computer system through pieces of equipment called *devices*. Devices are categorized by whether they handle data coming in (*input*) or going out (*output*) or both. The keyboard, for example, is an input device; the computer gets information from it. A printer is an output device; the computer sends information to it. A disk drive is both an input device and an output device; the computer can either read a file from a disk or write a file onto a disk.

Some devices, such as the keyboard, don't usually need much attention from you because MS-DOS requires no special instructions to operate them. You can, however, fine-tune their performance, as you'll see later in this chapter. Other devices can be operated in different ways and so allow you to tell MS-DOS how you want to use them. For example, modems follow certain rules in transferring information, and some displays allow you to change the number of lines shown on the screen. With MS-DOS, you can set up a modem to work as it should and just as easily alter your display to work as you want.

Displays, keyboards, printers, and the computer's communications channels, called *ports*, can all be used in a variety of ways. This chapter shows you how to do the following with the MS-DOS commands that help you manage your system:

- Check on system memory with the Mem command
- Clear the screen with the Clear Screen (Cls) command
- Set with the Mode command the speed at which your keyboard repeats a key that is held down
- Specify the number of characters and lines appearing on the screen with the Mode command
- Control the width and line spacing of your printer with the Mode command
- Define the settings of the communications ports with the Mode command
- Copy from a device to a file or to another device with the Copy command
- Enable MS-DOS to print graphics with the Graphics command
- Change your keyboard layout for use with a different language with the Keyboard (Keyb) command

Device Names

Just as files have names, so do devices. You can use a device name in many MS-DOS commands just as you would use a file name. MS-DOS assigns all device names, however. You can't name a device yourself. Figure 7-1 on the following page shows the devices that make up a typical system, with the names assigned to them by MS-DOS.

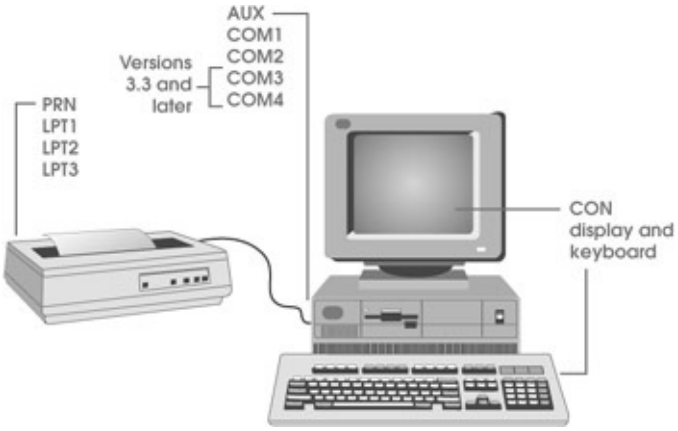


Figure 7-1: MS-DOS device names.

CON is short for *Console*. It is both an input device and an output device and refers to both the keyboard (input) and the display (output). Because the keyboard is input only and the display is output only, MS-DOS can tell which one to use by the way you use the name CON in a command.

PRN is short for *Printer*. It is an output device and refers to the parallel printer that MS-DOS uses unless you specify otherwise (much as MS-DOS looks for files on the current drive unless you specify otherwise). You can attach as many as three parallel printers (named LPT1, LPT2, and LPT3) to your computer; PRN usually means LPT1.

AUX is short for *Auxiliary.* It is for both input and output and refers to the communications port that MS-DOS uses unless you instruct otherwise. You can attach devices to one or two communications ports, named COM1 and COM2, with any version of MS-DOS; if you're using version 3.3 or later, you can attach devices to up to four communications ports (COM1 through COM4). Unless you or a program specifies otherwise, MS-DOS assumes that AUX means COM1. On a typical system, COM1 could be used for a modem, and COM2 could be used for a serial printer—or vice versa.

MS-DOS reserves these names for use with devices only; you cannot give any of these names to a file.

PREVIOUS NEXT

Preparing for the Examples

Note Devices on a network are shared resources set up to respond to requests from more than one authorized user. If your computer is part of a network, check with your network administrator before attempting to manipulate or change the settings of shared printers or other shared devices.

Devices often need very specific setup instructions and operating parameters. All the examples in this chapter work with IBM and IBM-compatible personal computers, displays, and printers. If you're not using one of these machines, you might need to use different instructions to manage your devices. Refer to your documentation for specific information.

When you try the examples, make sure the devices you name are attached to the system and are turned on. You won't hurt anything by entering a command naming a device that isn't present or isn't ready, but the command may cause an error that requires you to restart MS-DOS.

PREVIOUS NEXT

Checking System Memory with the Mem Command

Even though computer memory isn't a device as you would normally think of the term, it can be both useful and reassuring to know how much memory a system contains, what programs have been loaded into memory, and how much can be used by another program. Beginning with version 4 of MS-DOS, you can use the Mem command to request a report on memory and memory usage. Type this:

```
C:\>mem
```

MS-DOS quickly produces a report like this one:

```
Memory Type        Total =   Used  + Free
_________  ___        ___        ___
Conventional       640K      26K   614K
Upper              147K      53K    94K
Adapter RAM/ROM    237K      237K    OK
Extended (XMS)    3072K     2424K   648K
Expanded (EMS)      OK        OK     OK
_______      ___       ___     ___
Total memory      4096K     2740K  1356K

Total under 1 MB   787K      79K   708K

Largest executable program size      614K  (629120 bytes)
Largest free upper memory block       64K   (65552 bytes)
MS-DOS is resident in the high memory area.
```

This is the type of report you see with version 6.0 on an IBM or compatible computer that has 640 KB of regular, or conventional, memory plus 3.4 MB of extended memory (4 MB total memory).

The upper part of the report—the table—tells how much memory of each type is installed, how much is used, and how much is free:

- Conventional memory is the regular memory used by MS-DOS and application programs. The report shows that 640 KB of memory is installed in the computer; 26 KB is being used and 614 KB is free (available for use by other programs).
- Upper memory is the memory previously (before version 5) available only to devices such as displays but now available to programs. The report shows that the computer has 147 KB of upper memory; 53 KB is being used and 94 KB is free.
- Adapter RAM/ROM is memory installed on printed-circuit cards that operate devices (such as display adapters) attached to the computer. The report shows that the cards installed in the computer have a total of 237 KB of memory, and all of it is being used. (In versions 6.2 and later, the memory type *Adapter RAM/ROM* is identified as *Reserved*).

- Extended (XMS) memory is additional memory installed in a computer that can be used by application programs, as well as by MS-DOS. The report shows that 3072 KB of extended memory is installed in the computer; 2424 KB is being used and 648 KB is free.

> **Note** Expanded (EMS) memory is anothertype of additional memory that can be installed in a computer. It also can be used under certain circumstances by both application programs and MS-DOS. Mem reports the amount of expanded memory, if you have any. See Appendix C, "MS-DOS Command Reference," for a full description.

The middle two lines of the report summarize the total memory figures. Following these lines there are two lines that tell you the largest block of memory available in conventional memory and the largest block of memory available in upper memory. The figures for the largest blocks of conventional and upper memory available will vary with the amount of memory installed in a computer, the version of MS-DOS it runs, and the types of programs being used.

The last line of the report confirms that MS-DOS is loaded in the *high memory area*. (The high memory area, or *HMA*, is a special portion of extended memory.) This information is really quite important: It tells you that MS-DOS has placed a large part of itself outside the computer's 640 KB of conventional memory in order to give application programs more room to work. In MS-DOS, this ability to use high memory is a significant feature available only in versions 5 and later. More details on extended memory and high memory are in Chapter 17, "Tailoring Your System."

In MS-DOS version 6, the Mem command includes five parameters, most of which cause MS-DOS to produce a more detailed report of memory use. The detailed reports probably won't seem very meaningful unless you're interested in programming or in learning about the inner workings of your computer. The parameters are discussed in Appendix C, "MS-DOS Command Reference."

Clearing the Screen

Sometimes you might want to erase distracting clutter from the screen: old directory listings, perhaps, or the display of commands MS-DOS has already completed for you. You can clean things up with the Clear Screen (Cls) command, which erases everything on the screen and then displays the system prompt in the upper left corner.

The Clear Screen command has one form:

cls

To test it, type its name:

```
C:\>cls
```

The screen is cleared, except for the system prompt, which is placed in the upper left corner of the screen.

PREVIOUS NEXT

Fine-Tuning the Keyboard

Beginning with version 4 of MS-DOS, you can control how quickly a keystroke repeats when you press and hold a key. To change the way your keyboard operates, you use the Mode command, an all-purpose device-control command that lets you tell MS-DOS how you want to use not only your keyboard but many other parts of your computer system, including your display and printer.

To control your keyboard, the Mode command has three parameters:

mode con rate=<speed> delay=<pause>

con is the name of the console device (here, the keyboard).

rate=<speed> specifies the speed at which MS-DOS is to repeat the keystroke. You can specify any number from 1 through 32, with 32 being the fastest rate.

delay=<pause> specifies how long MS-DOS is to wait before starting to repeat the keystroke. You can specify 1 for a 0.25-second delay, 2 for a 0.5-second delay, 3 for a 0.75-second delay, or 4 for a 1-second delay.

You must specify both a rate and a delay. For example, the average keyboard repeats about 20 times per second with a delay of 0.5 second before repeating begins. To see the effect of this version of the Mode command, slow the keyboard repeat rate and delay. Type this:

```
C:\>mode con rate=1 delay=4
```

and try holding down the X key to see the result.

If you are a fast typist, impatient with a too-slow keyboard, set the rate to a high value, such as 32, and set the delay to a low number, such as 1. To slow keyboard repetition, lower the rate and, if you want, increase the delay time. To reset the keyboard to its regular repeat rate and delay, type this:

```
C:\>mode con rate=20 delay=2
```


PREVIOUS NEXT

Controlling the Display

A number of different displays and display adapters (the printed-circuit cards that operate the display) are used with IBM personal computers and compatible machines. Figure 7-2 describes the most commonly used types of displays and display adapters.

Short Name	Product Name	Description
MDA	Monochrome Display Adapter	Text only, medium resolution, one color (usually green on a dark background).
CGA	Color/Graphics Adapter	Text and graphics, low resolution, up to 16 colors.
Hercules	None (works with monochrome display)	Displays graphics on monochrome display. (Not compatible with CGA. An application program must specifically support the Hercules board, but most major applications do because of its popularity.)
EGA	Enhanced Graphics Adapter	Text and graphics, medium resolution, up to 16 colors.
MCGA	Multicolor Graphics Array	Text and graphics, low to medium resolution, up to 256 colors.
VGA	Video Graphics Array	Text and graphics, medium to high resolution, up to 256 colors.
SVGA	Super Video Graphics Array	Text and graphics, medium to very high resolution, up to 256 colors and beyond.

Figure 7-2: Common displays and adapters.

The Mode command has several display-related options. Which one you use depends on the type of display you have and how much you want to see on the screen.

Normally, a computer screen displays 80 columns (characters) across and 25 lines down. If you have an EGA or a VGA display, you can, beginning with version 4, change the size of the characters on your display and the number of lines on the screen. When used in this way, the Mode command has three parameters:

mode con cols=<columns> lines=<lines>

con is the name of the console device (here, it means the display).

cols=<columns> sets the number of characters displayed on each line. The <columns> parameter can be either 40 (for extra-large characters) or 80 (for normal-size characters).

lines=<lines> sets the number of lines on the screen. <lines> can be 25, 43, or 50, but not all display adapters can handle all three choices.

To set the screen to display 40 characters per line, type this:

```
C:\>mode con cols=40
```

To display 80 characters per line and 43 lines per screen, type this:

```
C:\>mode con cols=80 lines=43
```

Note If you try this command and MS-DOS displays the message *Function not supported on this computer,* the command does not work on your computer system. If you see the message *ANSI.SYS must be installed to perform requested function,* you need to add a reference to a program named ANSI.SYS in the startup file named CONFIG.SYS. Chapter 17, "Tailoring Your System," tells you how to do this.

In all versions of MS-DOS, you can also use another form of the Mode command to control the number of characters per line. When used to control the number of columns, the Mode command has one form:

mode <characters>

<characters> is either 40 or 80.

To display 40 columns with this command, type this:

```
C:\>mode 40
```

MS-DOS clears the screen and displays the system prompt in large characters in the upper left corner of the screen.

Controlling the Printer Width and Spacing

A dot-matrix printer normally prints a maximum of 80 characters per line and 6 lines per inch. It can also print in a smaller type, called *condensed,* that fits 132 characters on a line. This ability to change widths can be useful for printing documents wider than 80 characters. The printer can also print 8 lines per inch, to fit more lines on a page.

Note If you have a laser printer, note that the documentation that came with the printer tells you how to set the printer to work with MS-DOS and how to control the printer's characteristics, such as line and character spacing. The examples here most likely won't work unless your laser printer can emulate an IBM or an Epson dot-matrix printer.

If your printer is attached to a parallel port, you can use the Mode command to specify the line width (80 or 132) and spacing (6 or 8).

When used to control a dot-matrix printer attached to a parallel port, the Mode command has three main parameters, which you type in one of the following two forms. The first form applies to versions 4 and later; the second applies to any version of MS-DOS:

mode <printer> cols=<width> lines=<spacing>

or:

mode <printer> <width>,<spacing>

<printer> is the name of the printer, followed by an optional colon (*lpt1:, lpt2:,* or *lpt3:*).

<width> is either 80 or 132.

<spacing> is either 6 or 8. Notice the comma preceding <spacing> in the second form of the command.

You must always include <printer>. You can omit either <width> or <spacing>, and MS-DOS will leave the current width or spacing unchanged. In the second form of the command, however, if you omit <width> you must still type the comma before <spacing> to tell MS-DOS you omitted <width>.

Examples of Controlling a Dot-Matrix Printer

The following examples work with a dot-matrix printer connected to the computer's first parallel printer port (LPT1:). If this describes your equipment, be sure your printer is turned on before proceeding with the following examples. If you have a different type of printer or printer setup, you might prefer to read through the examples or skip ahead to the next section, "Controlling the Serial Communications Port."

Note Because different versions of MS-DOS accept different forms of the command, the examples first show the command used with versions 4 and later and then

show the command form for earlier versions of MS-DOS.

To cause a printer attached to LPT1 to print in small type (up to 132 characters per line if the printer can do so), use the command appropriate to your version of MS-DOS, designated at the top of the next page.

If you have version 4 or later, type this:

```
C:\>mode lpt1: cols=132
```

If you have an earlier version of MS-DOS, type this:

```
C:\>mode lpt1: 132
```

MS-DOS replies with a display like this one:

```
LPT1: not rerouted

LPT1: set for 132

No retry on parallel printer time-out
```

The second line of this message, which you see with any version of MS-DOS, tells you that the command worked correctly and your printer is now set for small characters. If you have version 4 or later, the other lines are additional messages telling you that MS-DOS will, indeed, use your parallel printer for output and that, if the printer is busy or turned off, it will not keep trying to send data to it.

To test the new printer setting, print the contents of the screen by pressing Shift-PrtSc. (If you have an IBM PS/2 keyboard, press Print Screen, not Shift-PrtSc.)

To set the spacing of the same printer to 8 lines per inch but leave the width unchanged, use one of the following commands.

If you have version 4 or later, type this:

```
C:\>mode lpt1: lines=8
```

If you have an earlier version of MS-DOS, type this:

```
C:\>mode lpt1: ,8
```

MS-DOS replies with a message like this:

```
LPT1: not rerouted

Printer lines per inch set

No retry on parallel printer time-out
```

(Again, if you don't have version 4 or later, your message is shorter.)

To see the effect of this setting, print the contents of the screen again by pressing Shift-PrtSc. This time the text is printed both in small type (as set in the previous example) and with closer line spacing.

To restore the printer to normal width and line spacing with version 4 or later, type this:

```
C:\>mode lpt1: cols=80 lines=6
```

If you have an earlier version of MS-DOS, type this:

```
C:\>mode lpt1: 80,6
```

MS-DOS displays a message similar to those you've already seen:

```
LPT1: not rerouted

LPT1: set for 80

Printer lines per inch set

No retry on parallel printer time-out
```


PREVIOUS NEXT

Controlling the Serial Communications Port

Serial communications is controlled by several characteristics, or *communications parameters,* that define how fast and in what form data is transmitted. Different devices often require different parameter settings. The parameters of your serial port must match those of the device or computer service with which you want to communicate. Before you can use a communications port, you must set these parameters with the Mode command.

The main communications parameters you can set include these:

- *Baud,* how quickly characters are sent or received
- *Parity,* the kind of error-checking technique used
- *Data bits,* the number of electrical signals required to define a character
- *Stop bits,* the amount of time between electrical signals that marks the end of a character

A more complete definition of these parameters is beyond the scope of this book. Figure 7-3 lists the main parameters you can set with the Mode command. The documentation of the device or computer service you want to use shows the required setting. Compare each of these settings with the values MS-DOS assumes, shown in Figure 7-3, to see which parameters you must change.

Name	Valid Settings	How You Specify	Value MS-DOS Assumes
Baud	110, 150, 300, 600, 1200, 2400, 4800, 9600, 19,200 [a]	You can abbreviate to the first two digits (11 for 110, 24 for 2400)	None (you must set a value)
Parity	None, Odd, Even, Mark, [b] or Space[b]	N, O, E, M, or S	Even (E)
Databits	5,[b] 6,[b] 7,[b] or 8	5, 6, 7, or 8	7
Stopbits	1, 1.5,[b] or 2	1, 1.5, or 2	2 if baud = 110; 1 otherwise

[a]19,200 is available from version 3.3 onward and can be used only with a computer capable of that speed.

[b]Versions 4 and later.

Figure 7-3: Serial communications parameters.

When used to set the parameters of a serial communications port, the Mode command has the following form in versions 4 and later:

mode <port> baud=<baud> parity=<parity> data=<databits> stop=<stopbits>

In earlier versions, the command is the following:

mode <port> <baud>,<parity>,<databits>,<stopbits>

<port> is the name of the communications port followed by an optional colon—*com1:* through *com4:* (*com1:* or *com2:* if you're using a version earlier than 3.3). The remaining parameters, separated by commas, are those described in Figure 7-3.

You must specify a value for <baud> each time you enter this form of the Mode command. MS-DOS assumes the values for the other parameters listed in the last column of Figure 7-3 unless you specifically change them; you needn't specify these parameters unless the device or service with which you want to communicate requires values different from those that MS-DOS assumes.

If you omit any parameter from the second form of the Mode command shown above, you must still type the comma that precedes it, to show MS-DOS that you omitted the parameter.

Examples of Controlling the Serial Communications Port

These examples show you different uses of the Mode command. Don't enter them unless you have a serial communications port and you want to change the settings.

To set the baud rate for COM1 to 1200 (and let MS-DOS assume values for the other parameters), you would type one of the following commands.

If you have version 4 or later, type this:

```
C:\>mode com1: baud=1200
```

If you have an earlier version of MS-DOS, type this:

```
C:\>mode com1:  1200
```

MS-DOS would reply by reporting the current setting of each parameter:

```
COM1: 1200,e,7,1,-
```

This report informs you that <baud> is 1200, <parity> is even, <databits> is 7, and <stopbits> is 1. The hyphen at the end tells you that MS-DOS will not keep trying to send to a device that isn't ready but will stop after a brief time.

To set <baud> for COM1 to 2400, set <parity> to none, leave <databits> set to 7, and set <stopbits> to 2, you would type one of the following commands.

If you have version 4 or later, type this:

```
C:\>mode com1: baud=2400 parity=n stop=2
```

If you have an earlier version of MS-DOS, type this:

```
C:\>mode com1: 2400,n,, 2
```

(Note the two commas before the 2, telling MS-DOS that you omitted <databits>.) MS-DOS confirms the settings:

```
COM1: 2400,n,7,2,-
```


PREVIOUS NEXT

Connecting a Serial Printer

If you want to use a serial printer attached to a communications port, you must use the Mode command to tell MS-DOS to send printer output to the communications port instead of to the regular (parallel) printer port. This is called *redirecting* or *rerouting* the printer output. (You might recall from the earlier examples of controlling a dot-matrix printer that the MS-DOS responses to the Mode command included the line *LPT1: not rerouted*. Here, you can see the meaning of that message: MS-DOS was reporting that output to the regular printer port was not redirected to a serial port.) Before you redirect the printer output, you must first set the parameters of the serial communications port to the values required by the printer, as described in the preceding section.

When used to redirect printer output to a serial communications port, the Mode command has one form:

mode <printer>=<port>

<printer> is the name of the printer port (*lpt1:, lpt2:,* or *lpt3:*) whose output is to be redirected.

<port> is the name of the serial communications port (*com1:* or *com2:* in all versions of MS-DOS, *com1:* through *com4:* in 3.3 and later versions). To redirect printer output, you must enter both parameters.

Example of Connecting a Serial Printer

To redirect printer output from LPT1 to serial port COM1, you would first set the serial port to match the communications parameters of your printer and then type this:

```
C:\>mode  lpt1:=com1:
```

MS-DOS would acknowledge the change:

```
LPT1:  rerouted to COM1:
```

Now all output that would normally go to LPT1: would be sent to COM1: instead. To cancel the redirection and restore the printer output to LPT1:, you would type this:

```
C:\>mode  lpt1:
```


PREVIOUS NEXT

Finding Out About Your System

You have, up to this point, seen a number of ways to use the Mode command to control the devices on your system. But just as it's useful to know how to control your devices, it's also helpful to know what devices MS-DOS is prepared to use and how it "sees" them. Beginning with version 4, you can use the Mode command not only to describe devices to MS-DOS but to find out about them.

To check on the status of any or all devices on your system, you can use this command:

mode <device> /status

<device> is the name of a particular device you want to check on. If you omit <device>, MS-DOS reports on all the devices it recognizes.

/status, which you can abbreviate as */sta,* is needed only when you want to check the status of a parallel printer you've redirected. (The */sta* is needed because simply typing *mode lpt1:* would cancel the redirection.)

To check on the status of a single device, you type *mode* and the name of the device. For example, if the display were set to show 80 characters per line and 43 lines per screen, the command

```
C:\>mode  con
```

would produce a report like this:

```
Status for device CON:
- - - - - - - - - - -
Columns=80
Lines=43

Code page operation not supported on this device
```

(The message *Code page operation not supported on this device* simply means that MS-DOS, on this computer, has not been told to recognize international language, date, decimal, and currency conventions. If you need this capability, refer to the online help for the Mode command.

To check on all the devices on your system, you would type this:

```
C:\>mode
```

and MS-DOS would respond with a report like this:

```
Status for device LPT1:
- - - - - - - - - - -
LPT1: not rerouted
Retry=NONE
```

```
Code page operation not supported on this device

Status for device LPT2:
- - - - - - - - - - -
LPT2: not rerouted

Status for device LPT3:
- - - - - - - - - - -
LPT3: not rerouted

Status for device CON:
- - - - - - - - - - -
Columns=80
Lines=25

Code page operation not supported on this device

Status for device COM1:
- - - - - - - - - - -
Retry=NONE
```

Although parts of the report might look unfamiliar, here's what it means:

- The report covers five devices: three parallel ports (LPT1, LPT2, and LPT3), the display (CON), and a serial port (COM1).
- The message *LPTx*: *not rerouted* simply means that information sent to any of these output channels has not been diverted to a serial port.
- The message *Retry=NONE* means that, if the device connected to a port is busy, MS-DOS will not keep trying to send information to the device (a printer or modem, for example).
- The *Code page...* message means that MS-DOS has not been told to recognize international language, date, decimal, and currency conventions.

Copying from a Device to a File or Another Device

As you saw in earlier examples, you can use the Copy command to copy from a device to a file. You have used this technique several times to create sample files by copying from the keyboard to a file, and you will find it handy for creating short text files.

You can also copy from one device to another. Copying from the keyboard to the printer, for example, is a convenient way to print short notes or lists.

When you copy from one device to a file or another device, MS-DOS continues to copy until it encounters the character (Ctrl-Z) that marks the end of a file. Whenever you copy from the keyboard, you can send this end-of-file character by pressing the key labeled F6 and then pressing the Enter key (or, as you've done before, by pressing Ctrl-Z and Enter).

When used to copy from a device to a file or another device, the Copy command has two parameters:

copy <source> <target>

<source> is the name of the source device.

<target> is the name of the target file or device.

Examples of Copying from a Device to a File or Another Device

To copy from the keyboard (CON) to the printer (PRN), make certain the printer is turned on and type this:

```
C:\>copy con prn
```

Now everything you type will be both displayed and sent to the printer. Type a few lines, and then end the copy by pressing F6 or Ctrl-Z (shown as ^Z in the example because that's how MS-DOS displays it):

```
These lines are being
copied from the
keyboard to the printer.
^Z
        1 file(s) copied

C:\>_
```

Note If this command doesn't produce a printed copy, you might need to tell the printer to eject the printed page and move to a new page. Try taking your printer off line and pressing the formfeed button.

Printing Graphics Images

Pressing Shift-PrtSc prints the text displayed on either a monochrome or a color display, but it does not print graphics images from a display that is attached to a graphics adapter. The Graphics command enables MS-DOS to print these graphics images on any of several different printers.

You need enter the Graphics command only once. After you enter it, pressing Shift-PrtSc prints everything on the screen of the active display, including graphics images, accented characters, lines, and boxes. Color on noncolor printers is simulated with shading. Depending on the resolution of the display, the graphics image itself may be printed across the page (as you see it on screen), or it may be printed sideways (rotated 90 degrees) and enlarged.

The Graphics command loads a program that increases the amount of memory that MS-DOS uses. The command has four main parameters:

graphics <printer> /R /B /lcd

<printer> can be any of the IBM and non-IBM printers listed in Figure 7-4.

Specify	For Printer Model
color1	IBM Personal Computer Color Printer with a black ribbon or the black band of a color ribbon
color4	IBM Personal Computer Color Printer with a red-green-blue ribbon
color8	IBM Personal Computer Color Printer with a cyan-magenta-yellow ribbon
compact	IBM Personal Computer Compact Printer (MS-DOS versions prior to 4)
graphics	IBM Graphics Printer, Proprinter, Pageprinter, or Quietwriter
graphicswide	IBM Quietwriter or Proprinter with an 11-inch carriage (MS-DOS versions 4 and later)
thermal	IBM PC Convertible Printers (MS-DOS versions 3.3 and later)
hpdefault	Any Hewlett-Packard PCL printer (MS-DOS versions 5 and later)
deskjet	Hewlett-Packard DeskJet (MS-DOS versions 5 and later)
laserjet	Hewlett-Packard LaserJet (MS-DOS versions 5 and later)
	Hewlett-Packard LaserJet Series II (MS-DOS versions 5 and

laserjetII	later)
paintjet	Hewlett-Packard PaintJet (MS-DOS versions 5 and later)
quietjet	Hewlett-Packard Quietjet (MS-DOS versions 5 and later)
quietjetplus	Hewlett-Packard Quietjet Plus (MS-DOS versions 5 and later)
ruggedwriter	Hewlett-Packard Rugged Writer (MS-DOS versions 5 and later)
ruggedwriterwide	Hewlett-Packard Rugged Writer with wide carriage (MS-DOS versions 5 and later)
thinkjet	Hewlett-Packard ThinkJet (MS-DOS versions 5 and later)

Figure 7-4: Printers supported by the Graphics command.

/R (Reverse) tells MS-DOS to print the screen as you see it—in other words, light characters on a dark background.

/B tells MS-DOS to print the background color if you have specified *color4* or *color8* for <printer>. If you don't specify /B, then MS-DOS doesn't print the background color.

/lcd tells MS-DOS to print the contents of the liquid crystal display of the IBM PC Convertible computer.

When you enter the Graphics command, MS-DOS loads the program, adds it to the parts of the system kept in memory, and displays the system prompt. You needn't enter the command again until the next time you start MS-DOS.

If you're using a color display and you have a printer that can print graphics, you can test the Graphics command by entering the command and all appropriate parameters, loading a graphics image on your display, and then pressing Shift-PrtSc.

PREVIOUS NEXT

Changing the Keyboard Layout

Whenever you start MS-DOS, it assumes the language and keyboard layout of the country for which your computer was manufactured. Since the mid-1980s, however, successive releases of MS-DOS have offered increasing amounts of support for languages other than its original American English. As part of this international support, MS-DOS versions 3 and later include a Keyboard (Keyb) command that changes the keyboard layout to accommodate the special characters of different languages and to match the arrangement of keys used in different countries.

The first time the Keyboard command is carried out, it loads a small program that increases the size of MS-DOS in memory by about 5000 bytes. Subsequent Keyboard commands then tell MS-DOS to use different sets of characters that correspond to different country-specific keyboard layouts.

Keyboard Layouts

When you use the Keyb command, you tell MS-DOS which keyboard layout you want by adding a two-letter country code to the command. If you're using any of IBM's PC-DOS releases 3.0 through 3.2 or MS-DOS version 3.2, you can choose from six layouts by typing the command as shown in the center column:

Code	Command	Country/Language
dv	keybdv	Dvorak (an alternative English-language layout)
fr	keybfr	France
gr	keybgr	Germany
it	keybit	Italy
sp	keybsp	Spain
uk	keybuk	United Kingdom

If you're using version 3.3 or later, you have a significantly expanded group of keyboard codes to choose from because these versions of MS-DOS offer much more international support. The following table shows the keyboard codes readily available for IBM and compatible computers in most countries.

Code	Command	Country/Language
be	keyb be	Belgium
bg	keyb bg [a]	Bulgaria (version 6.22)
br	keyb br	Brazil (versions 5 and later)
cf	keyb cf	French-speaking Canada

cz	keyb cz	Czech Republic—Czech (versions 5 and later)
dk	keyb dk	Denmark
fr	keyb fr	France
gk	keyb gk[a]	Greece (version 6.22)
gr	keyb gr	Germany
hu	keyb hu	Hungary (versions 5 and later)
is	keyb is[a]	Iceland (versions 6.0 and later)
it	keyb it	Italy
la	keyb la	Latin America
nl	keyb nl	Netherlands
no	keyb no	Norway
pl	keyb pl	Poland (versions 5 and later)
po	keyb po	Portugal
ro	keyb ro[a]	Romania (version 6.22)
ru	keyb ru[a]	Russia (version 6.22)
sf	keyb sf	French-speaking Switzerland
sg	keyb sg	German-speaking Switzerland
sl	keyb sl	Slovakia—Slovak (versions 5 and later)
sp	keyb sp	Spain
su	keyb su	Finland
sv	keyb sv	Sweden
tr	keyb tr[a]	Turkey (version 6.22)
uk	keyb uk	United Kingdom
us	keyb us	United States, Australia, English-speaking Canada
yc	keyb yc	Macedonia and Serbia/Montenegro (version 6.22)
yu	keyb yu	Yugoslavia (versions 5 and later)

[a]See the COUNTRY.TXT file in your DOS directory for more information on using this keyboard code.

Because entering a Keyboard command changes the location of common keys (especially

punctuation marks), the character that results from pressing a key doesn't always match the label on the key, as you'll see shortly.

Typing Accented Characters with Dead Keys

Many languages use *accented characters* that combine an accent mark and a common character (such as Å or ñ). Some of these accented characters are assigned locations on the keyboard; on the French keyboard, for example, you type *è* by pressing the 7 key in the top row of the keyboard (to type the number 7, you press Shift-7).

Often there aren't enough available keys to provide all the accented characters, however, so MS-DOS also uses *dead keys* to combine accent marks and characters. Some typewriters use this same technique, so dead keys might be familiar.

A dead key is one that represents just an accent mark. Pressing a dead key doesn't produce any apparent result, but it tells MS-DOS to combine the accent mark with the next key you press. On the French keyboard, for example, you type *ô* by pressing the key labeled with a { and a [, which is the dead key for the circumflex (*ô*), and then pressing the key labeled O.

If you press a dead key and then press a character that cannot be combined with the accent mark represented by the dead key, MS-DOS beeps and displays the accent mark, followed by the key you pressed, to show you it can't combine them as an accented character. For example, ^p indicates that MS-DOS cannot put a circumflex over the letter P. If the dead key represents the diaeresis (¨), MS-DOS displays a small dot (.) or a filled-in square (■) followed by the second key you pressed in an incorrect dead-key sequence.

To correct the error, backspace to erase the two characters and type the correct dead-key sequence.

Example of Using the Keyboard Command

If you're using MS-DOS version 3.0 through 3.2, type this:

```
C:\>keybfr
```

If you're using MS-DOS version 3.3 or later, type the following to change the keyboard arrangement to French:

```
C:\>keyb fr
```

If MS-DOS responds *Bad or missing Keyboard Definition File,* you need to expand the command to tell MS-DOS where to find the command file it needs. The file is named KEYBOARD.SYS, and it should be in your MS-DOS directory. Retype the command as follows, substituting the name of your MS-DOS directory if it is not C:\DOS:

```
C:\>keyb   fr,,c:\dos\keyboard.sys
```

If you are accustomed to the United States layout, here are the most obvious changes with the

French layout: You must hold down the Shift key to type a number; the locations of two pairs of letter keys are reversed (Q-A and W-Z); M is to the right of L (where the semicolon is located on the American English keyboard); and most symbols and punctuation marks are in different places.

Now that you have switched keyboards, you must follow the new layout. For example, the command to display a directory in wide format is *dir /w*. If you don't follow the French keyboard layout, you'll type something like *dir =z* because the equal sign or another character is where the forward slash used to be, and the W and Z have also changed places.

Suppose you wanted to type the sentence *L'hôtel célèbre est grand* (meaning *The famous hotel is big*). Both the é and the è are on the keyboard (2 and 7, respectively), but the ô is not, so you must use the dead key for the circumflex (to the right of P). And remember, the Q and the A keys have changed places. Here, step by step, is how you would type the phrase:

C:\>L

For the apostrophe, press the 4 key in the top row. After the apostrophe, type *h:*

C:\>L'h

Now you have to use the circumflex dead key, which is to the right of the P. Press the circumflex dead key ([). Nothing happens yet. Now type *o*; MS-DOS displays ô. Continue by typing *tel c*:

C:\>L'hôtel c

For é, press the 2 key in the top row; then type *l*:

C:\>L'h\ôtel cél

For è, press the 7 key in the top row; then type *bre est gr*:

C:\>L'h\ôtel célèbre est gr

And finally, press Q to get the a, and type *nd*:

C:\>L'h\ôtel célèbre est grand

Although that's a painfully long list of instructions just to type a simple sentence, in fact it goes quite quickly after a bit of practice. Press Esc to clear the line, and type *keyb us* to return to the United States layout.

If your computer was manufactured for use in the United States and you have an IBM keyboard or a strict compatible, you can switch from a foreign layout to the United States configuration by pressing Ctrl-Alt-F1 (hold down both the Ctrl and Alt keys and press the F1 function key). Notice that MS-DOS is still set to the foreign layout, even though functionally it uses the United States configuration. To switch back to the foreign configuration, press Ctrl-Alt-F2.

Chapter 8: A Tree of Files

Overview

As you have seen, when MS-DOS formats a disk it creates a directory that describes each of the files on the disk. The directory holds a fixed number of entries: 112 on a 360-KB or 720-KB floppy disk, 224 on a 1.2-MB or 1.44-MB floppy disk, 240 on a 2.88-MB floppy disk, and 512 or more on a hard disk. (The number varies with the size of the disk.) To make your computer filing system more flexible, MS-DOS lets you create additional directories, called *subdirectories,* on a disk. The subdirectories divide the disk into different storage areas, each of which you can use as if it were a different disk.

To distinguish the main directory that MS-DOS creates from the subdirectories that you create, the main directory is known as the *root directory* (*root* because a multilevel directory structure can grow from it).

As you add levels to your file structure, a block diagram would show it spreading from the root directory and branching to other directories, like a tree branching from its root. This type of file structure is often called a *tree-structured* file system.

PREVIOUS NEXT

Defining a Subdirectory

To MS-DOS, a subdirectory is simply a file that contains directory entries; these entries are identical in form to the entries in the main directory, but there is no limit to the number of entries you can put in a subdirectory.

You name a subdirectory the way you name any other file, but because the subdirectory defines other files, you cannot use the normal file commands to copy or erase a subdirectory. This chapter shows you how to use several commands that enable you to do the following:

- Create a subdirectory with the Make Directory (md) command
- Change or display the name of the current directory with the Change Directory (cd) command
- Move files from one directory to another and rename directories with the Move command (6.0 and later versions)
- Delete a subdirectory with the Remove Directory (rd) command
- Delete a directory structure with the Deltree command
- Display a list of files and the directories on a disk with the Directory (dir) and Tree commands
- Tell MS-DOS where to look for a command file if it's not in the current directory by using the Path command
- Tell MS-DOS where to look for a data file if it's not in the current directory by using the Append command (3.3 and later versions)

Using these features of MS-DOS, you can create and manage a computer filing system that is tailored to the way you work.

PREVIOUS NEXT

Preparing for the Examples

The examples in this chapter require one formatted floppy disk. Put the formatted floppy disk in drive A. If any files from earlier examples are stored on the floppy disk, delete them by typing the following; be sure to include the *a:* in the command:

```
C:\>del a:*.*
```

MS-DOS asks you to confirm that you want to erase all the files:

```
All files in directory will be deleted!
Are you sure (Y/N)?_
```

Before you respond, check the command you entered and be certain that you typed the drive letter (*a:*). If you didn't, press Ctrl-Break to cancel the Delete command, and retype the command correctly. If you did type the drive letter, respond *y.* Because a mistake in using the Delete command in this way could cause the loss of valuable files, you must also press the Enter key after typing *y* before MS-DOS will carry out the command.

After you have deleted the files, change the current drive to A by typing this:

```
C: \>a:
```

This completes the preparation.

PREVIOUS NEXT

Creating a Multilevel File Structure

Suppose you work at a small company and provide services to two departments, Marketing and Engineering. You keep all your papers in a file drawer. You keep miscellaneous items in the front of the drawer, and dividers labeled MKT and ENG separate the parts where you store papers that relate to each department.

Now that you're going to be using a computer, you can set up your computer file system to match your paper files by creating two subdirectories, named MKT and ENG. You can store miscellaneous computer files in the main, or root, directory of the disk, and you can store the files that relate to each department in separate subdirectories. Figure 8-1 on the next page shows the filing cabinet and a block diagram of this corresponding MS-DOS file structure.

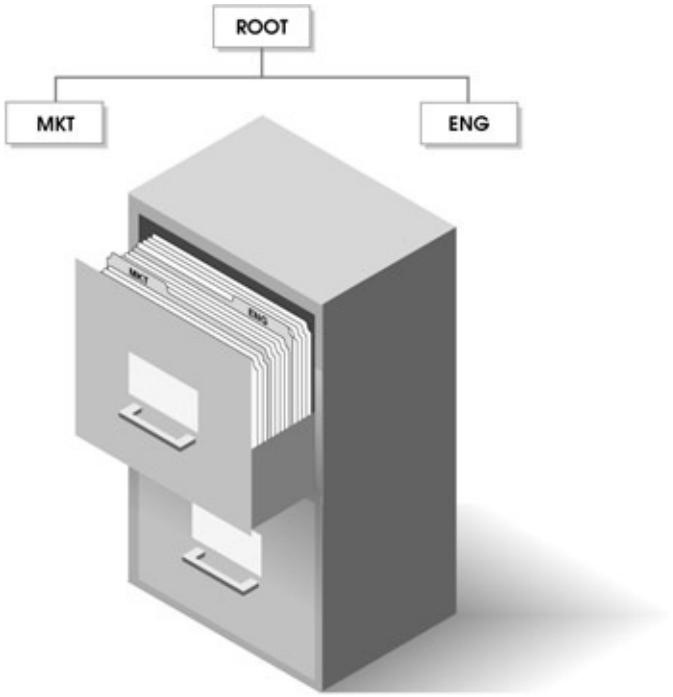


Figure 8-1: Two-level file system.

Creating a Subdirectory

The Make Directory (md or mkdir) command creates a subdirectory. The only parameter that you must include is the name of the subdirectory you want to create. The command is described later in more detail; for now, type the following to create two subdirectories, named MKT and ENG:

```
A:\>md mkt

A:\>md eng
```

You can see the subdirectories you just created by displaying the entries in the root directory. Type this:

```
A:\>dir
```

MS-DOS shows two files, named MKT and ENG:

```
Volume in drive A is EXAMPLES 2
 Volume Serial Number is 1839-10EE
 Directory of A:\

MKT         <DIR>     01-05-95  10:54a
ENG         <DIR>     01-05-95  10:54a
       2 file(s)          0 bytes
                    1456640 bytes free
```

Note that the directory identifies the files as subdirectories by displaying <DIR> after their names. The backslash (\) in the third line of the display is the character that MS-DOS uses to refer to the root directory of a disk. You've seen the backslash as part of the system prompt throughout this book; you'll see more of it and its uses in later examples.

Because MKT is a subdirectory, you can display its contents with the Directory command, just as you can display the contents of the root directory. Type this:

```
A:\>dir mkt
```

MS-DOS displays the contents of MKT:

```
Volume in drive A is EXAMPLES 2
 Volume Serial Number is 1839-10EE
 Directory of A:\MKT

.           <DIR>     01-05-95  10:54a
..          <DIR>     01-05-95  10:54a
       2 file(s)      0 bytes
                    1456640 bytes free
```

Even though you just created it, MKT seems to contain subdirectories named . (dot) and .. (double dot). These really aren't subdirectories; they're abbreviations you use to refer to other directories. The dot and double-dot subdirectories do not occur in the root directory. You'll see how these abbreviations are used a bit later.

The Path to a Directory

The third line of the preceding directory display tells you that you're looking at the directory of A:\MKT. The \ (backslash) refers to the root directory, and MKT is the name of the subdirectory whose contents you're displaying. Together, they are called the *path name* of the directory, or just the *path,* because they describe the path MS-DOS follows to find the directory. The path names of the two subdirectories you created, \MKT and \ENG, tell MS-DOS that the subdirectories are in the root directory.

You can also include a path name with a file name, to tell MS-DOS where to find a file. The path name goes just before the file name (after the drive letter, if one is included) and is

separated from the file name by a backslash. For example, if the subdirectory \MKT contained a file named BUDGET.JAN, the full path and file name would be \MKT\BUDGET.JAN.

The Current Directory

Just as MS-DOS keeps track of the current drive, it also keeps track of the current directory. If you start MS-DOS from a hard disk, the current directory after startup is usually either the root directory (C:\) or your MS-DOS directory (C:\DOS). If you start MS-DOS from a floppy disk, the current directory after startup is normally the root directory of the floppy disk drive (A:\).

Just as you can change the current drive, you can change the current directory so that you don't have to type the path name each time you want to work with a directory other than the current one. The Change Directory (cd or chdir) command changes or displays the name of the current directory. If you enter the command with no parameter, it displays the name of the current directory. To see what the current directory is, type this:

```
A:\>cd
```

The current directory is the root directory, so the response is short:

```
A:\
```

It tells you that any command you enter will apply to the root directory of the floppy disk in drive A, unless you specify a different path name. Change the current directory to the subdirectory named MKT by typing this:

```
A:\>cd mkt
```

If your system prompt normally shows the current directory, MS-DOS acknowledges

```
A:\MKT>
```

as soon as it carries out the Change Directory command.

If your system prompt doesn't display the current directory, MS-DOS acknowledges the command merely by displaying the system prompt. But if you display the current directory again by typing this:

```
A>cd
```

MS-DOS responds this way:

```
A:\MKT
```

Note When you work with subdirectories, it's useful to show the current directory as part of the system prompt. If your system prompt doesn't include the current directory, change it: Type *prompt pg.* Chapter 14, "Creating Your Own Commands," tells you how to make this command a normal part of your system's startup routine.

You've changed the current directory to \MKT, so any command you enter applies to the subdirectory MKT in the root directory. Type the Directory command again:

```
A:\MKT>dir
```

MS-DOS displays the entries in the subdirectory \MKT:

```
Volume in drive A is EXAMPLES 2
 Volume Serial Number is 1839-10EE
 Directory of A:\MKT

.          <DIR>     01-05-95  10:54a
..         <DIR>     01-05-95  10:54a
       2 file(s)          0 bytes
                    1456640 bytes free
```

This display is the same as the one you saw earlier when you typed *dir mkt,* but this time you didn't have to name the subdirectory because you had changed the current directory to \MKT.

Using Subdirectories

Your floppy disk now has the directory structure shown in Figure 8-1. You can use each of these directories as if it were a separate disk. The current directory is \MKT. Create a file named SAMPLE.TXT in the root directory by typing the following lines:

```
A:\MKT>copy con \sample.txt
This is a sample file.
^Z
       1 file(s) copied
```

Notice that you included the backslash to tell MS-DOS to put the file in the root directory. You also use the backslash to display the contents of the root directory when it's not the current directory. Type the following:

```
A:\MKT>dir \
```

Again MS-DOS displays the entries in the root directory:

```
Volume in drive A is EXAMPLES 2
 Volume Serial Number is 1839-10EE
 Directory of A:\

MKT         <DIR>     01-05-95  10:54a
ENG         <DIR>     01-05-95  10:54a
SAMPLE   TXT         24 01-05-95  11:13a
       3 file(s)         24 bytes
                    1456128 bytes free
```

The root directory contains two subdirectories and the file you just created.

Copying from One Directory to Another

Because you can treat directories as if they were separate disks, you can copy a file from one directory to another. Copy SAMPLE.TXT from the root directory of drive A to a file named ACCOUNT in the current directory (\MKT) by typing this:

```
A:\MKT>copy  \sample.txt account
        1 file(s) copied
```

You included the path (the backslash, meaning the root directory) in front of SAMPLE.TXT to tell MS-DOS where to find the file; you didn't have to include a path for ACCOUNT because you were putting it in the current directory. Now display the current directory by typing the following:

```
A:\MKT>dir

 Volume in drive A is EXAMPLES 2
 Volume Serial Number is 1839-10EE
 Directory of A:\MKT

.           <DIR>     01-05-95  10:54a
..          <DIR>     01-05-95  10:54a
ACCOUNT              24 01-05-95  11:13a
       3 file(s)         24 bytes
                   1455616 bytes free
```

The file is there. You can copy files from one directory to another as easily as you can copy them from one disk to another.

Just as MS-DOS doesn't confuse two files with the same name on different disks, it doesn't confuse two files with the same name in different directories. MS-DOS can tell the latter apart because their paths are different. You can demonstrate this by copying the file named ACCOUNT from \MKT to the subdirectory \ENG, giving it the same file name. Type the following:

```
A:\MKT>copy account  \eng
        1 file(s) copied
```

You didn't include the file name after the path name of the target directory because you wanted to give the copy the same name as the original. Assure yourself that the file was copied correctly by displaying the directory of \ENG:

```
A:\MKT>dir  \eng
 Volume in drive A is EXAMPLES 2
 Volume Serial Number is 1839-10EE
 Directory of A:\ENG
```

```
.          <DIR>     01-05-95  10:54a
..         <DIR>     01-05-95  10:54a
 ACCOUNT           24 01-05-95  11:13a
       3 file(s)        24 bytes
                  1455104 bytes free
```

You now have two files named ACCOUNT on the same disk, but they are in different subdirectories, and their different path names make them as different to MS-DOS as if you had given them different file names.

Time Out for a Quick Review

Before completing your multilevel file structure, review the following definitions. They summarize the terms and concepts introduced in the preceding examples.

Directory entry: A description of a file that includes the name, extension, and size of the file, and the date and time it was created or last updated.

Directory: A list of directory entries. You'll also see it used with a sense of place: "Which directory am I in?"

Root directory: The list of directory entries that MS-DOS creates and maintains on each disk. It is called the root directory (or simply the root) because the entire directory structure on the disk grows from it. Because the root has no name to MS-DOS, it is represented by a backslash (\).

Subdirectory: A file that contains directory entries. Like the term *directory,* it is also sometimes used with a sense of place: "Which subdirectory is that file in?"

Path name: The list of directory names that defines the path to a subdirectory. The directory names are separated by backslashes (\). The root directory is represented by a backslash at the beginning of the path. If a file name is included at the end of the path, it is separated from the last directory name by a backslash.

Current directory: The directory that MS-DOS assumes you want to use unless you specify another in a command. The current directory is similar in concept and effect to the current drive.

Adding More Levels to Your File Structure

The subdirectories you create can contain any type of file, including other subdirectories. Like putting dividers between other dividers in a file drawer, this further structuring narrows the subject of a storage area. Suppose you do the following kinds of work for the Marketing and Engineering departments:

Marketing	Engineering
Word processing	Word processing
Budgets	Budgets
Customer lists	Project scheduling
Sales forecasts	

You decide to set up your file structure to match your work. The following list shows the subdirectories you could create to match your computer files to the work you do (MKT and ENG are the departmental subdirectories you created):

In MKT:	In ENG:
WP	WP
BUDGET	BUDGET
CUSTOMER	SCHEDULE
SALES	

You would then have created the file structure shown in Figure 8-2.

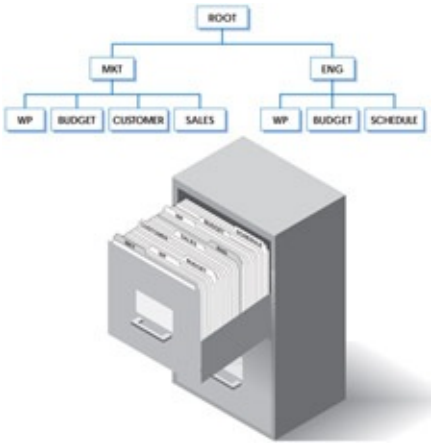


Figure 8-2: Three-level file system.

Making a Subdirectory: The Md Command

As you saw earlier in this chapter, the Make Directory (md or mkdir) command creates a

subdirectory. The Make Directory command has three parameters:

md <drive><path><directory>

<drive> is the letter, followed by a colon, of the drive (such as *a:*) that contains the disk on which the subdirectory is to be created. If you omit <drive>, MS-DOS creates the subdirectory on the disk in the current drive.

<path> is the path name of the directory in which the subdirectory is to be created. If you omit <path>, the subdirectory is created in the current directory.

<directory> is the name of the new directory.

The current directory is \MKT. For the example in this chapter, you want four subdirectories in \MKT, named WP, BUDGET, CUSTOMER, and SALES. Type the following Make Directory commands to create the subdirectories:

```
A:\MKT>md wp

A:\MKT>md budget

A:\MKT>md customer

A:\MKT>md sales
```

Display the directory of \MKT by typing this:

```
A:\MKT>dir

 Volume in drive A is EXAMPLES 2
 Volume Serial Number is 1839-10EE
 Directory of A:\MKT

.          <DIR>     01-05-95  10:54a
..         <DIR>     01-05-95  10:54a
ACCOUNT             24 01-05-95  11:13a
WP         <DIR>     01-05-95  11:22a
BUDGET     <DIR>     01-05-95  11:22a
CUSTOMER   <DIR>     01-05-95  11:23a
SALES      <DIR>     01-05-95  11:23a
       7 file(s)         24 bytes
                   1453056 bytes free
```

The directory shows the file you copied a few minutes ago (ACCOUNT) and the four subdirectories you just created.

Your file structure calls for subdirectories named WP and BUDGET in \ENG as well. Remember, MS-DOS can distinguish between \MKT\WP and \ENG\WP, and \MKT\BUDGET

and \ENG\BUDGET, because their paths are different.

To create the subdirectory \ENG\WP, type this:

```
A:\MKT>md \eng\wp
```

You included the path (\ENG) because the current directory is \MKT. The Make Directory command doesn't change the current directory, so it's still \MKT, but you can verify that the subdirectory \ENG\WP was created by displaying the contents of \ENG. Include the path here, too, by typing this:

```
A:\MKT>dir \eng

 Volume in drive A is EXAMPLES 2
 Volume Serial Number is 1839-10EE
 Directory of A:\ENG

.          <DIR>     01-05-95  10:54a
..         <DIR>     01-05-95  10:54a
ACCOUNT              24 01-05-95  11:13a
WP         <DIR>     01-05-95  11:25a
       4 file(s)         24 bytes
                   1452544 bytes free
```

Now you're going to start moving around from subdirectory to subdirectory, so before creating the last two subdirectories in \ENG, here's a closer look at your navigator, the Change Directory command.

Changing the Current Directory: The Cd Command

You have already used the Change Directory (cd or chdir) command to change and display the current directory. The Change Directory command has two parameters:

cd <drive><path>

<drive> is the letter, followed by a colon, of the drive (such as *a:*) that contains the disk on which the current directory is to be changed. If you omit <drive>, MS-DOS changes the current directory of the disk in the current drive.

<path> is the path name of the directory that is to become the current directory. If you omit <path>, MS-DOS displays the current directory on <drive>.

If you omit both <drive> and <path> (enter the command with no parameters), MS-DOS displays the name of the current directory of the disk in the current drive.

Keeping Track of Where You Are

As a note earlier in this chapter showed, you can change the system prompt to display not

only the current drive but also other information, such as the current directory.

Up to this point, the system prompt has shown the current drive and directory in abbreviated form (for example, *A:\MKT >*). In the remainder of the chapter, you'll be changing directories fairly often, so change the prompt to one that helps you identify the current directory at a glance. Type the following, including a space at the end of the line, just before you press the Enter key:

```
A:\MKT>prompt Current Directory is $p$_Command: <Enter>
```

Now the system prompt becomes:

```
Current Directory is A:\MKT
Command: _
```

You could restore the system prompt to its more familiar form by typing the Prompt command with no parameters (*prompt*) or by typing *prompt pg*, but leave it this way for the rest of the chapter. The prompt takes up a bit more space, but it helps you keep track of where you are—a useful feature, especially if you are still adjusting to the idea of moving from one directory to another.

Using the Subdirectory Markers

Remember those markers (. and ..) listed in each subdirectory? They're designed to let you move quickly up and down a directory structure, particularly when several levels make the path names long.

The .. marker represents the directory that contains the current directory (sometimes called the *parent* of the current directory). The current directory is \MKT; to move the current directory up one level (toward the root directory), type this:

```
Current Directory is A:\MKT
Command: cd ..

Current Directory is A:\
Command: _
```

The system prompt still shows you the current directory, but, as you can see, the current directory has changed to the root directory, which is one level above \MKT.

To complete your file structure, you need two more subdirectories in \ENG. First change the current directory to \ENG, and then create \ENG\BUDGET and \ENG\SCHEDULE by typing the following:

```
Current Directory is A:\
Command: cd eng

Current Directory is A:\ENG
Command: md budget
```

```
Current Directory is A:\ENG
Command: md schedule
```

This completes the structure of your multilevel file system.

You have nine subdirectories, plus the root directory, any of which you can use as if it were a separate disk. To show you how easy this is, the next few examples have you put sample files in several of the subdirectories. Figure 8-3 shows how your final file system will look, including the path names of all directories (above the boxes) and the names of the files you'll add (inside the shaded boxes).

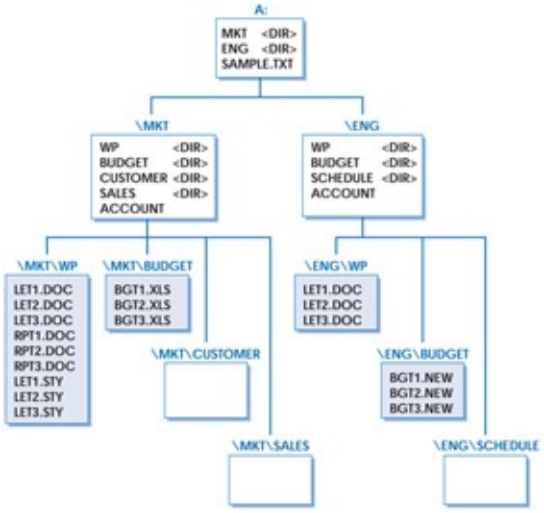


Figure 8-3: Two-department file structure.

To create the sample files in \ENG\WP, first change the current directory to \ENG\WP and copy the file named ACCOUNT from \ENG, naming it LET1.DOC, by typing the following:

```
Current Directory is A:\ENG
Command: cd wp

Current Directory is A:\ENG\WP
Command: copy \eng\account let1.doc
        1 file(s) copied
```

Notice that you have to use \ENG in the command, even though \ENG\WP is a subdirectory of \ENG.

Now copy LET1.DOC twice, to create LET2.DOC and LET3.DOC, and display the directory by typing this:

```
Current Directory is A:\ENG\WP
Command: copy let1.doc let2.doc
        1 file(s) copied
```

```
Current Directory is A:\ENG\WP
Command: copy let1.doc let3.doc
      1 file(s) copied

Current Directory is A:\ENG\WP
Command: dir

 Volume in drive A is EXAMPLES 2
 Volume Serial Number is 1839-10EE
 Directory of A:\ENG\WP

.          <DIR>    01-05-95  11:25a
..         <DIR>    01-05-95  11:25a
LET1    DOC       24 01-05-95  11:13a
LET2    DOC       24 01-05-95  11:13a
LET3    DOC       24 01-05-95  11:13a
      5 file(s)        72 bytes
                 1449984 bytes free
```

From this subdirectory, you can copy all three of these files to \MKT\WP with one command. Type the following:

```
Current Directory is A:\ENG\WP
Command: copy *.* \mkt\wp
```

MS-DOS lists the source files as it copies them:

```
LET1.DOC
LET2.DOC
LET3.DOC
      3 file(s) copied
```

These files could be word processing files that contain letters. Now create three more files in \MKT\WP that could represent word processing files that contain reports. First, change the directory to \MKT\WP, then copy the three files whose names begin with LET, changing their names so they begin with RPT. Type this:

```
Current Directory is A:\ENG\WP
Command: cd \mkt\wp

Current Directory is A:\MKT\WP
Command: copy let?.doc rpt?.doc
```

MS-DOS again lists the source files as it makes the copies:

```
LET1.DOC
LET2.DOC
LET3.DOC
      3 file(s) copied
```

To complete the files in this subdirectory, copy the same three files again, this time changing their extension to STY, which could identify word processing files that contain style sheets for formatting and printing documents. Type the following:

```
Current Directory is A:\MKT\WP
Command: copy let?.doc let?.sty
```

Now display the directory to verify that all nine files are there by typing this:

```
Current Directory is A:\MKT\WP
Command: dir

 Volume in drive A is EXAMPLES 2
 Volume Serial Number is 1839-10EE
 Directory of A:\MKT\WP

.          <DIR>    01-05-95  11:22a
..         <DIR>    01-05-95  11:22a
LET1    DOC       24 01-05-95  11:13a
LET2    DOC       24 01-05-95  11:13a
LET3    DOC       24 01-05-95  11:13a
RPT1    DOC       24 01-05-95  11:13a
RPT2    DOC       24 01-05-95  11:13a
RPT3    DOC       24 01-05-95  11:13a
LET1    STY       24 01-05-95  11:13a
LET2    STY       24 01-05-95  11:13a
LET3    STY       24 01-05-95  11:13a
      11 file(s)        216 bytes
                    1445376 bytes free
```

To complete the file system, you'll need three files in \MKT\BUDGET, named BGT1.XLS, BGT2.XLS, and BGT3.XLS, and you'll need three files in \ENG\BUDGET, named BGT1.NEW, BGT2.NEW, and BGT3.NEW. First, use the Copy command to create the files in \MKT\BUDGET by typing this:

```
Current Directory is A:\MKT\WP
Command: copy  let?.doc \mkt\budget\bgt?.xls
```

MS-DOS lists three source files as it makes the copies and then displays the prompt.

To finish, you could create the files for \ENG\BUDGET from the current directory, but that would mean typing the full path and file name. To save some keystrokes, change to the \MKT\BUDGET directory by typing this:

```
Current Directory is A:\MKT\WP
Command: cd ..\budget
```

The .. marker tells MS-DOS to move up to the parent (\MKT) of the current directory and,

from there, to move down to the \BUDGET directory. The prompt changes to the following:

```
Current Directory is A:\MKT\BUDGET
Command: _
```

Now you can finish creating the sample files by typing this:

```
Current Directory is A:\MKT\BUDGET
Command: copy  *.xls \eng\budget\*.new
```

MS-DOS lists three source files copied: BGT1.XLS, BGT2.XLS, and BGT3.XLS. Your file system now has the directories and files shown in Figure 8-3.

Moving Files from One Directory to Another

Like the Copy command, the Move command (available in versions 6.0 and later) makes a copy of one or more files on a different directory or disk; unlike Copy, Move doesn't leave the original in place. You can also use Move to change the name of a directory.

The Move command requires two parameters:

move /Y <source> <target>

/Y tells Move not to prompt you before creating the directory into which the specified files are moved.

<source> is the name of the file to be moved, and <target> is the name for the file in its new location. You can specify a drive letter and path name with both <source> and <target>. You can use wildcard characters to move a set of files, but if you do so you cannot change the files' names; if you want to give a file a different name in its new location, you can move only one file at a time.

If you are using version 6.2 or later and <target> is the name of an existing file, MS-DOS prompts you to confirm that you want to replace the existing file with <source>, unless you've included the /Y parameter, which turns off the prompt.

If you make <source> the name of a directory, you can rename the directory by specifying a different name in <target>. You cannot, however, move a directory to a different location in the directory structure.

Examples of Moving Files

The next three examples show you how to use the Move command and let you use the dot (.) and double-dot (..) markers listed in the subdirectories. First move the file BGT1.XLS from \MKT\BUDGET to \MKT\SALES by typing this:

```
Current Directory is A:\MKT\BUDGET
Command: move bgt1.xls \mkt\sales
```

MS-DOS responds by displaying the source and target file names:

```
a:\mkt\budget\bgt1.xls => a:\mkt\sales\bgt1.xls [ok]
```

Use the Directory command to verify that BGT1.XLS now has been moved to the new directory.

You can use wildcard characters to move more than one file with similar file names or extensions. Notice that in this example, instead of typing the full path name of the target, you'll use the double-dot representation for the directory that contains the current directory. Type the following Move command to move both BGT2.XLS and BGT3.XLS to \MKT\SALES:

```
Current Directory is A:\MKT\BUDGET
Command: move bgt?.xls ..\sales
a:\mkt\budget\bgt2.xls => a:\mkt\sales\bgt2.xls [ok]
a:\mkt\budget\bgt3.xls => a:\mkt\sales\bgt3.xls [ok]
```

The Move command lists only BGT2.XLS and BGT3.XLS because you already moved BGT1.XLS with the first Move command.

Now suppose you tell MS-DOS (version 6.2 or later) to move BGT1.XLS to BGT2.XLS. Type the following:

```
move ..\sales\bgt1.xls ..\sales\bgt2.xls
```

MS-DOS responds by prompting you to confirm that you want to replace (overwrite) BGT2.XLS with BGT1.XLS:

```
Overwrite a:\mkt\sales\bgt2.xls (Yes/No/All)?_
```

Yes means yes and No means no, but what does All mean? Well, if you had used a wildcard character to move a series of files with one command and knew that you wanted to overwrite an existing file in every instance, you could type *A* for All; if you type *Y,* MS-DOS prompts you for each instance in which a file would be overwritten.

Type *N;* MS-DOS cancels the Move command and displays the system prompt.

The last Move command moves all three files back to the current directory. This time you'll use the single dot to represent the current directory instead of typing its complete path name:

```
Current Directory is A:\MKT\BUDGET
Command: move ..\sales\*.xls .
a:\mkt\sales\bgt1.xls => a:\mkt\budget\bgt1.xls [ok]
a:\mkt\sales\bgt2.xls => a:\mkt\budget\bgt2.xls [ok]
a:\mkt\sales\bgt3.xls => a:\mkt\budget\bgt3.xls [ok]
```

Example of Renaming a Directory

Suppose you decide that a better name for the directory A:\ENG\WP is A:\ENG\EDIT. To change the name of the directory, type this:

```
Current Directory is A:\MKT\BUDGET
Command: move a:\eng\wp a:\eng\edit
```

MS-DOS responds by displaying the old and new names of the directory:

```
a:\eng\wp => a:\eng\edit [ok]
```

Now when you use the Directory command to view the contents of the ENG directory, you'll see on the following page that the name has changed.

```
Current Directory is A:\MKT\BUDGET
Command: dir a:\eng

 Volume in drive A is EXAMPLES 2
 Volume Serial Number is 1839-10EE
 Directory of A:\ENG

.          <DIR>     01-05-95  10:54a
..         <DIR>     01-05-95  10:54a
ACCOUNT           24  01-05-95  11:13a
EDIT       <DIR>     01-05-95  11:30a
BUDGET     <DIR>     01-05-95  11:30a
SCHEDULE   <DIR>     01-05-95  11:30a
```


PREVIOUS NEXT

Managing Your Subdirectories

Once subdirectories become part of your work with MS-DOS, they quickly become essential for organizing your programs and data files, especially on a hard disk. But the more subdirectories and files you create, the harder it is to keep track of where they are and what they contain. Giving descriptive names to your subdirectories and files is one way to maintain order. Another way is to group your files—especially data files—logically. At times, however, you'll still find yourself wondering what an old subdirectory contains and whether you still need it. And even if you're tremendously well organized, you'll also sometimes forget just where you saved *that* file (usually when you're in a desperate hurry to find it).

As you would expect, MS-DOS includes several commands that help you manage your subdirectories and the files they contain. You've already seen the Directory and Change Directory commands, which allow you to move around through your directory structure. The remainder of this chapter describes other commands that help you:

- Eliminate unneeded subdirectories (the Remove Directory command)
- Delete an entire directory structure, files and all (the Deltree command)
- Use command files from any subdirectory (the Path command)
- Search for particular files by name (the /S parameter of the Directory command), or attribute (the /A parameter of the Directory command)
- View the directory structure of a disk (the Tree and Check Disk commands)
- Help MS-DOS find data files in different subdirectories (the Append command)

Removing a Subdirectory: The Rd Command

As you work with a multilevel filing system, you might find that you no longer need a particular subdirectory, or that you want to combine the files from several subdirectories into one and then delete the unneeded subdirectories from your file structure. The Remove Directory (rd or rmdir) command removes a subdirectory. A subdirectory cannot be removed with this command if it contains any files or any other subdirectories.

The Remove Directory command has two parameters:

rd <drive><path>

<drive> is the letter, followed by a colon, of the drive that contains the disk with the subdirectory to be removed. You can omit <drive> if the subdirectory is on the disk in the current drive.

<path> is the path name of the subdirectory to be removed. You must specify <path> because MS-DOS will not remove the current directory.

Suppose you decide you don't need the subdirectory \ENG\EDIT. Tell MS-DOS to remove it by typing this:

```
Current Directory is A:\MKT\BUDGET
Command: rd \eng\edit
```

MS-DOS responds *Invalid path, not directory, or directory not empty* because \ENG\EDIT isn't empty: You previously put three files—LET1.DOC, LET2.DOC, and LET3.DOC—in it.

This example points out the difference in the ways you handle files and subdirectories. As you saw in earlier chapters, you use the Delete command to delete a file from a disk. To remove a directory, however, you use the Remove Directory command. Version 6 of MS-DOS offers the Delete Tree command for deleting directories and anything contained in them. See the next section for more information.

In the next example, you will delete three files with the Delete command and then remove a directory with the Remove Directory command. First, change the current directory to \ENG, and then delete the files with the Delete command by typing this:

```
Current Directory is A:\MKT\WP
Command: cd \eng

Current Directory is A:\ENG
Command: del edit\*.doc
```

You changed the current directory to \ENG, rather than to \ENG\EDIT, because MS-DOS won't remove the current directory. Now that \ENG\EDIT is empty, type the Remove Directory command and verify the change by displaying the directory:

```
Current Directory is A:\ENG
Command: rd edit

Current Directory is A:\ENG
Command: dir

 Volume in drive A is EXAMPLES 2
 Volume Serial Number is 1839-10EE
 Directory of A:\ENG
.           <DIR>     01-05-95  10:54a
..          <DIR>     01-05-95  10:54a
ACCOUNT              24 01-05-95  11:13a
BUDGET      <DIR>     01-05-95  11:30a
SCHEDULE    <DIR>     01-05-95  11:30a
       5 file(s)         24 bytes
                   1444352 bytes free
```

The subdirectory \ENG\EDIT is gone.

If you want to remove a directory but need some of the files it contains, copy the files you need to another subdirectory, then erase all the files and remove the unneeded directory.

Note The next few topics cover features in version 5 or later of MS-DOS. If you don't have version 5 or later, skip to the heading "The Path to a Command: The Path Command."

Deleting a Subdirectory and Its Contents: Deltree

The Remove Directory command is perfectly adequate for getting rid of directories, but deleting several directories can get tedious, especially if some of the directories contain subdirectories and all of the directories and subdirectories contain files. Starting with version 6.0, MS-DOS gives you a way to cut through all the delays and rid yourself of all those directories in one fell swoop: the Deltree command. Deltree deletes a directory and *all* its contents, even if those contents include subdirectories with their own files and subdirectories. It deletes an entire branch, or subtree, from the directory tree structure. Because of Deltree's potential for damage, MS-DOS prompts you to confirm the action just as it does when you tell it to delete all the files in a directory. If you told MS-DOS to delete a directory named SALES, for example, Deltree would respond *Delete directory "sales" and all its subdirectories? [yn].* If you reply by typing *N* (uppercase or lowercase), MS-DOS cancels the Deltree command.

Suppose you wanted to delete all the directories you created for the examples in this chapter. Using the Remove Directory command, you'd have to delete the directories one at a time, starting with the lowest in the directory structure, deleting the files with the Delete command before removing the directories with the Remove Directory (rd) command. Using Deltree, you could get rid of all the directories and files with just one command.

But there's a price to be paid for all this convenience: the danger that you'll inadvertently wipe out dozens, hundreds, even all of your files. Suppose that your root directory contained a directory named RUNBAS that contained several subdirectories and files—like the file structure you created for the examples in this chapter. You're ready to delete these directories and files, and you decide to save time by using the Deltree command instead of deleting the files and directories using several Delete and Remove Directory commands. You're tired, or rushed, or not paying attention, and you type *deltree* \ but forget to type *runbas.* If you were to respond affirmatively to the prompt, you'd erase every single file and directory on your disk.

So use Deltree with care. Make certain that you understand what you're telling MS-DOS to do before you use the command, and then go over it one more time before you press Y and Enter.

Viewing Files in More Than One Subdirectory

If you have version 5 or later, display the contents of the current subdirectory and all subdirectories below it by using the /S parameter of the Directory command. When used to

display the contents of more than one subdirectory, the command is this:

dir <filename> /S

If you include <filename>, MS-DOS searches for the file in the current directory and all subdirectories below it. You can use wildcards to specify a group of files. If you don't include <filename>, MS-DOS displays the names of all files in the current directory and all its subdirectories.

You can also combine /S with other parameters of the Directory command. For example, you can include /P to halt the display after each screenful, and you can include the /W parameter to display the directories in wide format.

To see how the /S parameter works, begin by changing to the root directory of the disk in drive A to start at the highest level of your directory tree. Type the following Change Directory command (remember, \ means the root directory):

```
Current Directory is A:\ENG
Command: cd \
```

Now request a listing of all subdirectories and files stored in the root directory. The list is long, so use the /P parameter as well as /S so that MS-DOS will pause after each screenful. Type this:

```
Current Directory is A:\
Command: dir /s /p
```

The first screenful looks like the following:

```
Volume in drive A is EXAMPLES 2
 Volume Serial Number is 1839-10EE

 Directory of A:\

MKT         <DIR>     01-05-95  10:54a
ENG         <DIR>     01-05-95  10:54a
SAMPLE   TXT        24 01-05-95  11:13a
       3 file(s)         24 bytes
 Directory of A:\ENG

.           <DIR>     01-05-95  10:54a
..          <DIR>     01-05-95  10:54a
ACCOUNT              24 01-05-95  11:13a
BUDGET      <DIR>     01-05-95  11:30a
SCHEDULE    <DIR>     01-05-95  11:30a
       5 file(s)         24 bytes

Directory of A:\ENG\BUDGET
```

```
.          <DIR>     01-05-95  11:30a
..         <DIR>     01-05-95  11:30a
Press any key to continue . . .
```

Notice that MS-DOS displays the name of each subdirectory on a separate line above the list of subdirectory entries. If you want to display another screenful, press any key. If you want to cancel, press Ctrl-Break.

Viewing Specified Files in More Than One Subdirectory

A listing of every file in every subdirectory on a disk can be very long. You can limit the display by telling MS-DOS to show only the names of files that you specify. For example, use the asterisk wildcard to tell MS-DOS to list any file in any subdirectory whose extension is DOC. Type this:

```
Current Directory is A:\
Command: dir *.doc /s
```

MS-DOS responds:

```
Volume in drive A is EXAMPLES 2
 Volume Serial Number is 1839-10EE

Directory of A:\MKT\WP

LET1     DOC        24 01-05-95  11:13a
LET2     DOC        24 01-05-95  11:13a
LET3     DOC        24 01-05-95  11:13a
RPT1     DOC        24 01-05-95  11:13a
RPT2     DOC        24 01-05-95  11:13a
RPT3     DOC        24 01-05-95  11:13a
       6 file(s)        144 bytes

Total files listed:
       6 file(s)        144 bytes
                  1444352 bytes free
```

Only the subdirectory \MKT\WP contains files with the extension DOC.

The /S parameter can be particularly useful if you remember the name of a file but you can't remember where you put it. Change to the root directory of the disk, and use the /S parameter to search every subdirectory. Use wildcards, if needed, to search for a group of files or for a file whose name you're not sure of.

The Path to a Command: The Path Command

In a multilevel filing system, you'll probably change the current directory as you use data files in different subdirectories. But you'll use command files too, such as the external MS-DOS commands and application programs. When you type a command, MS-DOS looks for the command file in the current directory. If you've changed directories, chances are the current directory won't contain the command file you need.

The Path command lets you tell MS-DOS where to look for a command file if it's not in the current directory. You can name one or more directories—the root directory or any subdirectory on the disk in any disk drive. This command lets you work in any subdirectory you want and still be able to use any command file.

The Path command has three parameters:

path <drive><path> ;

<drive> is the letter, followed by a colon, of the drive (such as *a:*) with the disk that contains the command file. If you omit <drive>, MS-DOS looks on the disk in the current drive.

<path> is the path name of the directory that contains the command file.

You can specify several command paths in one command, separating them with semicolons. If you enter a Path command with no parameters (just type *path*), MS-DOS displays the command paths you have defined. If you type *path* followed only by a semicolon, MS-DOS cancels any command paths you have defined.

If you have a hard disk, the root directory probably contains a special file named AUTOEXEC.BAT that MS-DOS reads each time you start or restart your computer. AUTOEXEC.BAT contains instructions that help MS-DOS work well for you. One of those instructions is probably a Path command that enables you to use application programs, as well as external MS-DOS commands, no matter what the current directory is. For example, suppose you use Microsoft Word, Lotus 1-2-3, R:BASE, and MaxThink, and your MS-DOS files are in a directory named \DOS. Your Path command probably looks something like this:

```
path  c:\dos;c:\word;c:\123;c:\rbfiles;c:\max
```

The order in which the directory names are specified determines the order in which MS-DOS searches for the command file. Because the drive letter and root-directory symbol (\) precede each subdirectory name, MS-DOS knows where to find each program, and you can use any program in any of these directories even if the current drive isn't C.

If MS-DOS has no trouble starting application programs or carrying out external commands when drive A is the current drive, you can be certain you have an AUTOEXEC.BAT file with a Path command in it. You can check on your own command path by typing *path* with no parameters. Type this:

```
Current Directory is A:\
Command: path
```

If you do have any problems starting programs from another drive, consult Chapter 14,

"Creating Your Own Commands," which describes how you can create or modify an AUTOEXEC.BAT file.

Displaying the Directory Structure: Tree and Check Disk

As described in Chapter 11, "The MS-DOS Shell," the Shell lets you quickly display a diagram of the directory structure of any disk and use that diagram to move from one directory to another. Tree and Check Disk are commands that allow you to view your directory structure from the system prompt.

The Tree command has two main parameters:

tree <drive> /F

<drive> is the letter of the drive, followed by a colon, that contains the disk whose directory structure is to be displayed.

/F displays a list of the files in each directory.

Checking the Directory Structure of a Disk

Note The next two examples use the Tree command. If your version of MS-DOS does not include this command, a later example shows you how to use the Check Disk command to see a list of directories and files on a disk.

Suppose you're about to create a new file and you want to check on the most appropriate subdirectory for it. As its name implies, the Tree command shows you the structure of the directory tree on whichever disk you specify. For example, check the directories on your sample disk. To be sure you see the entire structure, first check that the current directory is the root directory. If it isn't, change directories by typing *cd* \. Now ask MS-DOS to show you the names and relative levels of the directories on the disk. Type this:

```
Current Directory is A:\
Command: tree
```

If you're using version 4 or later of MS-DOS, you see a report like this:

```
Directory PATH listing for Volume EXAMPLES 2
Volume Serial Number is 1839-10EE

A:.
├─MKT
│  ├─WP
│  ├─BUDGET
│  ├─CUSTOMER
│  └─SALES
└─ENG
   ├─BUDGET
   └─SCHEDULE
```

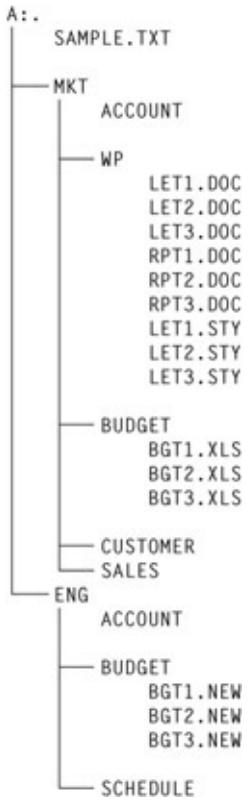
If you're using an earlier version of MS-DOS, you see a report like the following. (Several

line spaces that appear on the screen have been removed to condense the display.)

```
DIRECTORY PATH LISTING
Path: \MKT
Sub-directories:  WP
                  BUDGET
                  CUSTOMER
                  SALES
Path: \MKT\WP
Sub-directories:  None
Path: \MKT\BUDGET
Sub-directories:  None
Path: \MKT\CUSTOMER
Sub-directories:  None
Path: \MKT\SALES
Sub-directories:  None
Path: \ENG
Sub-directories:  BUDGET
                  SCHEDULE
Path: \ENG\BUDGET
Sub-directories:  None
Path: \ENG\SCHEDULE
Sub-directories: None
```

But suppose a report on the directory structure isn't detailed enough. You want to know the name of each file in each subdirectory—perhaps because there is one file in particular you need. To do this, you use the /F parameter, which shows not only the directory structure of the disk but also the names of the files in each directory. For example, versions 4 and later produce a report like this when you type *tree /f*.

```
Directory PATH listing for Volume EXAMPLES 2
Volume Serial Number is 1839-10EE
```



Now the tree shows the names of your files beneath the names of the subdirectories that contain them.

If your version of the Tree command doesn't produce the graphic type of report shown above, or if your version of MS-DOS doesn't include a Tree command, you can use the /V parameter of the Check Disk command to see a list of files and directories. (Note that you should not use Check Disk if Microsoft Windows is running.) Type this:

```
Current Directory is A:\
Command: chkdsk /v
```

MS-DOS responds with a report like this:

```
Volume Serial Number is 1839-10EE
Directory A:\
Directory A:\MKT
A:\MKT\ACCOUNT
Directory A:\MKT\WP
A:\MKT\WP\LET1.DOC
A:\MKT\WP\LET2.DOC
A:\MKT\WP\LET3.DOC
A:\MKT\WP\RPT1.DOC
A:\MKT\WP\RPT2.DOC
A:\MKT\WP\RPT3.DOC
A:\MKT\WP\LET1.STY
A:\MKT\WP\LET2.STY
A:\MKT\WP\LET3.STY
Directory  A:\MKT\BUDGET
A:\MKT\BUDGET\BGT1.XLS
```

```
A:\MKT\BUDGET\BGT2.XLS
A:\MKT\BUDGET\BGT3.XLS
Directory  A:\MKT\CUSTOMER
Directory  A:\MKT\SALES
Directory  A:\ENG
A:\ENG\ACCOUNT
Directory  A:\ENG\BUDGET
A:\ENG\BUDGET\BGT1.NEW
A:\ENG\BUDGET\BGT2.NEW
A:\ENG\BUDGET\BGT3.NEW
Directory A:\ENG\SCHEDULE
A:\SAMPLE.TXT

   1457664 bytes total disk space
     4096 bytes in 8 directories
     9216 bytes in 18 user files
   1444352 bytes available on disk

      512 bytes in each allocation unit
     2847 total allocation units on disk
     2821 available allocation units on disk

   655360 total bytes memory
   528688 bytes free
```

Here, the directories are identified by *Directory* preceding the path and file name. The file in the root directory (SAMPLE.TXT) is listed at the end, above the usual Check Disk reports on disk space and memory.

A printed copy of either of these reports can be helpful, especially if your filing system has several levels. For a hard disk with several hundred files, however, bear in mind that either a Tree or a Check Disk report showing both directories and files could be several pages long. Here are some ways you can print the reports of either the Tree or the Check Disk command, or both.

If you want to print a list of both directories and files, use the Tree command with the /F parameter or the Check Disk command with the /V parameter. If you have version 4 or later and your printer is not an IBM or IBM-compatible printer, either use the Check Disk command or use the /A parameter of the Tree command, which tells MS-DOS to use hyphens, backslashes, and other common characters to print the diagram. Without the /A parameter, your printout might include accented letters, small open circles, or other such characters in place of the horizontal and vertical lines showing the branches of your directory tree.

For example, to print a complete report for the disk in drive A, type this:

```
Current Directory is A:\
Command: tree /f > prn
```

or:

```
Current Directory is A:\
Command: tree /f /a > prn
```

or:

```
Current Directory is A:\
Command: chkdsk /v > prn
```

To print only a list of directories on the disk in drive A, you can use either of the following commands.

With the Tree command, type this:

```
Current Directory is A:\
Command: tree > prn
```

With the Check Disk command, you use techniques described in Chapter 13, "Taking Control of Your System," to filter out all the file names from the output of the Check Disk command. Filtering leaves just the subdirectory names, which are then sent to the printer. Try it. (Notice that a slash, not a backslash, precedes the *v.*) Type this:

```
Current Directory is A:\
Command: chkdsk /v œ find "Di" > prn
```

The significant printed output of this command shows only the names of the directories (the lines that contain *Di*).

Any of these guides to your file system can help you keep track of both files and subdirectories and can help you use your system more efficiently.

Another Type of Path: The Append Command

If you're using 3.3 or a later version of MS-DOS, you can use the Append command as well as the Path command to tell MS-DOS where to look for a file if it's not in the current directory. Although it seems redundant to have two commands that can set a search path, there's a significant difference between Path and Append: Path sets the path to *command* files; Append sets the path to *data* files too. Just as with the Path command, you can name one or more directories on any disk drive.

The Append command has three main parameters related to helping MS-DOS find data files (and program files):

```
append <drive><path> ;
```

<drive> is the letter, followed by a colon, of the drive (such as *c:*) with the disk that contains

the files. If you omit <drive>, MS-DOS looks on the current drive.

<path> is the path name of the directory that contains the files.

You can specify several appended paths in one command, separating them with semicolons. If you enter an Append command with no parameters (just type *append*), MS-DOS displays the paths you have defined. If you type *append* followed only by a semicolon, MS-DOS cancels any paths you have defined.

It's easy to try the Append command. The root directory is the current directory, and the file RPT1.DOC is in the directory \MKT\WP. First, try to display the file by using the Type command:

```
Current Directory is A:\
Command: type rpt1.doc
```

MS-DOS replies *File not found* because RPT1.DOC isn't in the current directory. Now type two Append commands to set the data path to \MKT\WP and display the data path:

```
Current Directory is A:\
Command: append \mkt\wp

Current Directory is A:\
Command: append
APPEND=\MKT\WP
```

Now try the Type command again:

```
Current Directory is A:\
Command: type rpt1.doc
This is a sample file.
```

This time MS-DOS finds the file because you told it where to look. Cancel the appended path by typing *append* with just a semicolon, then check again:

```
Current Directory is A:\
Command: append ;

Current Directory is A:\
Command: append
No Append
```

The Path and Append commands are valuable tools for improving the efficiency and convenience of using a hard disk. Appendix C, "MS-DOS Command Reference," includes a complete description of each.

If you're going to continue using your system, restart your computer or type the following to restore the system prompt:

```
Current Directory is A:\
```

```
Command: prompt $p$g

A:\>
```

Note Be sure to save the examples disk; you'll use it again in Chapter 9.

PREVIOUS NEXT

Chapter Summary

Although this chapter introduced several new terms and concepts, it doesn't take many commands to set up a multilevel file system. The structure shown in Figure 8-3, for example, required only the 11 commands listed here. (Don't enter them; this list is just to show you the commands you entered.)

```
md mkt          md sales
md eng          md \eng\wp
cd mkt          cd \eng
md wp           md budget
md budget       md schedule
md customer
```

You might not create a file structure with this many levels on a floppy disk, although it's certainly possible if you use high-density floppy disks. As you noticed, subdirectories require a great deal of work from the floppy disk drive. But you might find that two or three subdirectories reduce the number of floppy disks you use, or that they let you use your system more efficiently. The examples in this chapter showed you how to use all the commands you need to create and manage a multilevel filing system.

Save the floppy disk that contains the file system you created in this chapter. You'll use it again to copy sample files in the next chapter, "Managing Your Hard Disk."

PREVIOUS NEXT

Chapter 9: Managing Your Hard Disk

Overview

A hard disk holds far more data than a floppy disk, and MS-DOS can use it much more quickly. You don't remove a hard disk; it's permanently fixed in the drive. Like a floppy disk, a hard disk stores data in tracks and sectors; unlike a floppy disk, however, a hard disk stores data on magnetically coated, rigid metal—not plastic—disks that are enclosed in a nonremovable case.

A typical hard disk drive contains two or more separate disks that give it a total storage capacity of 250 to 500 megabytes, but hard disks with even higher capacities are becoming common. Several computers offer a hard disk that can hold 1024 MB or more. This chapter focuses on how to use your hard disk efficiently.

Managing your hard disk requires more thought and planning than managing your floppy disks, simply because of the sheer number of files involved. But properly organized and managed, your hard disk is a fast and effective tool you use with a minimum of fuss. Two tasks are more important than any others in managing your hard disk efficiently. One is setting up a filing system that lets you take advantage of the disk's capacity without losing track of all your files; this chapter shows you how to plan a structure and use the commands to create that structure. The other task is backing up the files periodically, both to protect your data in the event your hard disk is inadvertently erased or damaged and to clear out old files you no longer use regularly; the topic of backing up is covered in the next chapter, "Protecting Your Disks and Files."

The previous chapter covered setting up and using a multilevel filing system. Everything you learned there applies to managing your hard disk properly. This chapter covers the additional tasks of organizing directories and files on your hard disk and managing large numbers of files.

In most ways, you treat a hard disk as if it were a large floppy disk, using the MS-DOS directory commands to create, change, and remove directories, and the MS-DOS file commands to copy, erase, rename, and otherwise work with your files. The Volume and Label commands described in Chapter 6, "Managing Your Floppy Disks," can also be used with a hard disk. But two commands—Diskcopy and Diskcomp—don't work with the hard disk because they are designed to work only with entire floppy disks.

This chapter suggests some guidelines to simplify the job of managing your hard disk; it shows you how to set up a filing system, control the archive attribute of a file with the Attribute command, and copy files selectively with the Replace and Xcopy commands. If you are running MS-DOS version 6, you can increase the capacity and efficiency of your hard disk with the DoubleSpace (MS-DOS version 6.0 and 6.2) or DriveSpace (MS-DOS version 6.22) and Defrag programs, and, if you have version 6.2 or later, use a program called ScanDisk to check for and correct disk errors. See the section later in this chapter named "Checking a Disk for Errors" for more information about ScanDisk.

Putting Application Programs on Your Hard Disk

When you set up your directory structure, you'll have to copy the command and data files of your application programs onto the hard disk. Most application programs install themselves in their own subdirectories on a hard disk; others don't include an installation (or setup) program, leaving it to you to create the subdirectories and copy the files. You shouldn't put all your programs in the root directory or in a single large program directory. If you do, you'll have difficulty telling one program file from another, and you'll end up with as much disorganization as you would if you tossed all your program floppy disks into one big drawer.

Remember, too, that MS-DOS itself is a collection of command (program) files. Like application-program files, the MS-DOS files can be stored in a variety of places on your hard disk, including the root directory. Versions 4 and later, in fact, install themselves in a \DOS subdirectory. If, for some reason, your MS-DOS files are in the root directory, you should move them to a subdirectory named \DOS so they'll be easy to find and won't clutter up either your root directory or your directory displays. Instructions for doing this are in Appendix A, "Installing MS-DOS."

PREVIOUS NEXT

Preparing for the Examples

The examples in this chapter rely on the information presented in the preceding chapter, "A Tree of Files," and use the directory structure and files you created there. If you haven't completed those examples, do so before continuing.

The examples in this chapter require the sample subdirectories and files you see in Figure 9-1 on the next page. You will create these on the hard disk in a subdirectory named \RUNDOS, which will keep all your examples in one place for easy removal.

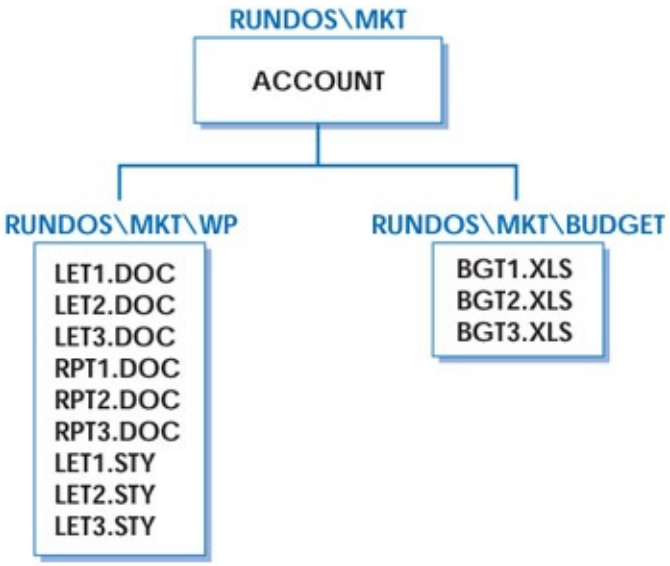


Figure 9-1: Subdirectories and files for hard disk examples.

If your system prompt doesn't show the current directory, change it to an abbreviated version of the *Current Directory is...* prompt you used in the last chapter. Type the following:

```
C>prompt $p$g<Enter>

C:\>
```

Now your system prompt shows both the current drive and the current directory and will help you keep track of where you are in your directory structure.

To create the \RUNDOS subdirectory, type this:

```
C:\>md \rundos
```

Now change the current directory to \RUNDOS by typing this:

```
C:\>cd \rundos
```

Copying Files to Your Hard Disk

The remainder of your preparation depends on which version of MS-DOS you're using. If you have version 3.2 or later, follow the instructions under the heading "Copying Files with the Xcopy Command." If you have a version of MS-DOS prior to 3.2, follow the instructions

under the heading "Copying Files with the Copy Command."

Copying Files with the Xcopy Command

If you're using version 3.2 or later of MS-DOS, copying everything from your practice floppy disk to the \RUNDOS directory is simple. All you need is one Xcopy command, which copies subdirectories and files from the source disk to the target disk. The Xcopy parameter you use here, /S, copies all subdirectories as well as all files. Xcopy is described in more detail later. For now, put the floppy disk that contains your sample files in drive A and type this:

```
C:\RUNDOS>xcopy a: /s
```

MS-DOS responds:

```
Reading source file(s)...
A:SAMPLE.TXT
A:MKT\ACCOUNT
A:MKT\WP\LET1.DOC
A:MKT\WP\LET2.DOC
A:MKT\WP\LET3.DOC
A:MKT\WP\RPT1.DOC
A:MKT\WP\RPT2.DOC
A:MKT\WP\RPT3.DOC
A:MKT\WP\LET1.STY
A:MKT\WP\LET2.STY
A:MKT\WP\LET3.STY
A:MKT\BUDGET\BGT1.XLS
A:MKT\BUDGET\BGT2.XLS
A:MKT\BUDGET\BGT3.XLS
A:ENG\ACCOUNT
A:ENG\BUDGET\BGT1.NEW
A:ENG\BUDGET\BGT2.NEW
A:ENG\BUDGET\BGT3.NEW
      18 File(s) copied
```

You don't need the file named SAMPLE.TXT, the directories named \ENG and \ENG\BUDGET, or the files they contain. If you have version 6 of MS-DOS, remove them by typing this:

```
C:\RUNDOS>del  sample.txt


C:\RUNDOS>deltree  eng
Delete directory "eng" and all its subdirectories? [yn] y
Deleting eng...
```

If you have an earlier version of MS-DOS, type this:

```
C:\RUNDOS>del  sample.txt
C:\RUNDOS>del  eng\account
```

```
C:\RUNDOS>del  eng\budget\*.new
C:\RUNDOS>rd  eng\budget
C:\RUNDOS>rd  eng
```

You're through.

Copying Files with the Copy Command

If you're using a version of MS-DOS prior to 3.2, put the floppy disk that contains your sample files in drive A and type the following commands to duplicate a part of the floppy disk's directory structure on your hard disk:

```
C:\RUNDOS>md  mkt
C:\RUNDOS>md  mkt\wp
C:\RUNDOS>md  mkt\budget
```

Now type the following to copy the files from the floppy disk to the hard disk:

```
C:\RUNDOS>copy a:\mkt\account mkt
        1 File(s) copied

C:\RUNDOS>copy a:\mkt\wp\*.* mkt\wp
A:\MKT\WP\LET1.DOC
A:\MKT\WP\LET2.DOC
A:\MKT\WP\LET3.DOC
A:\MKT\WP\RPT1.DOC
A:\MKT\WP\RPT2.DOC
A:\MKT\WP\RPT3.DOC
A:\MKT\WP\LET1.STY
A:\MKT\WP\LET2.STY
A:\MKT\WP\LET3.STY
       9 File(s) copied

C\RUNDOS>copy a:\mkt\budget\*.* mkt\budget
A:\MKT\BUDGET\BGT1.XLS
A:\MKT\BUDGET\BGT2.XLS
A:\MKT\BUDGET\BGT3.XLS
       3 File(s) copied
```

This completes the preparation.

Changing the Attributes of a File or a Directory

The Attribute command lets you control the following attributes of a file or directory: *read-only*, *hidden*, *archive*, and *system*. You can use the Attribute command to control access to a file or a group of files by using read-only and hidden attributes. In short, if a file has the read-only attribute, you can view the contents of a file but can't change it. If a file has the hidden attribute, the file is hidden from view; you won't see it listed when you use the Directory command.

The hidden attribute is particularly useful with directories, both because it can help reduce screen clutter by omitting hidden directories from directory listings and because it can help you hide directories and the files they contain from casual view.

If you're using version 3.2 or a later version of MS-DOS, you can also use the Attribute command to control the archive attribute of a file, which tells MS-DOS (or any other program that checks it) whether the file has been changed since the last time it was backed up for archival storage. Another attribute is the system attribute that is used to tell MS-DOS to treat a file as a system (program) file. If you have version 5 or later, you can use the system attribute. This attribute is normally used only by programmers.

This chapter describes the Attribute command in more detail and shows other examples of its use. The command has the following parameters:

attrib +R -R +A -A +H -H +S -S <filename> /S

+R turns on the read-only attribute; -R turns it off.

+A turns on the archive attribute (sometimes called the *archive bit*); -A turns it off.

+H turns on the hidden attribute; -H turns it off. If you have version 5 or later, you can use the hidden attribute.

+S turns on the system attribute; -S turns it off. If you have version 5 or later, you can use the system attribute.

<filename> is the drive, path, and name of the file or directory whose attributes are to be changed or displayed. You can use wildcard characters to specify a group of files. In versions 5 and later, you can omit <filename> to view or change the attributes of all files in the current directory. In effect, omitting the <filename> parameter in versions 5 and later is the same as specifying <filename> as *.*.

/S, in versions 3.3 and later, applies the Attribute command to every file in each subdirectory contained in <filename>. If you specify <filename> as the root directory of a disk and include /S, the Attribute command is applied to every file in every subdirectory on the disk.

Through version 4 of MS-DOS, if you omit all parameters except <filename>, MS-DOS displays the name or names of the files preceded by an A if the file has the archive attribute, an R if the file has the read-only attribute, or both if the file has both attributes set.

In versions 5 and later, omitting all parameters, including <filename> if you choose, causes MS-DOS to display the name or names of all files in the current directory. It precedes the file names with A and R for archive and read-only as it does in earlier versions, and uses H and S to indicate the hidden and system attributes.

Examples of Changing the Archive Attribute

The archive attribute is a part of the directory entry that isn't displayed by the Directory command but that can be examined or changed by MS-DOS or another program that checks it. This attribute can be used by the Xcopy command described later in this chapter under the heading "Using the Xcopy Command" and by some programs that back up files from a hard disk. The archive attribute is turned on by MS-DOS whenever a file is created or changed. Because the archive attribute tells MS-DOS or another program whether a file has been changed since the last time (if ever) it was backed up, it is used principally by Xcopy and other programs to determine which files must be backed up.

To complete the examples on the following pages, change the current directory to \RUNDOS\MKT\WP by typing this:

```
C:\RUNDOS>cd mkt\wp
```

As a first step, check the attributes of the files in the current directory. Type the following command if you have version 5 or later:

```
C:\RUNDOS\MKT\WP>attrib
```

Type this command if you have an earlier version:

```
C:\RUNDOS\MKT\WP>attrib *.*
```

MS-DOS responds with a display like the following:

```
A       C:\RUNDOS\MKT\WP\LET1.DOC
A       C:\RUNDOS\MKT\WP\LET2.DOC
A       C:\RUNDOS\MKT\WP\LET3.DOC
A       C:\RUNDOS\MKT\WP\RPT1.DOC
A       C:\RUNDOS\MKT\WP\RPT2.DOC
A       C:\RUNDOS\MKT\WP\RPT3.DOC
A       C:\RUNDOS\MKT\WP\LET1.STY
A       C:\RUNDOS\MKT\WP\LET2.STY
A       C:\RUNDOS\MKT\WP\LET3.STY
```

MS-DOS displays the name of each file with an A in column 3 because the files have never been backed up.

To turn off the archive attribute of all files with the extension DOC and check the result, type the following (in this and the following examples, you can omit the *.* if you're using version 5 or later):

```
C:\RUNDOS\MKT\WP>attrib -a *.doc

C:\RUNDOS\MKT\WP>attrib *.*
        C:\RUNDOS\MKT\WP\LET1.DOC
        C:\RUNDOS\MKT\WP\LET2.DOC
        C:\RUNDOS\MKT\WP\LET3.DOC
        C:\RUNDOS\MKT\WP\RPT1.DOC
        C:\RUNDOS\MKT\WP\RPT2.DOC
        C:\RUNDOS\MKT\WP\RPT3.DOC
  A     C:\RUNDOS\MKT\WP\LET1.STY
  A     C:\RUNDOS\MKT\WP\LET2.STY
  A     C:\RUNDOS\MKT\WP\LET3.STY
```

Now only those files whose extension is STY have the archive attribute.

If you're using 3.3 or a later version of MS-DOS, you can use the /S parameter to apply the Attribute command to all the files in subdirectories. To turn off the archive attribute of all files in the directories \RUNDOS\MKT, \RUNDOS\MKT\WP, and \RUNDOS\MKT\BUDGET and then check the results in all three, type the following commands to change the directory to \RUNDOS\MKT, turn off the archive attribute of all files in all subdirectories, and check the attributes of all files in all subdirectories:

```
C:\RUNDOS\MKT\WP>cd  ..

C:\RUNDOS\MKT>attrib -a *.* /s

C:\RUNDOS\MKT>attrib *.* /s
        C:\RUNDOS\MKT\WP\LET1.DOC
        C:\RUNDOS\MKT\WP\LET2.DOC
        C:\RUNDOS\MKT\WP\LET3.DOC
        C:\RUNDOS\MKT\WP\RPT1.DOC
        C:\RUNDOS\MKT\WP\RPT2.DOC
        C:\RUNDOS\MKT\WP\RPT3.DOC
        C:\RUNDOS\MKT\WP\LET1.STY
        C:\RUNDOS\MKT\WP\LET2.STY
        C:\RUNDOS\MKT\WP\LET3.STY
        C:\RUNDOS\MKT\BUDGET\BGT1.XLS
        C:\RUNDOS\MKT\BUDGET\BGT2.XLS
        C:\RUNDOS\MKT\BUDGET\BGT3.XLS
        C:\RUNDOS\MKT\ACCOUNT
```

The response to the second Attribute command shows that you turned off the archive attribute of all the files in \RUNDOS\MKT\WP, \RUNDOS\MKT\BUDGET, and \RUNDOS\MKT.

If you don't have version 5 or later, type the following command to turn on the archive attribute of all files in all subdirectories in \RUNDOS\MKT. (Examples later in the chapter

assume the attribute is turned on.)

```
C:\RUNDOS\MKT>attrib +a *.* /s
```

Now go on to the heading "Copying Selected Files."

If you do have version 5 or later, the following examples give you practice with other file attributes.

Examples of Changing Other Attributes

As already mentioned, the hidden attribute can be useful in minimizing screen clutter and, to some extent, in hiding files and directories from casual view. Because an experienced user of MS-DOS would know how to search for hidden files and directories, however, consider this attribute more of a convenience than a security measure.

To apply and change the hidden attribute with both directories and files, begin by hiding the \RUNDOS\MKT\BUDGET directory. Type this:

```
C:\RUNDOS\MKT>attrib +h budget
```

Now tell MS-DOS to show the directory for \RUNDOS and all its subdirectories, pausing after each screenful:

```
C:\RUNDOS\MKT>dir \rundos /s /p
```

MS-DOS lists \RUNDOS, \RUNDOS\MKT, and \RUNDOS\MKT\WP and all the files they contain, but it does not list the BUDGET directory or any of its files. However, if you use the /A parameter of the Directory command to tell MS-DOS to list all hidden files and directories in \RUNDOS and its subdirectories:

```
C:\RUNDOS\MKT>dir \rundos /ah /s
```

MS-DOS displays this:

```
Volume in drive C is HARD DISK
 Volume Serial Number is 1608-5A30

Directory of C:\RUNDOS\MKT

BUDGET      <DIR>     01-05-95  8:21a
      1 file(s)          0 bytes

Total files listed:
      1 file(s)          0 bytes
            15159296 bytes free
```

And if you check the attributes of \RUNDOS\MKT\BUDGET

```
C:\RUNDOS\MKT>attrib budget
```

you see this:

```
H     C:\RUNDOS\MKT\BUDGET
```

Next, to see how hiding files differs from hiding a directory, first remove the hidden attribute from \RUNDOS\MKT\BUDGET:

```
C:\RUNDOS\MKT>attrib -h budget
```

To reduce your typing chores, change to the \RUNDOS\MKT\BUDGET directory:

```
C:\RUNDOS\MKT>cd budget
```

Now apply the hidden attribute to the three files in the directory:

```
C:\RUNDOS\MKT\BUDGET>attrib +h *.xls
```

Type the command *dir \rundos /s /p* to again display the directory for \RUNDOS and all its subdirectories. This time, MS-DOS shows that \RUNDOS\MKT\BUDGET exists, but notice that the directory doesn't seem to contain any files. You removed the hidden attribute from the directory, so MS-DOS has no trouble including it in the listing, but you added the hidden attribute to the files in the directory, so now MS-DOS refuses to report on the contents of the directory. Even when you display the directory itself, MS-DOS does not show the hidden files:

```
C:\RUNDOS\MKT\BUDGET>dir

 Volume in drive C is HARD DISK
 Volume Serial Number is 1608-5A30
 Directory of C:\RUNDOS\MKT\BUDGET

.          <DIR>     01-05-95   8:21a
..         <DIR>     01-05-95   8:21a
       2 file(s)          0 bytes
                 15159296 bytes free
```

Check the attributes of the files in this directory, however, and MS-DOS responds:

```
C:\RUNDOS\MKT\BUDGET>attrib  *.xls
     H      C:\RUNDOS\MKT\BUDGET\BGT1.XLS
     H      C:\RUNDOS\MKT\BUDGET\BGT2.XLS
     H      C:\RUNDOS\MKT\BUDGET\BGT3.XLS
```

When files are hidden, you can't change their other attributes. For example, suppose you decide to make the files read-only; try it by typing this:

```
C:\RUNDOS\MKT\BUDGET>attrib +r *.xls
```

MS-DOS refuses:

```
Not resetting hidden file C:\RUNDOS\MKT\BUDGET\BGT1.XLS
Not resetting hidden file C:\RUNDOS\MKT\BUDGET\BGT2.XLS
```

```
Not resetting hidden file C:\RUNDOS\MKT\BUDGET\BGT3.XLS
```

You can, however, change other attributes if you also specify the hidden attribute. Make the files both read-only and hidden with one Attribute command:

```
C:\RUNDOS\MKT\BUDGET>attrib +r +h *.xls
```

Check the attributes again:

```
C:\RUNDOS\MKT\BUDGET>attrib  *.xls
    HR      C:\RUNDOS\MKT\BUDGET\BGT1.XLS
    HR      C:\RUNDOS\MKT\BUDGET\BGT2.XLS
    HR      C:\RUNDOS\MKT\BUDGET\BGT3.XLS
```

MS-DOS reports the files are now both hidden (H) and read-only (R).

To return all the files in RUNDOS and its subdirectories to their original state, first remove the hidden and read-only attributes from the budget files:

```
C:\RUNDOS\MKT\BUDGET>attrib -r -h *.xls
```

Now change the current directory to C:\RUNDOS\MKT, and turn on the archive attribute of all files in C:\RUNDOS\MKT and all its subdirectories:

```
C:\RUNDOS\MKT\BUDGET>cd ..
C:\RUNDOS\MKT>attrib +a *.* /s
```


◄ PREVIOUS NEXT ►

Copying Selected Files

In earlier chapters, you saw that you can copy files with similar names or extensions by using the Copy command and wildcard characters. If you're using 3.2 or a later version of MS-DOS, you have even more flexibility in copying files:

- The Replace command lets you replace all files in every subdirectory of a target disk that have the same name as the files on a source disk. The Replace command also lets you copy only the files on a source disk that *don't* exist on the target disk or in a target directory. Beginning with version 4, the Replace command also lets you update files by replacing only those on the target disk that are *older* than those of the same name on the source disk.

- The Xcopy command, as you saw in preparing for this chapter's examples, lets you copy whole subdirectories and the files in them. It also lets you copy only files that have changed since they were last backed up or those that have changed since a particular date.

Replacing Files on a Disk

The Replace command, like the Copy command, copies files from one disk or directory to another. The Replace command, however, is more selective:

- If used without the /A or /U parameter, the Replace command copies only source files that also exist on the target; it *replaces* files, hence its name.

- If used with the /A parameter, it lets you reverse the operation of the Replace command and tell it to copy only source files that *don't* exist on the target; with this option, it only *adds* files.

- If used with the /U parameter, the Replace command lets you copy only source files that are newer than files of the same name on the target; with this option, it *updates* files.

  The Replace command has eight parameters:

  replace <source> <target> /A /S /R /P /U /W

<source> is the name of the file to be copied. You can use wildcard characters to replace a set of files that have similar file names or extensions.

<target> specifies where <source> is to be copied. You can include a drive letter and a path name.

/A (for *add*) copies only the files specified in <source> that don't exist in <target>. This lets you add files to <target> without replacing files that already exist. If you don't specify /A, only files specified in <source> that also exist in <target> are copied. If you specify /A, you cannot specify /S or /U.

/S applies the Replace command to all subdirectories contained in <target>. If you specify <target> as the root directory of a disk, the command is applied to every subdirectory on the disk. If you specify /S, you cannot specify /A.

/R replaces those files in <target> that are read-only in addition to those that are normally replaced.

/P prompts you for confirmation before it replaces or adds each file.

/U (for *update*) replaces only those files on the target disk that are older than the corresponding files on the source disk. If you specify /U, you cannot specify /A.

/W prompts you to press a key before the Replace command begins. This lets you put in the correct floppy disk before starting to replace or add files.

To see how the Replace command works, copy the file named LET1.DOC from \RUNDOS\MKT\WP to \RUNDOS\MKT\BUDGET and then check the directory by typing this:

```
C:\RUNDOS\MKT>copy  wp\let1.doc budget
        1 file(s) copied

C:\RUNDOS\MKT>dir budget

 Volume in drive C is HARD DISK
 Volume Serial Number is 1608-5A30
 Directory of C:\RUNDOS\MKT\BUDGET

.          <DIR>     01-05-95   8:21a
..         <DIR>     01-05-95   8:21a
BGT1     XLS        24 01-05-95   8:17a
BGT2     XLS        24 01-05-95   8:17a
BGT3     XLS        24 01-05-95   8:17a
LET1     DOC        24 01-05-95   8:17a
       6 file(s)         96 bytes
                  15157248 bytes free
```

Now use the Replace command with only the <source> and <target> parameters to replace every file in the directory \RUNDOS\MKT\BUDGET whose name begins with LET and whose extension is DOC that also exists in \RUNDOS\MKT\WP. Type this:

```
C:\RUNDOS\MKT>replace wp\let*.doc budget
```

MS-DOS responds:

```
Replacing  C:\RUNDOS\MKT\BUDGET\LET1.DOC

1 file(s) replaced
```

The Replace command copied only the file—the one you copied at the beginning of this example—that already existed in \RUNDOS\MKT\BUDGET. The message tells you that MS-DOS *replaced* one existing file.

Now type the Replace command again, but this time include the /A parameter, which tells MS-DOS to copy only those files that do *not* exist in the target; you're doing the opposite of what you did with the previous Replace command:

```
C:\RUNDOS\MKT>replace wp\let*.doc budget /a
```

MS-DOS responds:

```
Adding  C:\RUNDOS\MKT\BUDGET\LET2.DOC

Adding  C:\RUNDOS\MKT\BUDGET\LET3.DOC

2 file(s) added
```

This time the Replace command copied the files that it didn't copy in the previous example. The message tells you that MS-DOS *added* files to the target directory.

Finally, if you have version 4 or later of MS-DOS, see how you can update files. Begin by creating a newer version of LET1.DOC in the directory \RUNDOS\MKT\WP (if you have version 6.2 or later of MS-DOS you'll be asked whether you want to replace the existing file; answer yes here and for later examples in this chapter):

```
C:\RUNDOS\MKT>copy con wp\let1.doc
This is a newer file.
^Z
       1 file(s) copied
```

Now you have one version of the file in \RUNDOS\MKT\BUDGET and a newer version with the same file name in \RUNDOS\MKT\WP. Update all files in the BUDGET subdirectory by typing this:

```
C:\RUNDOS\MKT>replace wp\let*.doc budget /u
```

MS-DOS responds:

```
Replacing  C:\RUNDOS\MKT\BUDGET\LET1.DOC

1 file(s) replaced
```

MS-DOS compared matching file names and updated only the one, LET1.DOC, that had, this time, a more recent version in the source directory than in the target directory.

Using the Xcopy Command

In preparing for this chapter, you used the Xcopy command to move an entire directory structure from one disk to another. If the corresponding subdirectories don't exist on the

target disk or directory, the Xcopy command creates them.

Unlike Copy and Replace, Xcopy includes the ability to copy only files whose archive attribute is on or files that have been changed since a date you specify.

The Xcopy command has 10 main parameters:

xcopy <source> <target> /A /M /D:<date> /E /P /S /V /W

<source> is the name of the file to be copied. You can include a path name, and you can use wildcard characters to copy a set of files with similar file names or extensions.

<target> specifies where <source> is to be copied. You can include any combination of drive letter, path name, and file name. If you specify different drives, the target directory is then allowed to have the same name as the source directory.

/A copies only those files whose archive attribute is on and leaves the archive attribute of the source unchanged.

/M copies only those files whose archive attribute is on and turns off the archive attribute of the source. This tells MS-DOS (or any other program) that the file hasn't been changed since it was last backed up and therefore doesn't need to be backed up again.

/D:<date> copies only files created or changed on or after <date>. (The date of creation or last change is the date shown in the directory entry for any file. Enter <date> just as you would for the Date command.)

/E creates subdirectories in <target> even if they're empty in <source>. This parameter can be used only when /S is used.

/P prompts for confirmation before each file specified in <source> is copied.

/S applies the Xcopy command to all subdirectories contained in <source> that are not empty. If you specify <source> as the root directory of a disk, the Xcopy command is applied to every non-empty subdirectory on the disk.

/V verifies that the copy of the file on <target> was stored correctly. This can slow the operation of the Xcopy command somewhat, but it's good insurance if you're copying critical data and must be certain that it was copied correctly.

/W prompts you to press a key before the Xcopy command begins. This wait gives you a chance to put in the correct floppy disk before starting to copy files.

Be aware that in MS-DOS version 6.0 and later, Xcopy does not copy files that have *hidden* or *system* attributes. This is a change from how version 5 and earlier handled such files. To have Xcopy copy these files, first use the Attribute command to turn off hidden and system attributes, then perform the Xcopy. After that, you can use the Attribute command to set one or the other or both of the attributes back on.

For this example, you create a temporary subdirectory named FRED in the root directory and dispose of it at the end of the example. First, change the current directory to \RUNDOS:

```
C:\RUNDOS\MKT>cd ..
```

Now type the following Make Directory command,

```
C:\RUNDOS>md \fred
```

and add the sample file you see at the top of the next page to \RUNDOS.

```
C:\RUNDOS>copy con test.doc
This is a test file.
^Z
        1 file(s) copied
```

All the files in \RUNDOS and its subdirectories have the archive attribute turned on. For the first Xcopy command, use the /A parameter to copy all the files in \RUNDOS whose extension is DOC and whose archive attribute is turned on:

```
C:\RUNDOS>xcopy *.doc \fred /a
```

MS-DOS responds by displaying the name of each source file as it is copied:

```
Reading source file(s)...
TEST.DOC
        1 File(s) copied
```

MS-DOS copied TEST.DOC, the only file in \RUNDOS whose extension is DOC. Verify this by displaying the directory of \FRED:

```
C:\RUNDOS>dir \fred

 Volume in drive C is HARD DISK
 Volume Serial Number is 1608-5A30
 Directory of C:\FRED

.           <DIR>     01-05-95  12:28p
..          <DIR>     01-05-95  12:28p
TEST     DOC        22 01-05-95  12:35p
        3 file(s)         22 bytes
                   14096384 bytes free
```

Now type the same Xcopy command, but add the /S parameter to copy all files whose extension is DOC and whose archive attribute is turned on, not only in \RUNDOS but also in all its subdirectories:

```
C:\RUNDOS>xcopy *.doc \fred /a /s
Reading source file(s)...
TEST.DOC
MKT\WP\LET1.DOC
```

```
MKT\WP\LET2.DOC
MKT\WP\LET3.DOC
MKT\WP\RPT1.DOC
MKT\WP\RPT2.DOC
MKT\WP\RPT3.DOC
MKT\BUDGET\LET1.DOC
MKT\BUDGET\LET2.DOC
MKT\BUDGET\LET3.DOC
      10 File(s) copied
```

This time MS-DOS copied 10 files; it also copied the subdirectories named MKT\WP and MKT\BUDGET. Verify this by displaying the directory of \FRED\MKT:

```
C:\RUNDOS>dir  \fred\mkt

 Volume in drive C is HARD DISK
 Volume Serial Number is 1608-5A30
 Directory of C:\FRED\MKT

.          <DIR>     01-05-95  12:28p
..         <DIR>     01-05-95  12:28p
WP         <DIR>     01-05-95  12:39p
BUDGET     <DIR>     01-05-95  12:39p
       4 file(s)          0 bytes
                   14071808 bytes free
```

The Xcopy command created the subdirectories named WP and BUDGET in the \FRED directory, then copied the files whose extension is DOC and whose archive attribute is on from the \RUNDOS\MKT\WP directory to \FRED\MKT\WP and from the \RUNDOS\MKT\BUDGET directory to \FRED\MKT\BUDGET.

The value of the archive attribute, however, is that it can tell you which files have changed since they were last backed up—if the archive attribute is turned off when a file is backed up and it is turned on when a file is changed. The /M parameter, like the /A parameter, tells Xcopy to copy only those files whose archive attribute is on; but it also turns the archive attribute off on the source file so that you can mark the files as backed up. Type another Xcopy command to copy the files whose extension is DOC, but this time use /M instead of /A:

```
C:\RUNDOS>xcopy *.doc \fred /m /s
```

MS-DOS responds just as it did to the previous Xcopy command because it copied the same 10 files, but this time it turned their archive attributes off. It's easy to check this; retype the last Xcopy command:

```
C:\RUNDOS>xcopy *.doc \fred /m /s
      0 File(s) copied
```

This time no files were copied because the previous command turned off the archive attributes of the files.

Now suppose you change one of the files. A new file's archive attribute is on, just as if it were an existing file you changed with the MS-DOS Editor or with a word processor, so type this to create a different version of \RUNDOS\TEST.DOC:

```
C:\RUNDOS>copy con test.doc
A new version of TEST.DOC.
^Z
        1 file(s) copied
```

Type the same Xcopy command you typed twice before; the first time Xcopy copied 10 files, the second time it copied no files, and this time it should copy one file:

```
C:\RUNDOS>xcopy *.doc \fred /m /s
Reading source file(s)...
TEST.DOC
        1 file(s) copied
```

Xcopy copied only the file whose archive attribute was on.

You'll be using only the RUNDOS directory structure in later chapters, so if you have version 6 of MS-DOS, type the following command to dispose of the directory you added for this example:

```
C:\RUNDOS>deltree \fred
Delete directory "\fred" and all its subdirectories? [yn] y
Deleting \fred...
```

If you have an earlier version, type the following commands:

```
C:\RUNDOS>del  \fred\mkt\wp\*.doc
C:\RUNDOS>rd  \fred\mkt\wp
C:\RUNDOS>del  \fred\mkt\budget\*.doc
C:\RUNDOS>rd  \fred\mkt\budget
C:\RUNDOS>rd  \fred\mkt
C:\RUNDOS>del  \fred\test.doc
C:\RUNDOS>rd  \fred
```

Next change the current directory back to \RUNDOS\MKT:

```
C:\RUNDOS>cd mkt
```

And, finally, clean up your \RUNDOS\MKT\BUDGET directory:

```
C:\RUNDOS\MKT>del  budget\let*.doc
```

Stretching Your Hard Disk

A hard disk is fast, durable, holds a lot more than a floppy disk, and generally does its job with little attention from you. It has one drawback, though: Hard disks, like garages and kitchen drawers, tend to fill up much more quickly than we expect (or would like).

When your hard disk fills up, the quickest way to alleviate the problem is to delete files you no longer need and archive files you might need at some point in the future. This housekeeping is time consuming, but you should do it periodically to make sure that you don't lose track of important files in the welter of other files that inevitably accumulate.

For a longer term solution, you can attack the problem from either the hardware side or the software side. On the hardware side, you can buy a larger hard disk and transfer your files to it or buy another hard disk and add it to your system. Both of these are fairly expensive propositions, and transferring your files to a new hard disk can take quite a while.

The software solution is simpler and more appealing: Using mathematical techniques, a program can reduce the number of bytes in a file—*compress* the file—without losing any of the information in the file. Starting with version 6.0, MS-DOS includes a hard disk compression program; in MS-DOS version 6.0 and 6.2, it's known as DoubleSpace, while in version 6.22 it's called DriveSpace. A compression program compresses all the files on your hard disk. If you have version 6.21, go on to the heading "Checking a Disk for Errors." However, if you have version 5 or earlier, go on to the heading "MS-DOS and Your Hard Disk."

Although the files on your disk will be compressed, the only difference you'll see on your disk will be a lot more available space. MS-DOS does the compressing behind the scenes. When you write a file to the disk, MS-DOS automatically compresses it. When you read the file, MS-DOS expands it, returning the file to its original state. For example, when you use the MS-DOS Editor to read a file, MS-DOS expands that file so that Edit can read its contents. You'll also notice that when you use the Directory command to see a list of the files in a directory on your compressed disk, MS-DOS displays the files' expanded sizes. And, if you copy a file from your compressed disk to an uncompressed floppy disk, MS-DOS will expand the file before copying it.

Compression Isn't Free

DriveSpace or DoubleSpace will, indeed, make your disk seem about twice as large as it was before. Although the physical capacity of the disk doesn't change, compressing the files lets the disk hold about twice as much.

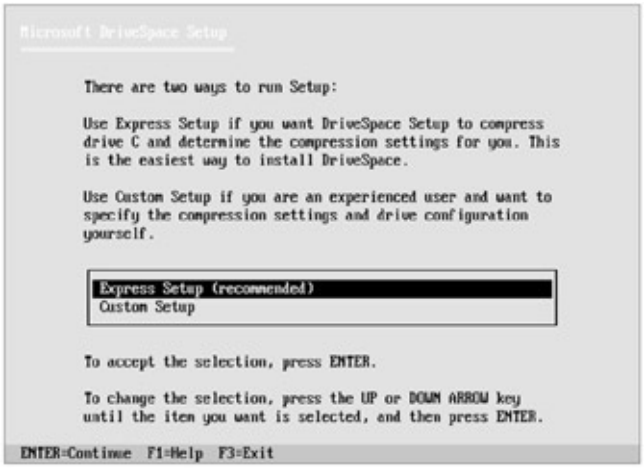
Nothing is free, of course. The price you pay for doubling the capacity of your hard disk is as much as 50 KB of memory that the compression program requires. The compression program, DriveSpace or DoubleSpace, will always be running, after you compress your disk, so that MS-DOS can compress and expand files as they're read from and written to the disk. Thanks to the memory management capabilities of MS-DOS, however, you can run

the compression program in upper memory so that you can minimize the effect of using up a lot of memory. In fact, if you're running MS-DOS version 6.2 or later, and have DOS=HIGH in CONFIG.SYS, MS-DOS will attempt to load part of the compression program into the high memory area (HMA) during the startup process.

Compressing Your Hard Disk

The remainder of this section covers the DriveSpace disk compression program introduced with MS-DOS version 6.22. The DoubleSpace program shipped in version 6.2 has for the DoubleSpace command and the DBLSPACE.SYS device driver exactly the same switches and parameters as those used for the DriveSpace command and the DRVSPACE.SYS device driver. The switches and parameters for the version 6.0 DoubleSpace command and DBLSPACE.SYS device driver are slightly different; for all versions, the online Help is a huge collection of relevant information.

To compress your hard disk, run the DriveSpace Setup program by typing *drvspace* at the system prompt. DriveSpace displays its opening setup screen, which lets you choose to continue, learn more about DriveSpace Setup, or quit. Press Enter to continue. The next screen tells you that you can either let DriveSpace make most of the decisions about the setup for you (*Express Setup*) or make the decisions yourself (*Custom Setup*):



Press Enter to accept *Express Setup*.

The next screen gives you a chance to turn back before installing DriveSpace and compressing your disk. Press C to start the process.

The DriveSpace setup program first runs the ScanDisk program to make sure that your hard disk is free of errors. (You'll read more about ScanDisk later in this chapter under the heading "Checking a Disk for Errors.") The setup program then examines your system and restarts your computer. Next, DriveSpace defragments and compresses the files on your hard drive, which *Express Setup* assumes is drive C. This might take a long time; the larger your hard disk, the longer it will take. DriveSpace lets you know how much time it estimates the process will take. During the compression process, DriveSpace displays the name of the file it is currently compressing. When it finishes with the compression, DriveSpace restarts

your system to complete the process.

After your system restarts, everything should work just as it did before. In normal use, the only difference you'll notice is a lot more room on your hard disk. A bit of checking, though, will reveal an extra disk drive that wasn't there before. For example, if you had no drives beyond drive C, now you have a drive H too. (Your computer might use another drive letter such as I.)

But you haven't added a hard disk drive, of course, so how can there be another disk drive? DriveSpace is responsible. To perform its magic, DriveSpace creates a file called a *compressed volume file* or *CVF*, which acts as drive C (the compressed drive). Drive H is the uncompressed drive, or *host* drive, that contains the compressed volume file. Drive H also stores hidden system files that MS-DOS uses when you start your machine and any other files that you don't want compressed. If DriveSpace finds a Windows permanent swap file on the drive being compressed, DriveSpace will move the file to the uncompressed drive. By default, DriveSpace leaves 2 MB of disk space free on drive H. Copy files to drive H only if you don't want them compressed.

Compressing a Floppy Disk

You can use DriveSpace to compress a high-density floppy disk just as you compressed your hard disk. Insert the floppy disk in drive A, and type *drvspace*. This time, you see DriveSpace's main screen (Figure 9-2), which shows you that drive C is a compressed drive.

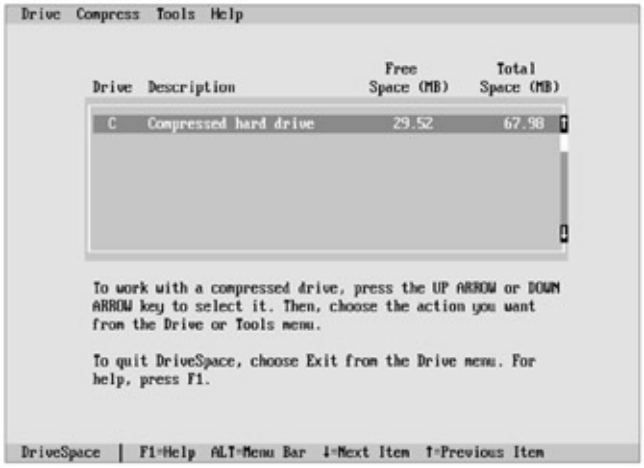


Figure 9-2: DriveSpace's main screen.

The bar at the top of the main screen contains four menus: Drive, Compress, Tools, and Help. These menus contain commands that cause DriveSpace to perform actions. To access a menu, press the Alt key and then the first letter of the menu's name. (If you have a mouse, you can access a menu by positioning the mouse pointer over the menu name and pressing the left mouse button.) To compress a floppy disk, first open the Compress menu by pressing Alt and then C, and then choose the *Existing drive* command in the menu by pressing Enter. (You can access the other commands in a menu by pressing the Down arrow key until the command you want is highlighted.) DriveSpace checks to see whether

any drives can be compressed and then tells you which ones it found. Chances are your list includes only drive A, but if you have another hard disk that you didn't compress or if you've defined a RAM disk, the list might include more than one drive letter. Using the arrow keys if necessary, highlight the line for the A drive and press Enter.

Now DriveSpace gives you a chance to change your mind; press C to start compressing the floppy disk in drive A. DriveSpace tells you approximately how long the process will take and then starts compressing. When it finishes, it displays the list of compressed drives again, this time adding drive A and identifying it as a *Compressed floppy disk*.

Now your floppy disk has nearly twice as much space available as it did before. If it had files on it before you compressed the drive, you'll see that the files are still available. As when you compressed your hard disk, your system has another disk drive: drive G. DriveSpace uses this uncompressed drive for maintaining the files on the disk. Because it doesn't contain much free space, drive G isn't much use for storing uncompressed files. If you need to put uncompressed files on a disk, simply use an uncompressed disk.

Note that you can use DriveSpace-compressed floppy disks only on machines that are running DriveSpace. If you want to transfer files to a computer that isn't using DriveSpace, be sure to copy the files to an uncompressed floppy disk. Also, MS-DOS doesn't allow you to use the Format command to format a compressed disk. Instead, use the /format parameter with the DriveSpace command. For example, when you type this,

```
C:\>drvspace /format a:
```

DriveSpace reformats the compressed disk in drive A. It deletes all the files but keeps the information that makes it compressed.

Mounting or Unmounting a Drive

Note DriveSpace automatically recognizes, or *mounts*, a compressed floppy disk unless you choose to turn off automatic mounting.

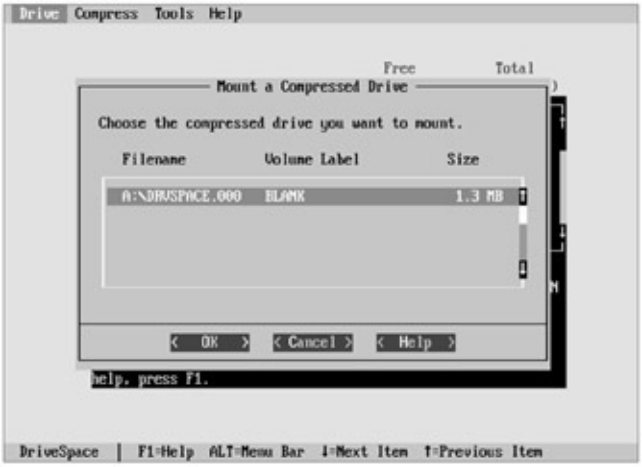
When you first compress a floppy disk, DriveSpace makes it available to MS-DOS for you, a process DriveSpace calls *mounting* the disk. The process of making the disk unavailable to MS-DOS is called *unmounting*.

The terms mount and unmount may seem odd. They are names that have survived from the earliest days of computing, long before personal computers were even imagined. Think about how computers are usually pictured in movies or cartoons: a refrigerator-like appliance with two large round devices at about eye level. This machine is a tape drive, used on early mainframe computers for most data storage because disk drives were enormously expensive. The round devices are the supply and take-up reels for the magnetic tape, and the act of loading a reel of tape into a tape drive was (and still is) called *mounting* the tape. You can see that DriveSpace uses the term a bit differently.

Your hard disk should remain mounted at all times. In fact, DriveSpace won't let you

unmount the drive from which you start MS-DOS.

To mount a compressed floppy disk from the main DriveSpace screen (Figure 9-2), insert the floppy disk in a drive. Use drive A for this example. Choose the Mount command from the Drive menu. (Press Alt then D to open the menu, then press the Down arrow key three times and press Enter.) DriveSpace searches for unmounted compressed drives and displays the ones it finds, as shown here.



In addition to the name of the uncompressed drive, DriveSpace lists the name of the compressed volume file, which is the file on your disk that contains the compressed drive. Next use the Up or Down arrow keys to select the compressed disk you want to mount. In this example, you would select A:\DRVSPACE.000. DriveSpace mounts the floppy disk and then updates the list of compressed drives on the main DriveSpace screen. Quit DriveSpace by choosing Exit from the Drive menu. The compressed floppy disk remains mounted.

To mount a compressed floppy disk at the system prompt, insert the floppy disk in a drive. Type *drvspace /mount* and the letter of the drive the floppy disk is in. For example, if you're using drive A, type this:

```
C:\>drvspace /mount a:
```

DriveSpace displays messages telling you it is mounting the drive and then that it has mounted the drive successfully.

Although it means a bit more effort when you use compressed floppy disks, DriveSpace is nonetheless a virtually cost-free way to double the storage capacity of all the disks you use.

Note You can use DriveSpace to further manage your compressed drives. You can change their sizes, create new drives from free disk space, view information about all your drives, and more. See Appendix C, "MS-DOS Command Reference," the online help for DRVSPACE, and your *MS-DOS User's Guide* for more information.

Removing DriveSpace

If you decided that you wanted to stop using file compression and return to the normal capacity of your hard disk, you could use the /Uncompress parameter of the DriveSpace command. The process takes about the same amount of time that it took to compress the disk. The /Uncompress can work only if there is enough free space on the uncompressed (host) drive to hold all the files from the compressed drive as they are decompressed. You may need to delete some unnecessary files or copy certain files you wish to keep to another hard drive or a floppy, in order to get the size of the uncompressed files to fit on the uncompressed drive.

When the last mounted drive is uncompressed, DriveSpace will remove itself from memory.

Note If you're not yet familiar with MS-DOS, you should probably get some assistance from a more experienced user before you delete DriveSpace. When you uncompress a drive, the host drive's letter will no longer be valid (unless there were multiple compressed drives on the same host drive), which will cause problems for any programs which have set up explicit path names and drive letters referring to the now-deleted host drive.

If your hard disk was less than half full when you installed DriveSpace and you haven't stored a lot of new files on the compressed disk since then, you can probably use DriveSpace to reduce the amount of available space on the compressed disk and increase the amount of uncompressed space.

To decrease the amount of compressed space, you start DriveSpace by typing *drvspace*, then choose *Change Size* from the Drive menu. DriveSpace displays a dialog box showing the current, minimum, and maximum sizes of the volume file (DRVSPACE.000) on both the uncompressed drive and the compressed drive (probably drive C). The cursor is in the field labeled *New size*, which should contain the same value as the *Current Size* field for DRVSPACE.000. Enter a value a bit larger than the *Minimum Size* field for DRVSPACE.000, and press Enter. DriveSpace tells you that it's changing the size of drive C, then tells you that it's remounting drive C. By reducing the size of the compressed drive, the size of the uncompressed drive is increased, providing more space for uncompressing files.

If you don't have enough space on your hard disk to hold uncompressed versions of the files you want to save, you'll have to back them up to floppy disks or another hard disk, using MSbackup or Xcopy. When you've backed them up, delete the compressed volume file (probably named DRVSPACE.000) from the host drive. Remember to restore these backed-up files after DriveSpace has been completely removed from the system. Now you must delete the Device command that DriveSpace puts in CONFIG.SYS and the files it creates in the root directory. You can use Edit to delete the line in CONFIG.SYS that reads *device=c:\dos\drvspace.sys / move*. The line might read *devicehigh*= instead of *device*=, and the command might include some parameters, such as */L:2,52272*.

Before you can delete the DriveSpace files in the root directory of drive C, you'll have to make the files accessible with the Attribute command (*attrib -s -h -r c:\drvspace.**), then

delete them with the Delete command (*del c:\drvspace.**). Press Ctrl-Alt-Del to restart the system, then type a Directory command and see how many bytes are available on the disk; your disk should be back to its uncompressed size.

Checking a Disk for Errors

Starting with version 6.2, MS-DOS includes a program called ScanDisk that does a more thorough job of checking for and correcting disk errors than the Check Disk (chkdsk) command of earlier versions. Both of the disk compression programs use ScanDisk to make sure that the files on a disk can safely be compressed. (The Check Disk command is still delivered as a part of MS-DOS, but you should use ScanDisk whenever you want to check your disks, compressed and uncompressed.)

ScanDisk checks both the directory structure information of the files on a disk and the surface of the disk itself for errors. In many cases, ScanDisk can correct structure errors—thereby recovering lost data—or move files from areas where the disk surface is damaged to undamaged locations. You can also use ScanDisk to check the compressed volume file that contains the files stored on a disk compressed with DriveSpace or DoubleSpace.

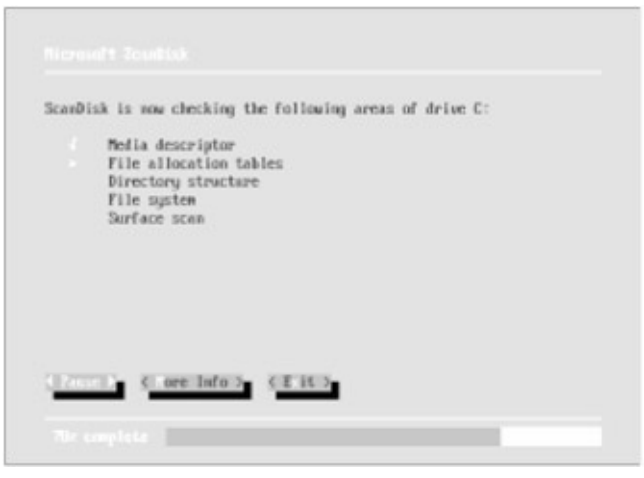
Like some other MS-DOS programs, such as MSbackup and DriveSpace, ScanDisk displays its own menus and dialog boxes, letting you choose your options with either the keyboard or a mouse.

Note You can specify all the ScanDisk options with command parameters, but that method of using ScanDisk is much less convenient and is included primarily to let you run the program from a batch file. If you would like to do this, the MS-DOS online help contains a complete description of all the parameters. Type *help scandisk* for this information.

The command to start ScanDisk has one parameter:

scandisk <drive>

<drive> is the letter, followed by a colon, of the drive to be checked. If you don't specify <drive>, MS-DOS checks the current drive.



Start ScanDisk by typing *scandisk*. ScanDisk displays its opening screen and begins checking the disk. This screen tells you which area of the disk ScanDisk is checking. The process takes just a few moments.

If ScanDisk detects errors in the file structure, it displays a dialog box titled *Problem Found* that lets you save the questionable data in a file in the root directory (just as the Check Disk command does), delete the data, or ignore it (leaving the disk unchanged). Unless you're missing data, choose the Delete option.

If you choose to delete the data, ScanDisk displays another dialog box, titled Create Undo Disk, that gives you the option of having ScanDisk create a file on a floppy disk—one that you can later use to reverse the steps that ScanDisk takes to delete the unwanted data and restore the disk to its previous condition. This only works if you use the undo disk before making any other changes to the disk, however. If you try to undo the deletion after changing the disk, ScanDisk won't be able to reconstruct the data and may damage the data structure, causing the loss of important files. Because the least bit of activity can cause one or two files to change on the disk, you probably shouldn't create an undo disk.

When ScanDisk finishes checking the file structure on the disk, it asks you whether you want to check the surface of the disk for defects. Although the estimates it displays are overly pessimistic, the process can still take several minutes. For now, choose No and return to the system prompt. You should, however, use ScanDisk to check the surface of your disk every couple of months to correct any problems that might develop before they cause the loss of valuable data.

PREVIOUS NEXT

Storing Files Efficiently on Disk

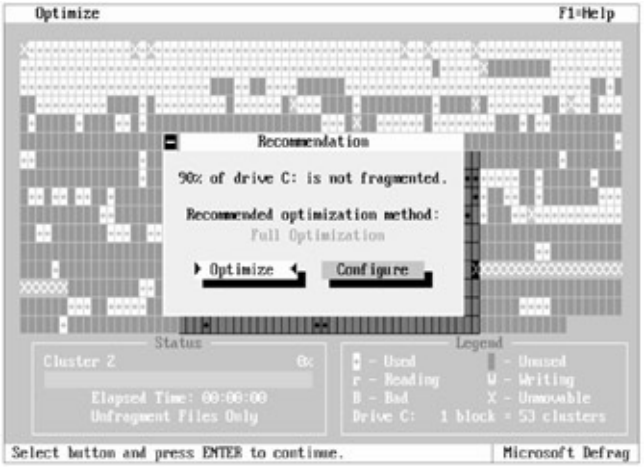
As described in Chapter 6, "Managing Your Floppy Disks," files on a disk can become *fragmented* (stored in two or more chunks in different parts of the disk) as you work with them, leaving unused spaces scattered around the disk. This can slow down your system, especially if many files are fragmented, because it takes MS-DOS longer to read and write files that aren't stored neatly in one place. You could experience a significant slowdown if all the available space on a disk were broken up into many small unused areas, requiring MS-DOS to break up every file you work with.

If you have version 6, you can use the MS-DOS Defrag program to pack all the files on a disk into one compact area, with each file stored in one location. Defrag lets you choose your options from menus; to start it, type *defrag*. After displaying a copyright notice, the Defrag program tells you it's testing system memory and then displays its opening screen:

Most likely, Defrag will have highlighted the line containing *C:*, suggesting that you optimize (or reorganize the files on) drive C. If it did, press Enter; if not, use the arrow keys to highlight *C:* and then press Enter. (If you prefer using the mouse, you can double-click on *C:* to choose it.) Defrag reads the files on the disk and then displays a box titled *Recommendation* that suggests whether you should optimize only the files, do a full optimization (moving directories, too, if necessary), or possibly not optimize at all.

Map of a Fragmented Disk

Regardless of the recommendation displayed in the dialog box, the main part of the screen shows a map of how the space on your disk is used. Each rectangle represents a portion of your disk. The map legend is in the lower right corner. Used areas appear white and unused areas appear shaded, so if you see any large white areas that contain smaller shaded areas, you're probably looking at a fragmented disk:



If your disk isn't badly fragmented, Defrag might recommend that you optimize only the files. Doing so will leave a few unused areas on the disk. To gain the maximum benefit from packing the files (and to see how full optimization works), you can request a full optimization. This isn't necessary, though; in most cases, if Defrag recommends that you optimize only the files, doing so will be sufficient.

If Defrag recommends full optimization, press Enter, and then go on to the next paragraph here. If Defrag does not recommend full optimization, press Esc to clear the Recommendation box. Defrag immediately displays the Optimize menu. Press the Down arrow key twice and then press Enter to choose *Optimization Method*. Now Defrag displays a box titled *Select Optimization Method* with *Full Optimization* selected. Press Enter, and then, when Defrag displays the Optimize menu again, highlight *Begin Optimization* by pressing the Up arrow key twice, and press Enter.

Now the optimization process begins. Defrag reorganizes all the files on the disk in order to pack them together. It displays a lowercase *r* in place of a white rectangle when it's reading a portion of a file to be moved, and an uppercase *W* when it writes the file in its new location. As you watch, any gray areas surrounded by white on the map should be disappearing, starting at the beginning of the disk (the upper left of the map).

Defrag displays *Finished condensing* when it finishes the defragmentation process. Press Enter to acknowledge the message. Next Defrag displays a dialog box that offers you the choice of condensing another drive, configuring the program, or exiting. Press the Right arrow key twice and press Enter to choose <Exit>, or press Esc to leave Defrag and return to MS-DOS.

You can also run Defrag by typing a command at the MS-DOS prompt and using parameters to specify the options. (The form of the command is shown in Appendix C, "MS-DOS Command Reference.") If you use a program that routinely modifies many files every day—for example, a word processor or data base program—and you want your system to run as quickly as possible, you could put the command *defrag c: /f* in your AUTOEXEC.BAT file. The /F parameter specifies a full optimization, so Defrag would fully optimize your hard disk every time you restarted your system.

MS-DOS and Your Hard Disk

Now that you're familiar with directories and with the commands you use to manage them, you can look at your own system and put some of your knowledge to work.

Because a hard disk can hold so many files, it's important to create a directory structure that lets you keep track of your files and programs. You can set up your subdirectories to match the way you use your computer, organizing them by department, application program, people's names, or any other way that's comfortable.

Until you get a directory structure established, however, files have a tendency to collect in the root directory, making it difficult to find a file you need. If you reserve the root directory for nothing but subdirectories and files that must be there, finding your way around your hard disk becomes much easier.

If you've been using your hard disk for some time and haven't known about or paid much attention to the idea of creating a directory structure on it, it's possible that the root directory of your hard disk contains many files that don't need to be there: MS-DOS files, for example, if MS-DOS is not in a subdirectory of its own; program files; data files; and perhaps files inherited from a former user of the computer. Some of these files, such as the MS-DOS command files, are essential but need not all be in the root directory; others, such as files you've inherited, might be unnecessary to your work; still others, such as application program files and data files, might be much easier to find and keep track of in directories of their own.

Few people, and certainly no book, can tell you the best way to organize your directories and files. If your own root directory contains many files that should be in subdirectories of their own, the Make Directory, Copy, Xcopy, and (when you've copied files to new subdirectories) Delete commands can help you with your housekeeping. MS-DOS 6 provides the Move command which can copy files, then delete them from the source location, all in one operation. Ultimately, you are the best judge of your own work and work habits, but on the next page are some general guidelines you can follow, either in setting up a new hard disk or in organizing one you've been using.

- Keep your MS-DOS command files in a subdirectory of their own. The only files that MS-DOS requires you to keep in the root directory are named COMMAND.COM, CONFIG.SYS, and AUTOEXEC.BAT. Versions 4 and later of MS-DOS install themselves so that these files are in the root directory, but all other MS-DOS files are in a separate subdirectory named \DOS. If you have an earlier version of MS-DOS and all of the MS-DOS files are in your root directory, read Appendix A, "Installing MS-DOS," to find out how to set up and move files to a \DOS subdirectory.

- Some application programs also require files in the root directory; the documentation that came with the program should tell you about this, and the program itself might create the necessary files when you install it. When in doubt,

follow the advice given in the documentation.

- If your hard disk has been used by someone else, print a listing of the root directory before you try reorganizing or deleting the files in it. Be particularly careful about removing any files with the extensions COM, SYS, or EXE; remember, they are program files. If you don't know what a particular program file is supposed to do, ask someone who does know.

- When you install a new program, put it in its own subdirectory if you can. Some programs propose this during the installation procedure. If yours doesn't, you might still be able to put it in a subdirectory, provided it can be installed on a hard disk and has the ability to work with different drives and subdirectories. Check the program's documentation for instructions. After a program is installed in a subdirectory, all you need do is add the subdirectory name to the command path in order to make the program's command files available at any time.

- Think about the data files you create. These are generally the files that take up the most room on a hard disk. They are also the ones that can become the most confusing as file names begin to proliferate and come close to duplicating one another. Although this book uses files with the names LET1.DOC, LET2.DOC, and so on, real data files with such names would mean little in a root directory cluttered with a hundred other letters. On the other hand, such names might mean a great deal in subdirectories named for clients, projects, proposals, and the like. In short, pay as much attention to organizing your hard disk as you would to organizing a large set of files in a file cabinet.

Taking Care of Your Hard Disk

Although MS-DOS can't help you physically protect your hard disk, there are some common-sense approaches you can use. Dire warnings aside, a hard disk is neither fragile nor temperamental. Many people use the same hard disk for years without encountering any problems. In fact, considering the close tolerances involved—the read/write heads literally float on air a tiny fraction of an inch above the disk itself—a hard disk is remarkably sturdy and reliable. All it normally needs is a little help from you. You might follow these practices, for example:

- Don't turn your system off and on repeatedly during the day. Leaving the system on minimizes wear on the mechanical parts of your hard disk.

- Don't turn the system off while the hard disk is active. By the same token, avoid bumping or jostling the system at those times too. You'll help preserve both disk and data.

- Keep the system and its surroundings reasonably clean. You don't have to sterilize anything; just clean and relatively dust-free will do.

- Save your work and turn off your system during storms and at other times when the

power is likely to surge or be interrupted. An unexpected loss of electricity or a sudden surge of excess power can cause problems, especially if the computer is reading from or writing to the hard disk at the time.

- If you have extra memory on your system, consider using some of it for a RAM disk. Not only will you reduce stress on your hard disk, you'll find that file access becomes noticeably quicker.
- Shut the system down before moving it. Don't worry about sliding it across your desk, but do be careful not to drop it or bang it into walls or furniture.

Barring equipment failure, about all it takes to keep your hard disk running smoothly are a little common sense and the same type of care you'd give your VCR or stereo system.

PREVIOUS NEXT

Chapter Summary

This chapter showed you how to set up a filing system, control the archive attribute, and copy files selectively with the Replace and Xcopy commands. It also showed how using the DriveSpace, or DoubleSpace, and Defrag programs lets you increase the capacity and efficiency of your hard disk and how using ScanDisk can help you check for and correct disk errors.

Because you can't see it, it's easy to take your hard disk for granted. But the hard disk is an essential part of your computer system. If you organize your file structure to match the way you work, the hard disk lets you move from job to job or program to program with a minimum of effort. Setting up a filing system and doing some periodic housekeeping takes a bit of time but pays off by letting you work more efficiently. More than any other part of the system, the hard disk determines how well your computer works for you.

PREVIOUS NEXT

Chapter 10: Protecting Your Disks and Files

Overview

You've seen that a hard disk is much faster and more convenient to use than floppy disks, even the latest high-capacity floppy disks that hold nearly three megabytes. But the same features that make a hard disk so valuable can also make loss of the disk or the files stored on it more distressing. It's bad enough to discover that MS-DOS either can't find or can't use files you stored on a floppy disk. It's almost always worse when the same thing happens to your hard disk.

MS-DOS tries to help protect your files by requesting confirmation before it carries out a Delete command that would remove all files in a directory. Recent versions of MS-DOS protect you from accidentally formatting a disk by requiring you to type a drive letter and, if you specify a hard disk, by prompting you to verify the command. In addition to these attempts to make you think twice before carrying out a potentially destructive command, MS-DOS also provides some commands that help you protect your valuable files.

The MSbackup command in version 6 (and the Backup and Restore commands in earlier versions) lets you copy files from your hard disk to floppy disks for safekeeping and then copy them back to the hard disk if you need them.

If, despite these safeguards, you still lose files accidentally, MS-DOS, starting with version 5, includes the Undelete and Unformat commands that recover a file after you delete it or restore the contents of a disk you inadvertently format. Version 6 continues this emphasis on guarding against or recovering from file damage by adding two extra levels of file protection to the Undelete command and by adding Undelete for Windows, a program that provides the same protection in Windows as the MS-DOS version of Undelete. Version 6 also offers anti-virus programs that run under both MS-DOS and Windows.

This chapter shows you how to use these protective features, including the MS-DOS versions of Undelete and anti-virus protection. See the documentation that comes with MS-DOS for a description of how to use the Windows versions of these programs.

The examples in this chapter require some preparation. If you still have the floppy disk you used for the examples in Chapter 5, "Managing Your Files," put it in drive A and go on to the section called "Recovering Deleted Files." If you don't have the example floppy disk, put a formatted floppy disk in drive A and type the following commands:

```
C:\>copy con a:report.txt
This is a dummy file.
^Z
    1 file(s) copied
C:\>a:
A:\>copy report.txt bank.txt
    1 file(s) copied
A:\>copy report.txt forecast.txt
    1 file(s) copied
```

```
A:\>copy report.txt budget.jan
    1 file(s) copied
A:\>copy report.txt budget.feb
    1 file(s) copied
```

Use the Directory command to check that all five files are on the floppy disk:

```
A:\>dir
 Volume in drive A is EXAMPLES 1
 Volume Serial Number is 1A2C-13F5
 Directory of A:\

REPORT   TXT        23 01-05-95   9:16a
BANK     TXT        23 01-05-95   9:16a
FORECAST TXT        23 01-05-95   9:16a
BUDGET   JAN        23 01-05-95   9:16a
BUDGET   FEB        23 01-05-95   9:16a
   5 file(s)       115 bytes
                1455104 bytes free
```

Depending on which floppy disk you put in drive A, your report may show more files than this, but the five file names shown here should appear there. The times and dates will be different, too, but the file names and sizes (23 bytes) should be the same.

This completes preparation for the examples in this chapter.

PREVIOUS NEXT

Recovering Deleted Files

Beginning with version 5, MS-DOS includes an Undelete command that can help you recover a file you've deleted. In most cases, when you delete a file, MS-DOS marks its storage space on the disk as available for reuse, but it doesn't physically remove the information contained in that space. When MS-DOS manages storage space in this way, it's possible to use the Undelete command to recover a deleted file until (and only until) MS-DOS reuses part or all of the file's disk space for another file. MS-DOS version 6 substantially changes the Undelete command by introducing two levels of file protection in addition to the one available in version 5. If you have version 5, skip to the section called "Undelete and Delete Tracking." The following section covers version 6.

Undelete's Levels of Protection

MS-DOS version 6 offers an enhanced Undelete command that helps you recover a file you've deleted. It includes three levels of protection: standard, Delete Tracker, and Delete Sentry. Standard uses information recorded by MS-DOS to recover a deleted file. It works by reinstating the information about the disk location where the deleted file was stored. Of the three levels of guarding against file deletion, standard provides the lowest level of protection. You can use it to recover a deleted file as long as MS-DOS has not placed another file in any portion of the deleted file's location. The standard level of protection is available when you switch on your computer.

Delete Tracker offers the next higher level of Undelete's file protection. It uses a hidden file named PCTRACKR.DEL to hold disk-storage information about the files you delete. Once you start Delete Tracker for a disk drive, Undelete records storage information for all subsequent deleted files. Undelete can then use this information to try to recover one or more deleted files. If MS-DOS has placed another file over part of the deleted file's location, Undelete attempts to restore part of the file. Note that Delete Tracker in version 6 works in the same way as the delete tracking feature in version 5, which is discussed in the next section, "Undelete and Delete Tracking."

To choose the Delete Tracker level of protection, type the Undelete command and specify the /T switch and the drive you want to protect. For example, to activate Delete Tracker on drive A, you would type the following at the command prompt (do not type this command now):

```
A:\>undelete /ta
```

Delete Sentry offers the highest level of protection against accidental file deletion. It creates a hidden directory named SENTRY to which a deleted file is moved without changing the record of the file's location. When you undelete the file, MS-DOS moves the file back to its original location. With Delete Sentry protection, you don't have to worry about MS-DOS placing another file in the deleted file's place. The deleted file remains tucked away on your disk, safe and sound. Undelete truly removes the file from your hard disk either after seven days or when Undelete needs more room in the SENTRY directory.

To activate the Delete Sentry level of protection, specify the /S switch and the drive you want to protect. If you're selecting the current drive, you don't need to include the drive letter designation. For example, if the current drive is A, type the following at the command prompt (type this command now after you're sure Microsoft Windows is not running):

```
A:\>undelete /s
```

Now delete the test files you created earlier:

```
A:\>del  report.txt
A:\>del  bank.txt
A:\>del  forecast.txt
A:\>del  budget.jan
A:\>del  budget.feb
```

Now check the directory again:

```
A:\>dir
Volume in drive A is EXAMPLES 1
Volume Serial Number is 1A2C-13F5
Directory of A:\

File not found
```

If there were other files on the floppy disk you put in drive A, they'll still appear in the report, but the five files you deleted should be gone.

If you have version 6, try using the Undelete command to retrieve all the files you deleted. The following list describes most of the parameters for the version 6 Undelete command. To find out about Undelete's advanced parameters, see Appendix C, "MS-DOS Command Reference."

undelete <filename> /ds /dt /dos /all /list /status /U /S<drive> /T<drive>

<filename> is the name of the file or files you want to restore. You can specify a drive and path, and you can use wildcards to specify a set of files. If you don't include <filename>, Undelete assumes you want to recover all deleted files in the current directory of the current drive.

/ds recovers only those files protected by the Delete Sentry level, prompting for confirmation on each file. Undelete assumes */ds* if the SENTRY directory exists, even if you don't include this parameter.

/dt recovers only those files protected by the Delete Tracker level, prompting for confirmation on each file.

/dos recovers only those files that are internally listed as being deleted by MS-DOS, prompting for confirmation on each file. If a deletion-tracking file exists, this switch causes Undelete to ignore it.

/all recovers all deleted files without prompting for confirmation on each.

/list lists the deleted files that are available to be recovered but does not recover any files. You can restrict the resulting list by using the <filename> parameter and the */ds /dt*, and */dos* switches.

/U disables the Delete Tracker or Delete Sentry protection.

/*status* displays the type of delete protection in effect for each drive. You cannot specify any other parameters when you use /*status*.

/S<drive> enables the Delete Sentry level of protection and loads the memory-resident portion of the Undelete program. If you omit <drive>, Undelete will protect the files on the current drive. You cannot specify any other parameters when you use /S<drive>.

/T<drive> enables the Delete Tracker level of protection and loads the memory-resident portion of the Undelete program. If you omit <drive>, Undelete will protect the files on the current drive. You cannot specify any other parameters when you use /T<drive>.

You just deleted several files from the disk in drive A. To request a list of these files, type this:

```
A:\>undelete /list
```

The Undelete command responds with a display like this:

```
UNDELETE - A delete protection facility
Copyright (C) 1987-1993 Central Point Software, Inc.
All rights reserved.

Directory: A:\
File Specifications: *.*

   Delete Sentry control file contains    5 deleted files.

   Deletion-tracking file not found.

   MS-DOS directory contains    5 deleted files.
   Of those,     0 files may be recovered.

Using the Delete Sentry method.

    REPORT   TXT       23 01-05-95 9:16a  ...A Deleted:  01-06-95 3:24p
    BANK     TXT       23 01-05-95 9:16a  ...A Deleted:  01-06-95 3:24p
    FORECAST TXT       23 01-05-95 9:16a  ...A Deleted:  01-06-95 3:25p
    BUDGET   JAN       70 01-05-95 9:16a  ...A Deleted:  01-06-95 3:25p
```

```
    BUDGET   FEB       23 01-05-95 9:16a  ...A Deleted:  01-06-95 3:25p
```

Let's examine this report. Although it's long, it's not difficult to interpret. After three lines of introductory text, Undelete lets you know where it is searching for the deleted files and what type of files it is searching for. In this case, it's looking in the root directory of the disk in drive A, and it's looking for all deleted files.

The next section of the report gives information about each level of protection. You can see that Undelete found five files protected by the Delete Sentry level, no files protected by the Delete Tracker level, and five files protected by the standard level (MS-DOS). Notice that in this case Undelete says that it can't recover the five files by using the standard method. You need to use the Delete Sentry method.

Finally, the report lists the five deleted files, giving their names, sizes, and dates and times of creation. The *...A* tells you that the files have not been backed up (archived), and the remainder of each line tells you the date and time the file was deleted.

Suppose you want to use the Undelete command to recover the file REPORT.TXT. By default, Undelete recovers only those files protected by the Delete Sentry level. Because you have been using the Delete Sentry level of protection, all you have to do is type the following:

```
A:>undelete report.txt
```

(Because we're now running under the Delete Sentry level of protection, if you had been using Delete Tracker when the file was deleted, you would have had to add the */dt* parameter, and if you had been using the standard level when the file was deleted, you would have had to add */dos*.) You see the same introductory text as before, and then Undelete displays this:

```
Directory: A:>
File Specifications: REPORT.TXT

   Delete Sentry control file contains    1 deleted files.

   Deletion-tracking file not found.

   MS-DOS directory contains    1 deleted files.
   Of those,     0 files may be recovered.

Using the Delete Sentry method.

    REPORT   TXT       23 01-05-95  9:16a ...A Deleted: 01-06-95 3:24p
This file can be 100% undeleted. Undelete (Y/N)?
```

Type *y*, and in a moment the recovery process ends:

```
File successfully undeleted.
```

To recover all deleted files on this disk, or to search through the deleted files and recover only those you want, you would then type *undelete*. For each file that Undelete could recover, it would display a message and prompt similar to those you just saw. In each case, you would type *y* or *n* to tell Undelete whether to recover or to ignore the file.

Before continuing, delete REPORT.TXT once again to return your sample disk to its earlier state.

Undelete and Delete Tracking

In version 5, Undelete works by reinstating the information about the disk locations where the deleted file was stored. Bear in mind, however, that MS-DOS does not necessarily store an entire file in a single location on disk. Sometimes, especially on much-used disks, it tucks sections of a file into widely separated storage areas in order to make the best possible use of available space. As you save files, even small ones, MS-DOS assigns and reassigns these storage areas. Because disk storage is so changeable, Undelete is successful only if it can follow a file's chain of storage locations from beginning to end. If, as often happens, MS-DOS uses a link in this chain for storing part of another file, Undelete reaches a dead end and cannot recover the entire file.

Note If you need Undelete, use it as soon as possible after realizing that you need to recover a deleted file. Each time MS-DOS saves another file on the disk, your chances of recovering the file are reduced.

You can use the MS-DOS 5 version of Undelete either on its own or with a feature called *delete tracking*. Delete tracking is activated by the Mirror command, which creates a special file named PCTRACKR.DEL for holding disk-storage information about the files you delete. (MS-DOS version 6 does not include the Mirror command because its functionality was added to Undelete.) Once you start delete tracking for a disk drive, the Mirror command records storage information for all subsequent deleted files. Undelete uses this information to try to recover one or more deleted files. Because delete tracking uses a file for holding disk-storage information, it tends to be more effective than using MS-DOS—based information in recovering files. The following examples show you how to start delete tracking and use it to restore deleted files.

When used for tracking deleted files, the Mirror command has this form:

mirror /T<drive>-<files>

<drive> is the letter (without a colon) of the drive for which you want to start delete tracking. You must include this parameter.

<files> is the number of deleted files you want to track. If you include <files>, you can specify from 1 through 999, separating it from <drive> with a hyphen. If you omit <files>, the delete tracker assumes a generous number based on the size of the disk: 25 for a 360-KB floppy disk, 50 for a 720-KB floppy disk, 75 for a 1.2-MB or a 1.44-MB floppy disk, and 101 to 303 for a hard disk, depending on its capacity.

To start delete tracking for the sample floppy disk in drive A, type this:

```
A:\>mirror /ta
```

Mirror responds with a set of messages, among them *Drive A being processed* and, when all goes well, *The MIRROR process was successful* and *Installation complete*.

Now delete the test files you created earlier:

```
A:\>del  report.txt
A:\>del  bank.txt
A:\>del  forecast.txt
A:\>del  budget.jan
A:\>del  budget.feb
```

Now check the directory again:

```
A:\>dir
 Volume in drive A is EXAMPLES 1
 Volume Serial Number is 1A2C-13F5
 Directory of A:\

File not found
```

If there were other files on the floppy disk you put in drive A, they'll still appear in the report, but the five files you deleted should be gone.

If you have version 5 or later, try out the Undelete command to retrieve the files you deleted. (If you don't have version 5, skip to the section called "Reformatting and Unformatting Disks.")

The version 5 Undelete command has the following parameters (version 6 uses the same parameters for delete-tracking file recovery):

undelete <filename> /dt /dos /all /list

<filename> is the name of the file or files you want to restore. You can specify a drive and path, and you can use wildcards to specify a set of files. If you don't include <filename>, Undelete assumes you want to recover all deleted files in the current directory of the current drive.

/dt tells Undelete to use the delete-tracking file recorded by the Mirror command. The */dt* parameter causes the Undelete command to prompt for confirmation before undeleting each file. Undelete assumes */dt* if a delete-tracking file exists, even if you don't include this parameter.

/dos tells Undelete to use information recorded by MS-DOS. Like the */dt* parameter, */dos* causes Undelete to prompt for confirmation. Because of the way MS-DOS deletes files, the */dos* parameter also causes Undelete to ask you to provide the first character in the name

of the file. Undelete assumes the */dos* parameter if a delete-tracking file does not exist, even if you omit this parameter.

/all causes Undelete to recover all possible deleted files without stopping to prompt for confirmation.

/list causes Undelete to display a list of files it can recover, without actually undeleting them.

You just deleted several files from the disk in drive A. To request a list of these files, type this:

```
A:\>undelete /list
```

The Undelete command responds with a display like this:

```
Directory: A:\
File Specifications: *.*
    Searching deletion-tracking file...
    Deletion-tracking file contains    5 deleted files.
    Of those,     5 files have all clusters available,
          0 files have some clusters available,
          0 files have no clusters available.

    MS-DOS directory contains    3 deleted files.
    Of those,    3 files may be recovered.

Using the deletion-tracking file.

    REPORT   TXT      23 01-05-95  9:16a  ...A Deleted: 01-06-95  3:24p
    BANK     TXT      23 01-05-95  9:16a  ...A Deleted: 01-06-95  3:24p
    FORECAST TXT      23 01-05-95  9:16a  ...A Deleted: 01-06-95  3:25p
    BUDGET   JAN      23 01-05-95  9:16a  ...A Deleted: 01-06-95  3:25p
    BUDGET   FEB      23 01-05-95  9:16a  ...A Deleted: 01-06-95  3:25p
```

Although longer and more detailed than most MS-DOS reports, this display is not difficult to interpret. First, Undelete tells you that it is checking the disk in drive A for all deleted files (*File Specifications: *.**).

The middle section of the report tells you what Undelete has found. If you use the delete-tracking file, Undelete can completely restore the five deleted files. (The message *all clusters available* means that Undelete can find all storage units allotted to each file.) On the other hand, if you use the information recorded by MS-DOS (by specifying the */dos* parameter), Undelete can find and recover three of the deleted files.

Because you started delete tracking, Undelete gives precedence to the delete-tracking file, so the bottom section of the report lists the five deleted files, giving their names, sizes, and dates and times of creation. The *...A* tells you that the files have not been backed up

(archived), and the remainder of each line tells you the date and time the file was deleted.

Note The delete-tracking file can help you recover files deleted after you have started delete tracking with the Mirror command. If you delete a file when this feature is not turned on, the delete-tracking file cannot help you recover it. However, because MS-DOS also records deletions, you might still be able to use the */dos* parameter of the Undelete command instead.

Suppose you want to recover one of the deleted files, REPORT.TXT, on the disk in drive A. Type this:

```
A:\>undelete report.txt
```

You see some preliminary messages telling you the recovery process has been started, and then Undelete displays this:

```
Using the deletion-tracking file.

    REPORT   TXT       23 01-05-95  9:16a  ...A Deleted: 01-06-95  3:24p
All of the clusters for this file are available. Undelete (Y/N)?
```

Type *y*, and in a moment the recovery process ends:

```
File successfully undeleted.
```

To recover all deleted files on this disk, or to search through the deleted files and recover those you wanted, you would type *undelete*. For each file that Undelete could recover, it would display a message and prompt similar to those you just saw. In each case, you would type *y* or *n* to tell Undelete whether to recover or to ignore the file.

Before continuing, delete REPORT.TXT once again to return your sample disk to its earlier state. Change the current directory to C:\ by typing these commands:

```
A:\>c:
C:\>cd \
```

Inadvertently deleting a file or files isn't the only way to lose valuable data. Until version 5, formatting a disk meant irretrievably erasing any files on the disk. Now, however, you can recover from that error too.

Reformatting and Unformatting Disks

If you have version 5 or later of MS-DOS, insert a new floppy disk in drive A to experiment with the Unformat command. If you don't have version 5 or later, skip to the section called "Developing a Backup Procedure."

You should always be careful about formatting disks that already contain program or data files. Starting with version 5, however, MS-DOS can help you recover from an inadvertent format. As you have seen, the Undelete command can often restore the deleted files. In much the same way, the Unformat command can help you restore the disk to its earlier state after you've mistakenly formatted it. The following examples show you how to reformat a disk with the /Q (quick) parameter of the Format command and then reverse the format with the Unformat command.

Note Version 5 of MS-DOS includes the Mirror command, which can be used to record the current status of a disk. This information can be used later when you unformat the disk. For more information about Mirror, see Appendix C, "MS-DOS Command Reference."

If you have version 5 or later, type the following to create a small file on the floppy disk in drive A so that you can see a change when you reformat it:

```
C:\>copy con a:myfile.doc
This is my sample file.
^Z
```

Use the Directory command to verify that MYFILE.DOC is on the floppy disk in drive A. While you're looking at the display, also make a note of the volume serial number.

Reformatting the Floppy Disk

Now reformat the floppy disk, this time using the /Q parameter of the Format command. Type this:

```
C:\>format a: /q
```

MS-DOS responds much as it does when you don't use /Q. First, it asks you to insert a floppy disk in drive A. The floppy disk is in the drive, so press Enter. Next, MS-DOS checks the existing disk format and, because the floppy disk has already been formatted, it tells you it's *Saving UNFORMAT information*. This message appears whenever you reformat a disk because MS-DOS itself saves disk-storage information.

When the quick format starts, your floppy disk drive becomes active for a short time, and MS-DOS tells you the size of the disk it's quick-formatting. This message is followed almost immediately by this:

```
Format complete
```

MS-DOS should now be asking you for a volume label:

```
Volume label (11 characters, ENTER for none)? _
```

Name the disk DOSDISK, and type *n* if MS-DOS asks this:

```
QuickFormat another  (Y/N)?
```

If you try using the Directory command again, you'll notice two changes: MYFILE.DOC no longer appears in the directory, and the volume serial number is now different. The changes show that the floppy disk has, indeed, been reformatted.

Unformatting the Floppy Disk

Note The following example shows how you can try to rebuild a floppy disk after mistakenly formatting it. Don't assume that you can reformat and rebuild floppy disks whenever you want. As you'll see shortly from some of the messages MS-DOS displays, rebuilding a disk can cause loss of stored data. Examples are included here to help you in emergencies. Copying and careful handling are still your best methods of protecting the information on your floppy disks.

Suppose now that you've just realized you formatted the wrong floppy disk. The one in drive A contained some valuable files that you want to recover. The Format command saves information that can be used to restore the disk to its former condition.

You can use the Unformat command to rebuild a disk, as long as you didn't use the /U parameter of the Format command to format the disk. This example uses only one parameter, the letter of the drive containing the disk to be unformatted. Other parameters for Unformat are described in Appendix C, "MS-DOS Command Reference."

To unformat the sample floppy disk in drive A, type this:

```
C:\>unformat a:
```

MS-DOS responds with the message asking you to insert the floppy disk to rebuild. The floppy disk is in the drive, so press Enter to begin unformatting.

Now you see a rather wordy response from MS-DOS, but read through it all. The message tells you that the information needed to rebuild the disk has been found in a file named MIRROR.FIL and that MS-DOS will use the file in attempting to unformat the disk. You also see a message telling you that rebuilding the disk could cause loss of information. Keep this message in mind; rely on this command to rebuild a disk, but don't make it part of your everyday work with MS-DOS. That's not how it's meant to be used.

If you're using version 5, MS-DOS tells you the last time the Mirror file was updated and asks you to press / (lowercase L) if you want to rebuild the disk using the file it found, *p* if you want to use the prior file (you rarely will), or Esc to end the command. You want to continue, so press /.

Versions 5 and later give you one more chance to change your mind (the following message appears on one line on your screen):

```
Are you sure you want to update the system area of your drive A
(Y/N)? _
```

This time you're sure, so press *y*. The drive becomes busy for a few moments, and MS-DOS finishes up with the message:

```
The system area of drive A has been rebuilt.

You may need to restart the system.
```

(You don't have to restart the system with this disk.)

It's done. If you check the directory of drive A, you'll find that Unformat has not only recovered MYFILE.DOC, it has even restored the original volume serial number of the disk.

Rebuilding a Hard Disk

Because of the extent of potential loss in terms of programs and data, recent versions of MS-DOS require deliberate effort on your part to format a hard disk. It's not a step to be taken lightly, and the most experienced computer user feels at least a small twinge after pressing Enter to start the process. Beginning with version 5, the Unformat command provides a means for you to undo the format of either a floppy disk or a hard disk.

Earlier in this chapter you used the Unformat command with floppy disks. You unformat a hard disk in the same way. Be aware, however, that files stored on a hard disk become unavailable after the disk has been formatted. In order for Unformat to work on a system with, for example, one hard disk and one floppy disk drive, you need a startup floppy disk with which you can start MS-DOS from drive A. You can create such a floppy disk with the Format /S command. Copy to this floppy disk the MS-DOS file UNFORMAT.COM as well as accurate copies of the files named AUTOEXEC.BAT and CONFIG.SYS, which are in the root directory of your startup disk.

You can use Unformat for one other task with your hard disk: rebuilding a damaged partition table. A damaged partition table means that MS-DOS can't find (and therefore start from) your hard disk. If you are unable to access your hard drive and you're using version 6 of MS-DOS, start your machine with a startup floppy disk and run the Unformat command. Unformat checks for a damaged partition table and fixes it if possible. If you are using version 5, you can use the Mirror command to save a copy of the partition table on a floppy disk. Then, if your hard disk's partition table becomes damaged, you can specify the */partn* parameter with the Undelete command to rebuild the partition table from the copy. See Appendix C, "MS-DOS Command Reference," for more information about Mirror and the version 5 parameters of Undelete.

Guarding Against Virus Programs

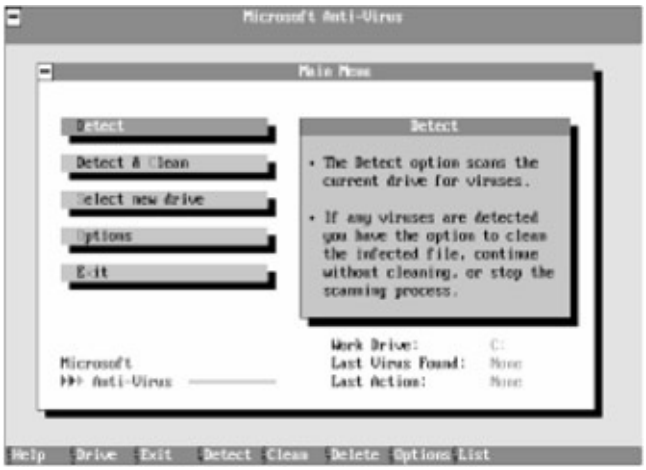
Computer virus is a term used to describe a program that hides itself on a disk—usually inside a program file—and can spread itself from system to system; a virus eventually does something to irritate you or even destroy your files.

There are now thousands of such programs, but the danger from them has been exaggerated in recent years. A virus can be introduced to your computer only if you use an "infected" floppy disk or if your computer is connected to another machine that has a virus or if you run a program that has been infected with a virus. Most software manufacturers check their disks for virus programs before boxing them, so if your machine isn't connected to any others and if you use only your disks and disks from programs you've bought, you shouldn't have to worry about viruses. However, it always makes sense to be cautious. We know that virus programs do exist and sometimes cause inconvenience or damage to your programs and data.

To let you protect your system against a virus, version 6 of MS-DOS includes programs that run under both MS-DOS and Windows to detect and remove virus programs from your disks. MS-DOS even includes a program you can install that runs all the time, guarding your system against a virus program that might wreak havoc on your files. We'll discuss the MS-DOS versions of these programs. See the *MS-DOS User's Guide*, which comes with MS-DOS, for a description of how to use the versions that were written for Microsoft Windows.

Checking Your Hard Disk

It's easy to check your hard disk for a virus; to start Microsoft Anti-Virus, the anti-virus program, type *msav*. You'll notice that the program first gets information about your disk. As it does so, a box on the screen titled "Reading disk information" displays which disk the program is reading and how many directories it has read so far. When it's done, Anti-Virus displays its opening screen:

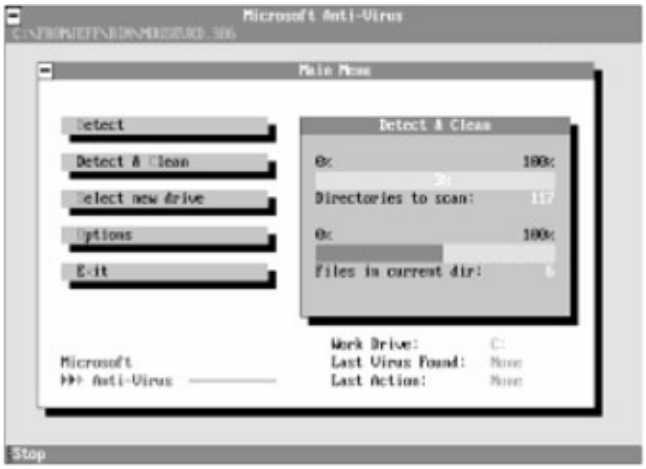


The window titled Main Menu offers five options, which are listed in the left half of the window: Detect, Detect & Clean, Select new drive, Options, and Exit. At this point the pane in the right half of the window describes the Detect option, which is the currently selected

option. It lets you know that if you choose Detect, Anti-Virus will scan the disk for viruses and give you the choice of correcting the problem (cleaning), continuing without cleaning, or quitting the scan.

For this example, you're going to scan the disk and get rid of any virus that Anti-Virus finds. Press the Tab key once to select the second option, Detect & Clean. As the pane to the right explains, if you choose Detect & Clean, Anti-Virus scans your disk for viruses and, if it finds any, cleans them without giving you a choice. Press Enter to begin the process now.

Anti-Virus first checks that there isn't a virus already running on your machine in memory. It then checks every file on the hard disk, showing you its progress in the pane on the right and displaying the name of the file it's currently checking in the upper left corner of the screen:



When Anti-Virus finishes, it displays a summary report showing how many disks and files it scanned and how many files it cleaned. Chances are your report shows that Anti-Virus found no infected files. If Anti-Virus does find an infected file, it cleans the file, adjusting for any changes the virus made to the file, and may prompt you to take additional action.

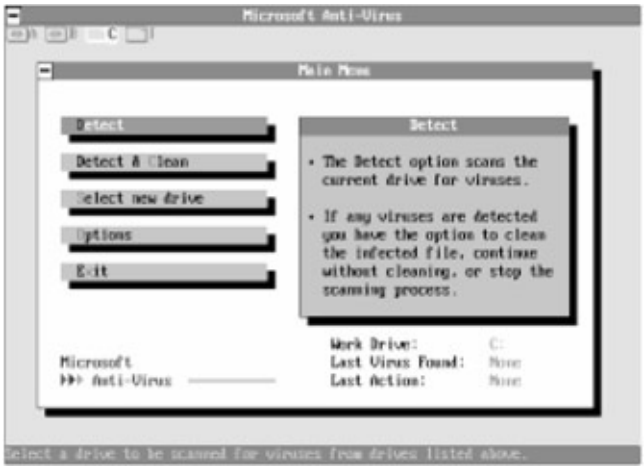
After checking the summary report (and taking any action recommended by Anti-Virus), close the report by pressing Enter to choose OK. Now quit the program by pressing the Tab key three times to select the Exit option. Then press Enter. Anti-Virus will display a box letting you know that you are closing Anti-Virus. If you're certain you want to quit, press Enter. If not, press Tab to select Cancel and then press Enter.

Checking a Floppy Disk

As mentioned earlier, floppy disks can be the transportation method of viruses from one computer to another. If you're using a disk whose origins you don't know, you would be wise to check the disk for viruses before copying anything from it to your hard disk.

Insert the disk you want to check in the disk drive. For this example, we'll use the A drive. Start Anti-Virus by typing *msav*. Choose the Select New Drive option by pressing the Tab key until the option is highlighted and then pressing Enter. A list of your disk drives appears

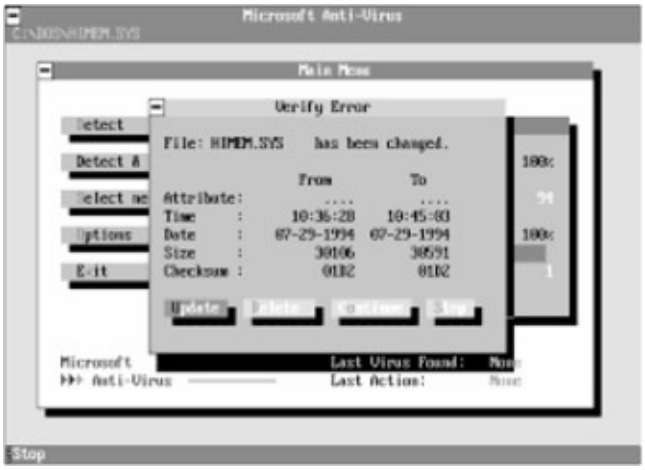
in the upper left corner of your screen:



Press the Left arrow key twice to select the A drive, and then press Enter. Anti-Virus gets information about your floppy disk the same way it gets information about your hard disk. After it finishes getting information, Anti-Virus is ready to check the disk. Choose the Detect option or the Detect & Clean option to start the scan.

Checking for Unknown Viruses

Anti-Virus checks for every virus that it knows. Of course, there are some new viruses that Anti-Virus doesn't recognize. Anti-Virus has a way to combat those as well. As mentioned earlier, most viruses infect program files. The first time Anti-Virus runs, it notes the size, attributes, date, and time of all the program files on your disk. The next time you check the disk with Anti-Virus, it checks that none of those program files have changed. A change might indicate the presence of a virus. If it finds a modified program file as it scans, Anti-Virus displays a box like the following one:



Anti-Virus lets you know that a program file has changed and gives you some choices of action. If you know why the file has changed—for example, if you've updated your program—press Enter to select Update and let Anti-Virus know that the file was legitimately changed. Be sure to check the name of the directory that contains the program file. It might give you a clue to a reason the file was changed. Suppose you have recently updated your

version of MS-DOS. You can see in the example above that the file HIMEM.SYS is in the MS-DOS directory. Even if you don't know what HIMEM.SYS does, the fact that it's in the MS-DOS directory gives you ample reason not to worry that it has been changed since the last time you checked for viruses. You should leave that file alone and simply select the Update choice. If you are absolutely positive that a file shouldn't have changed, then it might very well have a virus. Press Tab twice to select Continue, and then press Enter. If Anti-Virus finds a virus in the file you should delete the file and restore it from a backup.

Using Anti-Virus When Updating Software

If you update any software, such as your word processor or spreadsheet program, or even MS-DOS, many files will be changed legitimately. When you run Anti-Virus after updating the software, you will be prompted to accept the changes for every one of those files. You can avoid this time-consuming task by deleting the information Anti-Virus has saved about your program files before you update your software.

First, type *msav* to start Anti-Virus. Next choose the Detect & Clean option to scan your disk for viruses. When Anti-Virus completes its scan, press the F7 function key. A message box appears asking whether you're sure that you want to delete the checklist files. Press Enter to begin the process. (Press Tab and then Enter to cancel the operation.) Anti-Virus runs through the directories on your disk and deletes the information it saved about the program files. A box on the right side of the window keeps you informed of the progress. When the procedure is complete, exit Anti-Virus by pressing Tab until the Exit option is highlighted and then pressing Enter. Now you're ready to update your software. Follow the installation instructions that came with the software.

When your software update is complete, run Anti-Virus one more time to reinstate the information about your program files. Simply choose the Detect & Clean option again. Because you deleted the information about the older versions of your program files, Anti-Virus won't prompt you to accept the new versions.

Running Anti-Virus with a Command

You can also run Anti-Virus by typing a single command at the command prompt and using parameters to specify the options. The complete form of the command is shown in Appendix C, "MS-DOS Command Reference." For example, if you type the following:

```
C:\>msav c: /c
```

the /C parameter tells Anti-Virus to scan your memory and hard disk for viruses and clean any infected files it finds. If you want to be vigilant against viruses, you could run this command every time you start your computer. You can do this automatically by adding the command to your file named AUTOEXEC.BAT. See Chapter 14, "Creating Your Own Commands," for details about this special file. You probably don't need to run Anti-Virus that often, however, unless you routinely use a lot of disks or programs from other computer systems.

Constant Protection with Vsafe

If you want your system continuously monitored against the unwanted effects of a virus program, you can run the Vsafe program, available in version 6. This program monitors everything that happens in your system and makes sure that no unwanted programs are run. If Vsafe discovers something suspect, it will immediately display a message box containing a warning on your screen.

To load Vsafe, type *vsafe*. After Vsafe displays a message telling you it has successfully loaded, press Alt-V to display the Vsafe control menu, which contains a list of the actions Vsafe will protect against. The actions that are followed by an X are currently protected. An explanation of these actions and of the parameters that you can use with Vsafe is included in Appendix C, "MS-DOS Command Reference." To change whether a particular action is protected, type the number corresponding to the action. If the action is protected when you type the number, it will become unprotected; if the action is not protected, it will become protected. When you have finished making your selections, press Esc to return to the command line.

Vsafe uses quite a bit of memory (about 23 KB) and slows down your system because it checks all disk and memory activity. Note that you probably don't need the level of protection offered by Vsafe unless you routinely use a lot of disks or programs from other computer systems. You can unload Vsafe by typing *vsafe /u* or by pressing Alt-U when the control menu is displayed on the screen.

PREVIOUS NEXT

Developing a Backup Procedure

It could take a drawerful of floppy disks to back up all the files on a hard disk. If your average file were 10,000 bytes long (about 61/2 double-spaced typed pages), a full 40-MB hard disk would have more than 4000 files, and you'd need almost 35 high-density (1.2-MB) floppy disks to back them all up. A full 120-MB hard disk could require more than one hundred 1.2-MB floppy disks.

But you don't have to back up all your files. You needn't back up program files, for example, because you already have the original MS-DOS and application-program floppy disks. Some data files, such as a spelling dictionary, don't usually change, so it isn't necessary to back them up either.

How often you back up your other data files, such as word processing documents or spreadsheets, depends on how often they change. For example, spreadsheets might change often while the budget is being prepared but remain unchanged the rest of the year. The backup procedures you use depend on how you use your computer. But no matter how you decide to back up your files, be certain to do so regularly. A system failure can happen, but if you back up your files regularly, such a failure will be merely an inconvenience instead of a disaster.

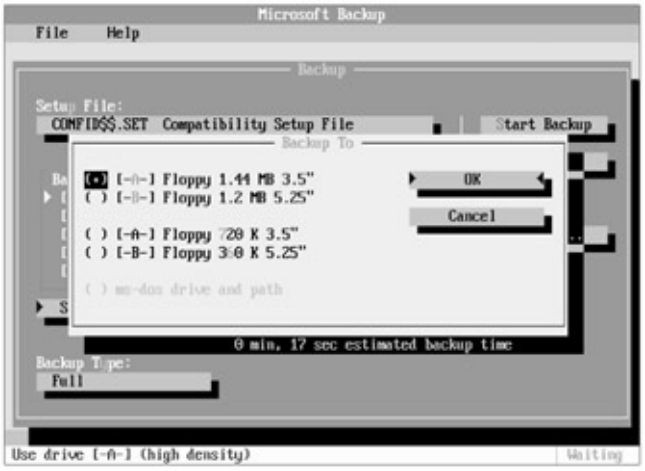
PREVIOUS NEXT

Backing Up and Restoring Files with MSbackup

If you're using version 6, you can use the MS-DOS program MSbackup (Microsoft Backup) to make backup copies of your files on floppy disks and restore files from the backup floppy disks to your hard disk. MSbackup does not require you to type commands. Instead it presents you with menus from which you choose the tasks you want the program to carry out.

To see how MSbackup works, make sure you have two formatted floppy disks. Both floppy disks must be the same type. Type *msbackup* at the system prompt. If this is the first time MSbackup has been run on your system, it tells you that *Backup requires configuration for this computer* and starts a series of steps that let it determine what sort of hardware you're using (especially what sort of floppy disk drives you have). Press Enter to proceed with configuration.

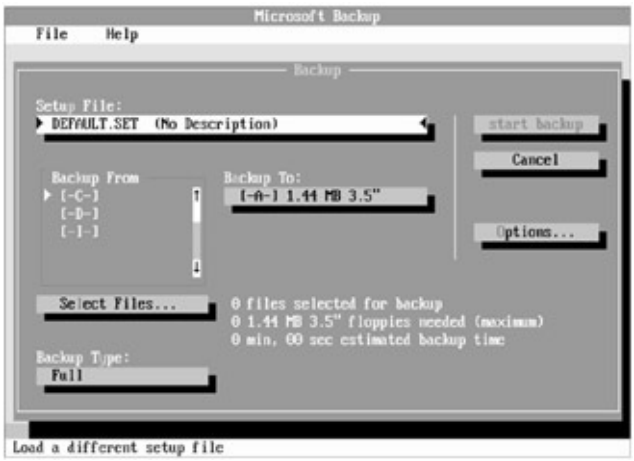
The configuration process is an automated run-through of many of the MSbackup commands, flashing through the screens almost faster than you can read them. Occasionally it will pause, display a message, and wait for you to press Enter. The fourth such instance occurs when MSbackup displays a box titled Floppy Disk Compatibility Test. The message explains that to ensure that MSbackup will create reliable backups, it is necessary to perform a test backup. Press Enter to start the test. This is the portion of the configuration in which you'll need the two disks. MSbackup automatically backs up some files from the MS-DOS directory. In the middle of the test, MSbackup displays a box titled Backup To in which you should specify which floppy disk drive you will use:



If you see a dot between the parentheses at the beginning of the line describing your floppy disk drive, your drive is selected. Simply press Enter. To select a different disk drive, type the character that is highlighted (if you have color, the letter is red) in the line you want to choose. You'll see that the dot between the parentheses moves to the correct line. Now press Enter to continue the process. MSbackup asks you to insert first one and then the other floppy disk in the drive you selected. After completing the backup, Msbackup will confirm the accuracy of the backup by comparing the files on the disks to the original files. The whole configuration process should take five minutes or so.

When configuration finishes, MSbackup displays a dialog box titled *Configure* with the option *Save* highlighted. Press Enter to save the configuration so that MSbackup will know how your system is configured from now on. This ends the configuration process.

Now, whether or not MSbackup checked the configuration of your system, it displays its main menu in a box titled *Backup* followed by a version number. The menu displays five choices: Backup, Restore, Compare, Configure, and Quit. Because *Backup* is highlighted, you can choose it now by pressing Enter. MSbackup displays the Backup screen:



This is the window from which you'll do most of your work.

Online Help

Like Anti-Virus and other menu-based MS-DOS programs, MSbackup includes online help that describes every function of the program. To see how it works, press F1. MSbackup displays a window titled *Backup Help*, which explains how to get help.

After reading this screen, press PgDn to move to the next screen, which shows a list of help topics (the choices on the Backup screen). Choose *Select Files button* by pressing the Tab key until it's highlighted and then pressing Enter; now MSbackup displays a description of the Select Backup Files screen and states how to select the files you want to back up. You'll see the Select Backup Files screen on page 228.

Scroll through the description; when you finish reading it, press Esc to return to the Backup screen. Remember, you can use online help to view a description of everything that MSbackup does.

Types of Backup

MSbackup offers three types of backup:

- Full, which backs up every file you select.
- Incremental, which backs up every file you select that has changed since the last full or incremental backup.

- Differential, which backs up every file you select that has changed since the last full backup.

If you start by doing a full backup and then do an *incremental* backup each night, you'd need the floppy disks for the full backup and all incremental backups to recover all the files. If you start by doing a full backup and then do a *differential* backup each night, you'd need only the floppy disks for the full backup and the latest incremental backup to recover all the files. The method you choose depends on how many files you want to back up and how many of those files you change in a typical work day.

If you don't change many files-and especially if the files you change tend to be the same files-then a differential backup is probably the most effective for you. You'll have the most current copies of your files saved. But if you change quite a few files every day, you're probably better off doing incremental backups and keeping all the incremental backup floppy disks until you do your next full backup.

Accessing the Parts of the Backup Screen

The Backup screen contains eight elements. The highlighted element is the one that is currently *selected*. To select a different element, press the Tab key until that element is highlighted. In most cases, once the highlight is on the element, you can press Enter to choose the element and perform an action. For example, press Tab until the highlight is on the element labeled Backup Type, and then press Enter. MSbackup displays a box that lets you choose the type of backup you want to perform:

This box contains a list of choices preceded by a set of parentheses, one of which contains a dot. The dot represents a selected item, and you can select only one item in this kind of list. There are two ways to select the item you want: You can press the key of the highlighted character in the line you want to select, or you can use the Up and Down arrow keys to move the highlight to the line you want to select and then press the Spacebar to move the dot to the highlighted line. If you want to cancel your selection and return to the Backup screen, either press Tab until the word Cancel is selected and press Enter or press the Esc key. To accept your selection, simply press Enter.

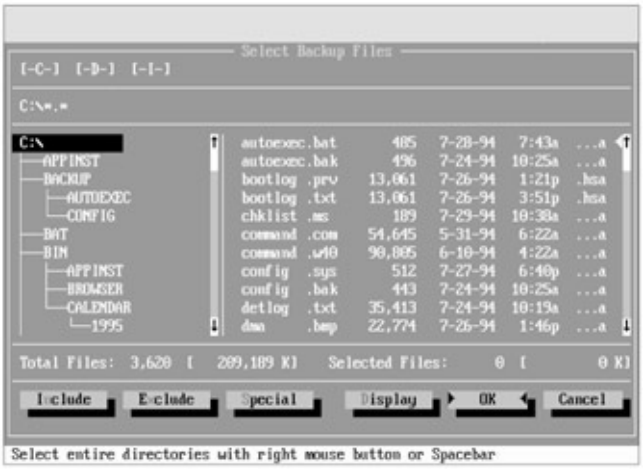
Now that you have an idea of how to move around in MSbackup, press Esc to return to the Backup screen.

Note This tutorial gives instructions for using the keyboard. If you have a mouse, feel free to use it. For example, with a mouse, choosing elements in the Backup screen is easy. All you have to do is position the mouse pointer over the element you would like to choose and then click (press and release) the left mouse button once. You can just as easily choose items from a list.

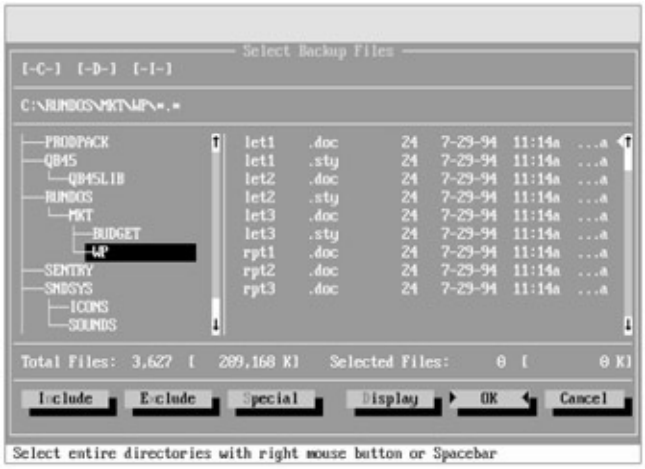
Selecting Files to Back Up

The examples that follow require the directory structure and files you created in the previous chapter. If you haven't completed those examples, do so before continuing.

You're going to back up some files now, so first choose *Select Files...* from the Backup screen. MSbackup reads all the file names from the current hard disk drive, then displays the Select Backup Files screen (you read the online help description of this screen on page 226):



As you can see, this screen contains a list of the directories on your disk and a list of the files in the currently selected directory. When MSbackup first displays this screen, the highlight is in the list of directories. Press the Down arrow key until you highlight the WP subdirectory in the C:\RUNDOS\MKT directory. The following figure shows the resulting display. The directories shown on your screen will most certainly be different from the ones shown here. However, your list of files in the WP directory should be the same.



Now press the Tab key to move the highlight to the section of the screen that contains the file names. To select a file for backup, use the Up and Down arrow keys to select the file name and then press the Spacebar. (If you're using a mouse, position the mouse pointer on the file name and press the right mouse button.) Try it: Press the Down arrow key twice to select the file named *let2.doc*, and then press the Spacebar; a check mark should appear to the left of the file name. In the same way, select the files RPT1.DOC, RPT2.DOC, and RPT3.DOC.

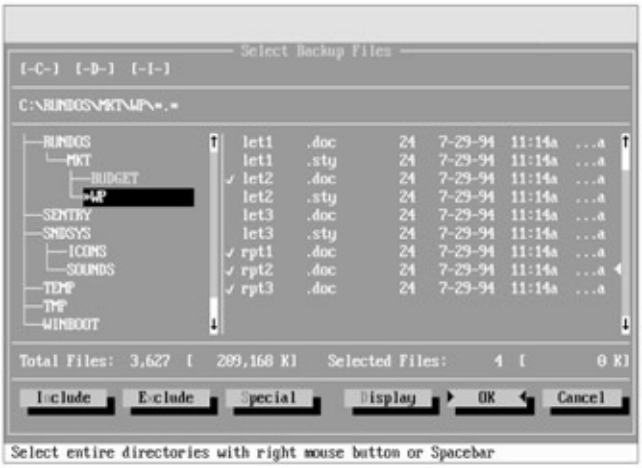
Note If you have a mouse, MSbackup makes it easy to select a series of files. All you have to do is position the mouse pointer over the first file name, press and hold the right mouse button, move the mouse pointer down to the last file in the series, and release the right mouse button.

Once you select it, a file stays selected. You can reverse the selection of a file by highlighting the file name and pressing the Spacebar again.

You can also select all the files in a directory. Press Tab until the highlight is back on the list of directories. Now press the Up arrow key once to select the directory named *BUDGET*, and then press the Spacebar. Now look at the list of file names on the right side; each one is preceded by a check mark. Press the Spacebar again. All the check marks disappear. Move back to the WP directory.

Backing Up the Files

You've marked four files to back up: They are LET2.DOC, RPT1.DOC, RPT2.DOC, and RPT3.DOC. The screen should look like this:



Press the Tab key until OK is selected, and then press Enter. MSbackup returns to the Backup screen. In the lower middle, the screen tells you that five files are to be backed up (the four you marked plus a catalog file that MSbackup creates), that you'll need one floppy disk, and that it will take about 15 seconds to back up the files. Insert one of the floppy disks that MSbackup used when it checked the configuration of your system, and then choose *Start Backup* at the upper right corner of the screen by highlighting it and pressing Enter.

Because you already have a disk in the drive, simply press Enter when MSbackup tells you to insert a disk in drive A. Because you're using a floppy disk that MSbackup wrote on while checking your system configuration, you will see a message telling you that you have inserted a disk that already contains backup files and asking whether to retry or to overwrite the disk. Because you won't need the files currently on the disk, select *Overwrite* and press Enter to continue.

It doesn't take long to back up the four files (probably less than the estimated 15 seconds). When MSbackup finishes, it displays a summary report and waits for you to press Enter. Now it returns once again to MSbackup's main menu.

MSbackup includes more options, but these are the basic functions you need to set up a backup procedure. The names of the files you chose to back up for this example are stored in a file named DEFAULT.SET. To make sure that these files won't be included in your backup procedure from now on, choose Backup to return to the Backup screen, choose *Select Files...*, and then remove the check marks from the four files (choose the directory \RUNDOS\MKT\WP, and then use the Spacebar to remove the check marks from the file names). Now choose *OK*, and check the description in the lower middle of the Backup screen; it should say you have zero (0) files selected.

Quit MSbackup by pressing Esc, pressing Enter when MSbackup asks if you want to save the changes to the DEFAULT.SET, and then finally choosing *Quit* from MSbackup's main menu.

Restoring Files Using MSbackup

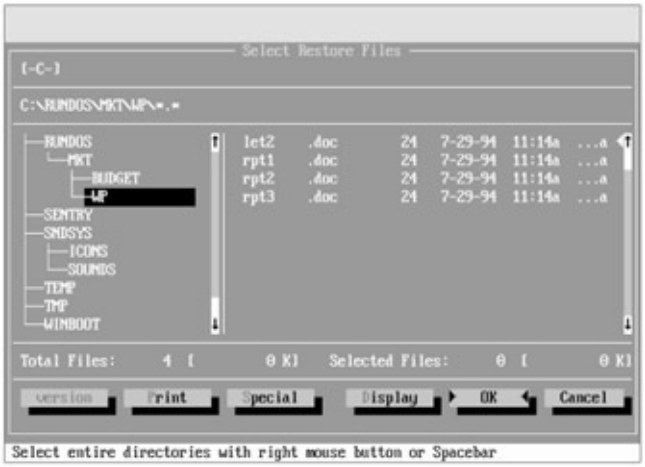
Restoring files works just about the same as backing them up. Now type the following to

delete three files whose file names start with RPT:

```
C:\>cd  rundos\mkt\wp
C:\RUNDOS\MKT\WP>del   rpt?.doc
```

Start MSbackup again by typing *msbackup*. This time, choose *Restore* from the main menu by selecting it and pressing Enter. The Restore screen looks very much like the Backup screen. Choose *Select Files...* at the lower left.

Again, the display of directories and files looks much as it does when you're backing up files. Highlight the directory \RUNDOS\MKT\WP. The names of the four files you backed up should appear:



As you did in the Select Backup Files screen, put check marks next to *rpt1.doc, rpt2.doc*, and *rpt3.doc*. Then choose *OK*. The Restore screen tells you *3 files selected for restore*. Choose *Start Restore* at the upper right of the screen. When MSbackup prompts you to put the disk in drive A, press Enter to choose OK because the floppy disk you used for the backup is still there.

Again, the restore process itself takes just a few seconds, and then MSbackup displays a summary report. Press Enter, and MSbackup returns to the main menu.

You have used the main features of both the Backup and Restore parts of MSbackup. You can experiment with the other features just as you did here. Whenever the meaning of a term or the purpose of a choice isn't clear, use the online help to get an explanation.

MSbackup for Windows

MSbackup for Windows works in much the same way as the version described here, although the appearance of the screens match Windows. It, too, offers online help; you should be able to use the Windows version, if you prefer, with no difficulty.

Backing Up Files with the Backup Command

If you're using version 5 or earlier, you must use the Backup command to back up files. Although it lacks the convenient menus and mouse features of the MSbackup program included with versions 6.0 and later, the Backup command (and its companion, the Restore command) nonetheless offers all the capability you need to back up your files.

The Backup command lets you select files on the basis of their path name, their file name, whether they have been changed since the last backup, or whether they have been changed since a particular date. The parameters can be combined, so you can back up files in just about any way you like.

The Backup command can have as many as nine parameters:

backup <source> <drive> /A /S /M /F /D:<date> /T:<time> /L:<logfile>

<source> is the file or set of files you want to back up. You can specify a drive (such as *a*:), a path (such as *a:\myfiles*), a file name with or without wildcards (such as *a:*.doc*), or a combination of these elements (such as *a:\myfiles\report.doc*).

<drive> is the letter, followed by a colon, of the drive (such as *a*:) that contains the disk that receives the backup files. You must specify <drive>.

/A adds the backup files to the backup disk, rather than erasing all files on the backup disk as the command usually does before making the backup copies.

/S backs up files from all subdirectories.

/M backs up only the files that have been modified since the last backup.

/F, in versions prior to 4, formats the target disk if it isn't already formatted. The /F parameter uses the Format command file (FORMAT.COM) to format the target disk, so FORMAT.COM must be in either the current directory or a directory that is in the command path. The /F parameter isn't required in versions 4 and 5. If you haven't already formatted the disk, note that MS-DOS automatically formats the floppy disk for the normal capacity of the drive, again assuming that it can find FORMAT.COM. You can, however, use the /F parameter either to format a set of floppy disks or to format one or more floppy disks with a capacity other than the one MS-DOS assumes for the floppy disk drive (for example, specify */f:360* to format 360-KB floppy disks in a 1.2-MB drive). When MS-DOS finishes whatever formatting you choose, it moves on to the backup procedure you've requested in your Backup command.

/D:<date> backs up all files that have changed since <date>. Enter <date> just as you would for the Date command.

/T:<time> backs up all files that have changed since <time> on <date>. Enter <time> just as you would for the Time command.

/L:<logfile> creates a log file on the disk in the source drive. The log file contains the date and time of the backup procedure and, for each file that is backed up, the path name, the file name, and the number (assigned by MS-DOS) of the floppy disk that contains the file. If a log file already exists, the backup information is added at the end, creating a history of backups for the source drive. If you include /L but omit the colon and <logfile>, MS-DOS names the log file BACKUP.LOG and stores it in the root directory of the source drive.

Note Although the Backup command exists in versions 2 and later of MS-DOS, not all the versions include all the parameters described here, nor are all the versions and releases of MS-DOS compatible with one another. For a list of the parameters available to you, check the documentation that came with your version of MS-DOS. You should use the same version of MS-DOS both to back up and to restore files.

Backing Up All the Files in a Directory

The simplest way to back up files is by directory. You back up all files in a directory by specifying just the path of the directory and the letter of the floppy disk drive that contains the backup floppy disk. To ensure that the examples proceed smoothly, have ready either a formatted blank floppy disk or a new floppy disk that matches the capacity of your floppy disk drive. (Remember, if you have version 3 or earlier, the floppy disk must be formatted before you back up any files.)

To back up all the files in the directory named \RUNDOS\MKT\BUDGET, type the following:

```
C:\RUNDOS\MKT>backup budget a:
```

MS-DOS displays a warning:

```
Insert backup floppy disk 01 in drive A:

WARNING! Files in the target drive
A:\ root directory will be erased
Press any key to continue . . .
```

If you don't use the /A parameter, MS-DOS erases any files on the backup floppy disk before it makes the backup copies. This warning gives you a chance to make certain the correct floppy disk is in the drive. Put your blank floppy disk in drive A, and press any key.

If you have version 4 or 5 and are using an unformatted floppy disk, MS-DOS begins the backup procedure by formatting the floppy disk to the maximum capacity of the drive, unless you used the /F parameter, or the /T and /N parameters. When the backup procedure begins, MS-DOS displays the name of each file as it makes the copies:

```
*** Backing up files to drive A: ***
Floppy disk Number: 01

\RUNDOS\MKT\BUDGET\BGT1.XLS
```

```
\RUNDOS\MKT\BUDGET\BGT2.XLS
\RUNDOS\MKT\BUDGET\BGT3.XLS
```

The directory of the backup floppy disk shows one or two files you might not expect. Type the following:

```
C:\RUNDOS\MKI>dir  a:
```

If you're using version 3.3 or later, the MS-DOS response to your directory command looks something like this:

```
Volume in drive A is BACKUP  001
 Volume Serial Number is 2543-14F7
 Directory of A:\

BACKUP   001        72 01-05-95   1:48p
CONTROL  001       311 01-05-95   1:48p
    2 file(s)        383 bytes
                  1456640 bytes free
```

MS-DOS has stored all the backed-up files in the file named BACKUP.001 and all the path names in the file named CONTROL.001. (On a second backup floppy disk, the extensions would be 002; on a third backup floppy disk, they would be 003.) Note that the size of BACKUP.001 corresponds to the total number of bytes in the three files it contains; CONTROL.001 contains all the extra information MS-DOS needs to restore those files.

If you're using version 3.2 or earlier, on a 360-KB floppy disk, MS-DOS responds:

```
Volume in drive A has no label
 Directory of  A:\

BACKUPID @@@      128  01-05-95  1:48p
BGT1     XLS      152  01-05-95  8:17a
BGT2     XLS      152  01-05-95  8:17a
BGT3     XLS      152  01-05-95  8:17a
       4 File(s)   358400 bytes free
```

BACKUPID.@@@ is a small file that MS-DOS stores on a backup floppy disk to identify it. Also, note that MS-DOS has added 128 bytes to each of the files you backed up; this addition contains the path and file name of the file that was backed up and is used by the Restore command. The Restore command deletes the path and file name information, so the restored version of the file is identical to the one you backed up. You'll work with the Restore command later in the chapter.

If a Backup command fills the floppy disk before backing up all the files you specified, MS-DOS prompts you to put in another floppy disk. It displays the same warning but refers to the second floppy disk as Floppy Disk Number 02.

If another floppy disk is required, MS-DOS prompts again, increasing the floppy disk

number each time. If you were actually backing up files, you would label the floppy disk you just used with the contents and date and store it in a safe place.

Backing Up All Subdirectories

You can back up the files in the current directory and all its subdirectories with the /S parameter. For example, the current directory is \RUNDOS\MKT. To back up all the files in it and all its subdirectories, you would type *backup *.* a: /s*. MS-DOS would display the names of all 13 files it backed up from the directories \RUNDOS\MKT, \RUNDOS\MKT\WP, and \RUNDOS\MKT\BUDGET. This backup floppy disk would contain all your marketing files, not just the files from one of the subdirectories.

Backing Up a Specific File

You can back up a specific file by including a file name with the Backup command. For example, if the current directory were \RUNDOS\MKT\WP, you would type *backup let1.doc a:* to back up just the file LET1.DOC. You can use wildcard characters to back up a set of files with the same name or extension. For example, typing *backup *.doc a:* would back up all files whose extension is DOC.

Backing Up Only Files That Have Changed

As the number of your files increases, you might want to be even more selective about the ones you back up. For example, a word processing directory might contain hundreds of documents, but only a few you're working with now. The /M (Modify) parameter of the Backup command backs up only those files that have changed since the directory was last backed up. To see how this parameter is used, you need a file that has changed since you backed up \RUNDOS\MKT in the last example. Create a short file by copying from the console:

```
C:\RUNDOS\MKT>copy con new.doc
This file has changed
since the last backup.
^Z
        1 file(s) copied
```

Now tell MS-DOS to back up any file that has changed since the directory was last backed up:

```
C:\RUNDOS\MKT>backup *.* a: /m
```

MS-DOS displays its warning and, when you press a key, displays the backed up files:

```
*** Backing up files to drive A: ***
Floppy disk Number: 01

\RUNDOS\MKT\NEW.DOC
```

Only the new file is backed up.

You can also back up only those files that have changed since a particular date with the /D:<date> parameter. For example, if you typed the command *backup *.* /d:01-05-95*, MS-DOS would back up all files that were created or changed on or after January 5, 1995, whether or not they had been backed up before.

Adding Files to a Backup Floppy Disk

Each form of the Backup command you have used so far starts by erasing any files on the backup floppy disk. There might be times, however, when you want to back up files from several different directories on one floppy disk or add a file or two to an existing backup floppy disk. The /A parameter adds a file to a backup floppy disk without erasing any backup files it contains.

If you periodically back up a few files from several different directories, you can use the /A parameter to put all the backup files on one floppy disk. A word of caution about this technique: With versions of MS-DOS through 3.2, if a file you add to a backup floppy disk has the same name and extension as a file already on the floppy disk, MS-DOS changes the extension of the added file to @01, regardless of what it was before.

PREVIOUS NEXT

Restoring Files to the Hard Disk with the Restore Command

It's easy to restore a file from a backup floppy disk to the hard disk using the Restore command. Simply insert the backup floppy disk in the floppy disk drive and type *Restore*, specifying the name of the file or files to be restored. The Restore command needs the path and file name information added to files by the Backup command, so you can restore only files that were backed up with the Backup command. The Restore command in versions 5 and later can restore files backed up with the Backup command in any prior version of MS-DOS. If you don't have version 5 or later, however, you should use the same version of MS-DOS for both the Backup and Restore commands.

The Restore command can have as many as 12 parameters:

> **restore <drive> <path><filename> /S /P /M /N /D /A:<date> /B:<date> /E:<time> /L:<time>**

<drive> is the letter, followed by a colon, of the drive (such as *a*:) that contains the backup floppy disk. You must include <drive>.

<path> is the path name, preceded by a drive letter and a colon, if appropriate, of the directory to which the file is to be restored. If you omit <path>, the file is restored to the current directory. Note that the file must be restored to the same directory from which it was backed up.

<filename> is the name of the file to be restored. You can use wildcard characters to restore a set of files. You must specify either <path> or <filename>.

/S restores files to all subdirectories.

/P tells MS-DOS to prompt you for confirmation before restoring read-only files or files that have changed since they were last backed up.

/M restores files that were *modified* since they were backed up.

/N restores files that have been deleted from the original source disk since they were backed up.

/D doesn't restore any files; it displays a list of the files stored on the backup floppy disk that match the path and file names you specified as part of the Restore command (version 5 and later, only). <path> must be specified, even though no files will be restored.

/B:<date> restores only those files that were changed on or *before* <date>. Enter <date> just as you would for the Date command.

/A:<date> restores only those files that were changed on or *after* <date>. Enter <date> just as you would for the Date command.

/E:<time> restores only those files that were changed at or *earlier than* <time> on <date>.

Enter <time> just as you would for the Time command.

/L:<time> restores only those files that were changed at or *later than* <time> on <date>. Enter <time> just as you would for the Time command.

Note Although the Restore command exists in versions of MS-DOS numbered 2 and later, not all the versions include all the parameters described here. For a list of the parameters available to you, check the documentation that came with your version of MS-DOS.

Warning Don't use the Backup or Restore command if you have entered an Assign, Join, or Substitute command to alter the way MS-DOS interprets drive letters. Because these commands can mask the type of drive, MS-DOS could damage or delete the files you specify in the commands or other files on the disk.

To prepare for the Restore command examples that follow, type the following command to back up all the files in \RUNDOS\MKT\WP:

```
C:\RUNDOS\MKT\WP>backup *.* a: /I
```

MS-DOS issues the usual warnings; press a key to back up the files. After MS-DOS displays the names of the files it backed up, delete all the files from the hard disk by typing this:

```
C:\RUNDOS\MKT\WP>del *.*
```

This completes preparation for the Restore examples.

Restoring a Specific File

Inadvertently erasing or changing a file is probably the most common reason for restoring a file. You can restore a specific file by specifying the source drive and the target drive, path name, and file name with the Restore command. (You must include the path if you are restoring a file to a directory other than the current directory, and the path you specify must be the same as the path from which the file was originally backed up.)

For example, if the current directory were \RUNDOS\MKT\WP, you would type *restore a: let1.sty* to restore only the file LET1.STY (You can omit the path because you're restoring the file to the current directory.)

You can use wildcard characters to restore a set of files with the same name or extension. For example, to restore all the files whose extension is DOC that you backed up from the current directory (\RUNDOS\MKT\WP), you would type *restore a:*.doc*.

If you enter the Restore command with a path name and the wildcard specification *.*, MS-DOS restores all files that belong in that directory.

Restoring All Subdirectories

Just as the /S parameter of the Backup command backs up the files in a directory and all its subdirectories, the /S parameter of the Restore command restores the files in a directory and all its subdirectories. To restore the files in \RUNDOS and all its subdirectories, you would type *restore a: \rundos*.* /s*.

Selecting Files to Be Restored

A file you restore replaces a file with the same name on the hard disk. You might not want this replacement, especially if you have changed the file on the hard disk since the backup floppy disk was made. You can protect yourself from unwanted changes by using the /P (Prompt) parameter of the Restore command, which tells MS-DOS to prompt for confirmation if the file on the hard disk has changed since the backup was made.

Warning If you're using version 3.2 or earlier and are restoring files that were backed up with a previous version of MS-DOS, be sure to use the /P parameter. If MS-DOS asks whether to restore files named MSDOS.SYS, IO.SYS, IBMBIO.COM, IBMDOS.COM, or COMMAND.COM, reply *n* (no). Otherwise, you would replace parts of your MS-DOS program with portions of an earlier version, and MS-DOS wouldn't start from your hard disk.

Remember, backing up your files is your best and cheapest insurance against hard disk problems and those occasional lapses that lead us to make mistakes. It doesn't matter how well you know how to use the backup capabilities if you don't use them.

Chapter 11: The MS-DOS Shell

Overview

Versions 4 through 6.0 of MS-DOS include a program called the MS-DOS Shell that lets you use menus and small on-screen images to manage your files and disks without typing MS-DOS commands or file names at the system prompt. With the Shell, you can see your directory structure outlined onscreen, and you can navigate from one directory to another with the keyboard or a mouse. You can also set up your application programs so you can run them by selecting from a list. If you have the Shell for version 5 or later of MS-DOS, you can even switch among multiple programs without having to quit one before you start another.

This program is called a *shell* because it surrounds MS-DOS. When you use it, you see the Shell and its menus rather than MS-DOS or its system prompt. The Shell even adds some capabilities that older versions of MS-DOS don't offer. For example, you can rename a directory, move one or more files without having to copy and erase them, and view the contents of two disks or directories at the same time.

This chapter gives you a quick tour of the Shell for versions 5 and later. It shows you how to use the Shell's major capabilities with either the keyboard or the mouse. If you have version 4 of MS-DOS, your Shell is visually similar but differs in some significant respects. You can use this chapter as a general guide to the Shell, but refer to your documentation and to online Help in your Shell for specifics. If you have version 6.2 or later of MS-DOS and would like to use the Shell, you can order the Supplemental Disk. (See the manual that came with MS-DOS.)

PREVIOUS NEXT

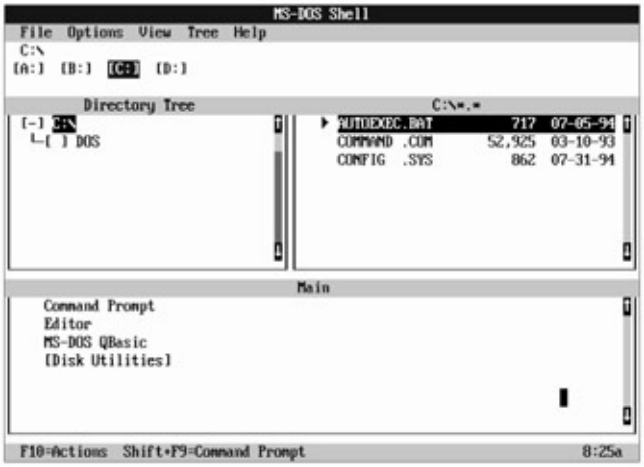
Starting the Shell

MS-DOS can be installed so that it runs the Shell whenever you start or restart your computer, or so that it waits until you give the command to run the Shell. If MS-DOS runs the Shell automatically, the Shell appears as soon as MS-DOS finishes its startup routine. If MS-DOS does not run the Shell automatically, you see the normal MS-DOS startup, ending with the system prompt. In this case, you start the Shell by typing its name:

```
C:\>dosshell
```

Note Be sure that Microsoft Windows is not running before you start the Shell.

When you start the Shell, the program begins by reading the name of each directory and file from the disk in the current drive. If the disk contains many directories and files, the Shell reports on its progress in a box headed *Reading Disk Information.* The Shell then displays its opening screen, which looks like the illustration below. For simplicity, the illustration shows only the files most likely to be in the root directory of a hard disk. You might see others.



Note If your opening Shell screen doesn't show areas headed *Directory Tree* and *Main,* you can change the display by choosing Program/File Lists from the View menu at the top of the screen. If you don't know how to do this, refer to the illustrations and disregard your display for a few minutes.

The Shell operates either in *text mode,* as shown in the preceding illustration, or in *graphics mode.* Text mode uses text characters only—no pictures—to create the display you see. Graphics mode, which you can use on EGA, VGA, and similar displays, combines text characters and some graphic images. Figure 11-1 on the next page shows how the screen in the preceding illustration looks in graphics mode. Notice that the mouse pointer is arrow-shaped instead of rectangular and that small images of disk drives and file folders replace the square brackets used in text mode.

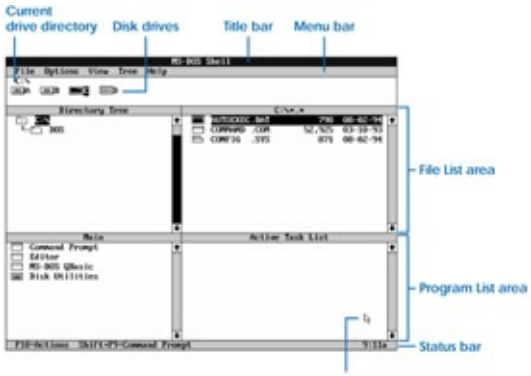


Figure 11-1: Parts of the MS-DOS Shell window in versions 5 and later.

The rest of this chapter shows screens in graphics mode. If you would like to switch to graphics mode, press the Alt key, then press the letter O, and then press the letter D. When a box labeled *Screen Display Mode* appears, press the Down arrow key until the dark highlight rests on the choice that begins with the word *Graphics.* Press Enter, and your screen will change to graphics mode.

Parts of the Shell Window

The opening Shell screen shows a large *window* containing a number of different elements. This window also shows two major parts of the Shell, the *File List* at the top and the *Program List* below it. From top to bottom, the principal elements in the Shell's opening window are listed and labeled in Figure 11-1.

- The top line of the screen (title bar) identifies the MS-DOS Shell.
- The second line (menu bar) displays the names of the available menus: File, Options, View, Tree, and Help. You choose commands from these menus.
- The next two lines identify the current drive and directory (probably C:\, as shown earlier on page 243) and show the disk drives attached to your system.
- The next one-third to one-half of the window comprises the File List area, which is divided vertically into two smaller sections.

  The left-hand portion, titled *Directory Tree,* shows all the directories on the current disk, arranged with subdirectories indented to the right and connected by lines that show the directory structure. This display is like the output of the Tree command. A minus sign inside the symbol that represents a directory (for example, the symbol to the left of C:\) means the directory contains one or more subdirectories that are currently displayed in the directory tree. A plus sign inside the symbol that represents a directory means that the directory contains one or more subdirectories that are not currently displayed.

  The right-hand portion lists the files in the current drive and the current directory (C:*.* in the illustration on page 243). Like a directory listing, these entries list the name, size, and date of creation or last change of each file in the selected directory.

- The lower section of the Shell window, titled *Main,* is the Program List area. It displays a list of programs you can run. Until you add your own programs to this window, it contains at least two basic choices, Command Prompt and Disk Utilities.

- Finally, the bottom line of the screen (status line) shows you the time of day and the effect of using two shortcut keystrokes, F10 and Shift-F9 (produced by holding down either Shift key and pressing F9). The F10 key activates the menus in the menu bar; Shift-F9 lets you temporarily leave the Shell and work at the MS-DOS system prompt. (To return to the MS-DOS Shell from the system prompt, type *exit* and press Enter.)

Using the Keyboard and the Mouse

Before you begin to explore the Shell, it's useful to know how to get around in it with a mouse or with the keyboard.

If you have a mouse, you already know that using one is easy and relatively standard among different application programs. In the Shell, maneuvering is as simple as you'd expect: Just roll the mouse on your desktop until the pointer is wherever you want it to be on the screen. Once you've pointed to the item or to the part of the window you want, you can select it by clicking the left mouse button. (The right button is inactive in the Shell.) For certain tasks, you double-click on an item by pressing the left button twice in rapid succession. In addition, you sometimes drag an item, as when you move a file from one directory to another, by positioning the mouse pointer on the desired file name, then pressing and holding the left button and moving the mouse pointer to a new location before releasing the button.

With the keyboard, you use the Tab key to move from one area of the Shell to another—for example, to move from the list of drives to the Directory Tree and then to the list of files in the current directory. To reverse direction, press Shift-Tab.

Figure 11-2 on the next page describes basic techniques you can use to perform the most common Shell operations with the keyboard or a mouse.

Keyboard	Mouse
To select a menu	
Press either Alt or F10 to activate the menu bar; press the key letter of the menu name (underlined in graphics mode, highlighted in text mode).	Click on the menu name.
To select a menu command	
Press the key letter of the command you want (underlined in the command name).	Click on the command.
To select a file	

Highlight the file name by using the arrow keys.	Click on the file name.
To choose (carry out) a command	
Highlight the command name by using the arrow keys, then press the Enter key.	Double-click on the command name.

Figure 11-2: Basic keyboard and mouse techniques.

If you're a keyboard user who is unfamiliar with menu-based programs, don't be concerned if the techniques don't have much meaning as yet; they soon will. For both mouse and keyboard, the hands-on examples in this chapter also help you along when necessary.

Note For the sake of standardization, this chapter assumes a computer with a hard disk (drive C) and at least one subdirectory (C:\DOS). If your system differs, the examples are still appropriate but might not work exactly as described.

PREVIOUS NEXT

Help Is a Keystroke Away

The Shell includes on-screen help information that can guide you through most of its features. You can request either general information or direct help with a specific situation. General help is a good way of finding out about the Shell and how it works—which keys you use, what procedures are available, and so on. Direct help provides details about whatever task you're working on at the time you request help. To request general help, you use the Help menu at the top of the screen. To request direct help, you choose the item or command you want and then press F1. You'll try both types of help in the following examples.

Start with general help by choosing the Help menu using one of the methods described below:

- With the keyboard, press Alt or F10 to activate the menus, and then press H.
- With the mouse, click on Help.

The Help menu opens, showing a list of choices with Index highlighted. Press Enter, and the Shell displays a window headed *MS-DOS Shell Help*:

The top line of the Help window contains the title of the screen. The main part of the Help window, titled *MS-DOS Shell Help Index,* describes what you can do next. The bottom line of the Help window—no matter what Help information is displayed—offers you five choices:

- *Close* ends Help, removes the Help window, and returns you to whatever you were doing.
- *Back* returns to the previous Help display if you've moved from one topic to another within Help.
- *Keys* displays a list of topics related to using the keyboard with the Shell.
- *Index* displays the Help index you're looking at now.
- *Help* displays help on Help.

Right now, notice that the box labeled *MS-DOS Shell Help Index* starts with a few lines that tell you how to choose an index topic. Immediately below this is the heading *Keyboard Help,* which is followed by a group of topics beginning with *General MS-DOS Shell Keys.* The index breaks Help information into broad categories with related topics grouped under them. To see what help is available, press the PgDn key a few times, stopping when you see the heading *MS-DOS SHELL BASICS HELP* and the topic *Welcome to MS-DOS Shell.* Now try requesting some help using one of the two methods listed below:

- With the keyboard, press the Tab key as many times as necessary to move the highlight or the small arrow at the left edge of the window to *Welcome to MS-DOS Shell*; press Enter.
- With the mouse, double-click on *Welcome to MS-DOS Shell.*

The title at the top of the window changes to *Welcome to MS-DOS Shell,* and the old display is quickly replaced by a description of the Shell. Press the PgDn key several times. At the end of the description, you see the words *Next topic* followed by *Scroll Bars.* Press the Tab key to select this new topic and press Enter, or double-click on it with the mouse. The window changes again, this time to one that tells you how to use the scroll bar the Shell displays along the right edge of certain windows.

Instead of moving deeper into Help, now try moving back the way you came. Choose the button marked *Back* at the bottom of the window. Either press the Tab key until the mouse pointer is on *Back* and press Enter, or click on *Back* with the mouse. The Help display changes to the previous screenful of information, *Welcome to MS-DOS Shell.* Choose *Back* again, and you see the original index screen.

If you want, experiment with other Help topics or with the other buttons at the bottom of the window. When you're ready to stop, either press Esc or use the Close button by tabbing to it and pressing Enter or by clicking on the button with the mouse.

Before leaving the subject of Help entirely, try requesting some direct help. First, choose the File menu (press Alt-F or click on File). Notice that the Open command is highlighted. Ask Help what this command does. Press the F1 key, and a Help window appears, this one telling you not only about the File Open command, but also referring you to additional information if you need it. Press Esc to close the Help window.

If you plan to use the Shell frequently, consider browsing through Help as you just did. It's a good way to learn to use the Shell. And remember the F1 key; it's all you need to request help on a particular task.

The File List

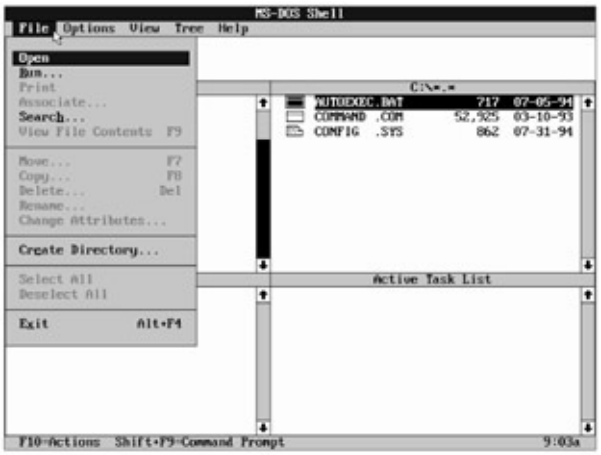
If you're using the keyboard to operate the Shell, you use the Tab key to activate the screen area you want to work in. If you're using a mouse, clicking in an area is enough to activate that portion of the window.

The current drive—probably drive C—should be highlighted. To prepare for the next examples, select the Directory Tree by pressing the Tab key or by clicking the mouse anywhere in that area. If the highlight is on the root directory (for example, C:\), use the Down arrow key or the mouse to highlight a different directory.

Note If you're using the Shell with a monochrome display and you see only two colors, one dark and one light, you can make your screen more closely match the shading in the illustrations in this chapter by changing to four-color monochrome. To do this, press the Alt key and then press the letter O twice. When a box labeled *Color Scheme* appears, press the Down arrow key until the dark highlight is on *Monochrome-4 Colors.* Press Enter, and your screen will change to shades of light and dark.

The File Menu

The Shell menus are lists of choices. Although they are menus like those in a restaurant, they show you both what items you can choose and whether the Shell will ask for more information when you make a choice. To see this, select the File menu again. The screen should look like this:



Notice that the menu items are displayed in both light (grayed) and dark characters. The Shell uses grayed characters to let you know which menu options you can't choose because they're not appropriate for what you're doing at the time.

Right now, the File menu tells you that you can open a file (to work with), run a program file, search for a file, delete (careful!) or rename the current directory, create a directory, or exit the Shell. Until you select one or more files, these are the only actions you can perform from the File menu. You can't print, for example, because you haven't selected a file to print.

The ellipsis (...) following certain choices, such as Run, means that the Shell needs a little more information before it can carry out the command. When you see an ellipsis following a command name, you can expect the Shell to prompt you for more information before it carries out the command. You'll see how this works a bit later. Keys and key combinations, such as *F8* and *Alt-F4* along the right edge of the menu, represent shortcut keys you can press instead of using the menu to choose the commands they carry out.

Press Esc to close the File menu.

Selecting a Directory

The Directory Tree outlines the directory structure of the disk in the current drive. If it's not already highlighted, select your \DOS directory. There are faster ways to move through a list, but for now, press an arrow key until DOS or the name of your MS-DOS directory is highlighted, or click on DOS (press the PgDn key, if necessary, to bring the directory into view). The area to the right should show the MS-DOS files. (If you don't have a hard disk, substitute the root directory of your startup disk for the \DOS directory unless the instructions here tell you otherwise.) Select the files area to the right of the Directory Tree by pressing the Tab key or by clicking anywhere in that portion of the screen.

Scrolling Through the Directories

The list of files in \DOS is probably too long to fit on the screen, so to see it all you need to *scroll* through the directory. Using the keyboard, you can scroll through this or any other long list of files or directories in any of several ways:

- By pressing the Down (or Up) arrow key to move a line at a time
- By pressing the PgDn (or PgUp) key to move a screen at a time
- By pressing a letter key to move directly to the first entry that begins with that letter
- By pressing the Home key to move to the first entry in the list or by pressing the End key to move to the last entry

If you're using a mouse, the Shell includes a mechanism that makes scrolling even easier: the *scroll bar*, for which you saw Help earlier. The scroll bar is the vertical bar at the right margin of each window. It has an arrow at the top and at the bottom and contains a shaded box. Clicking on one of the arrows moves a list one line in the direction of the arrow. Clicking in the blank part of the scroll bar moves a screenful, and dragging the shaded box to any position between the top and bottom moves you the same relative distance in the list.

To try scrolling through a list, use one of the techniques described in the list above to view different parts of the \DOS directory.

Selecting Files

In order to work with your files, both MS-DOS and the Shell require you to indicate which files you want to affect with any particular command. From the MS-DOS command prompt, you type a file name, with or without wildcards, to specify files. Within the Shell, you mark the names of the files you want to affect.

With the mouse, a file is selected as soon as you point to it and click the left button. With the keyboard, you tab to the list of file names and use the arrow keys to highlight the file name you want. In graphics mode, the symbol to the left of the file name is highlighted as soon as you select the file. In text mode, an arrow and a triangle appear before the file name. Your next file-oriented action will affect that particular file.

With a file selected, open the File menu again. Notice that all of the choices except Create Directory are now available. That's because all the other commands on the menu can be applied to a selected file. Press Esc to close the menu.

You can select more than one file at a time—for copying, perhaps, or moving to another directory:

- To select a series of files with the keyboard, move the highlight to the name of the first file, press and hold the Shift key, and extend the selection with the arrow keys.
- To select a series of files with the mouse, click on the first file name, press and hold the Shift key, and click on the last file in the group you want to select.
- To use the keyboard to select files scattered throughout a list, press and release Shift and the F8 key. When you do this, the word *ADD* appears near the right edge of the status line at the bottom of the screen. Using the arrow keys to highlight file names, press the Spacebar to select each file you want. To switch back to normal selection, press Shift-F8 again.
- To use the mouse to select files scattered throughout a list, hold down the Ctrl key instead of the Shift key as you click on each file name you want.

If you've practiced selecting groups of files, reduce the selection to one file by choosing Deselect All from the File menu. (Press Alt, then F, then L, or click on File and then on Deselect All.)

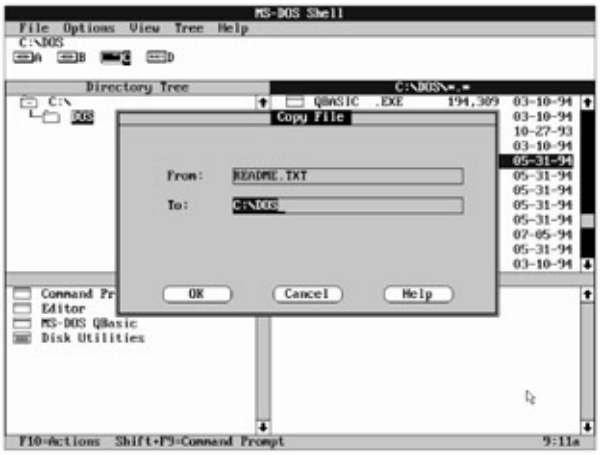
The next few topics show you some of the File menu items; you can explore the rest. Most of the items are self-explanatory, but a few might not seem obvious. *Associate,* for example, lets you tell the Shell to run an application program when you choose a file with a particular file extension. You might associate your word processing program, for example, with files whose extension is DOC. If you need explanations you don't find here, remember that Help is always available.

Copying a File

To see how the Shell prompts you for additional information, move to the files area and

highlight README.TXT (or any other TXT file) from your \DOS directory. Now press F8 or select Copy from the File menu.

Your screen should look like the graphic you see at the top of the next page.



The Shell displays a box titled *Copy File* in the middle of the screen. The box, called a *dialog box,* is what the Shell uses to ask for any additional information it needs in order to carry out a menu command. As you can see, the Shell makes some assumptions. Here, for example, the From field is already filled in with README.TXT, the file you selected to copy.

Because you can include a drive letter and path name when you copy files, the Shell tries to help here too by suggesting the current directory, placing the cursor at the end, and highlighting its guess. To add to this drive and path, you would press the Right arrow or the End key and start typing. To replace the suggested drive and path, you would simply start typing; the characters you typed would replace what the Shell displayed.

You're going to copy README.TXT to a file named README.NEW in the current directory, so type *readme.new* and press Enter. The dialog box disappears, and README.NEW is added to the list of files. It's as if you had typed the command *copy readme.txt readme.new* at the system prompt.

Moving a file with the Move command works just like copying a file, but the Shell deletes the original file after making the new copy. If you used the Shell to move README.NEW from the current directory (\DOS) to the root directory, for example, the result would be the same as if you had typed in MS-DOS version 6 *move readme.new* \ or, in earlier versions of MS-DOS, *copy readme.new* \ and *del readme.new.*

Note If you're using a mouse, you can move a file easily by selecting it and, holding down the left mouse button, dragging the file to the name of a different directory. When you do this, the Shell asks if you're sure you want to move the file. If you are, click on Yes or press Enter. The file is moved. No need to copy and delete; no need even to choose the Move command from the File menu.

Viewing a File

Viewing a file is a quick way to see what's in it; you can't change the file—you can only look at it. DOSSHELL.INI is a text file created when you install MS-DOS version 4, 5, or 6.0. Among other things, this file tells the Shell about your system and the way it should look when it starts. Select DOSSHELL.INI from the list of files, then select View File Contents from the File menu. The screen should look like this:

The top line identifies the file (DOSSHELL.INI), the next line lists the menus available (Display, View, Help), and the third line tells you which keys you can press to scroll through the file. (If you're using a mouse, you can also scroll by clicking on the PgUp, PgDn, and arrow symbols on this third line of the screen.) The main part of the screen shows part of DOSSHELL.INI.

The Shell can display files in either of two modes, ASCII or hexadecimal. As you see it now, the file is displayed as an *ASCII file,* so called because it is translated into readable text characters according to the American Standard Code for Information Interchange, or ASCII. In hexadecimal format, the characters in a file are translated into the mathematical language of computers—specifically, the base-16 numbering system often used by programmers. To see the hexadecimal view, press F9.

It's certainly different. Each line of the hexadecimal view shows 16 characters. The first column of numbers at the left margin is the address—in hexadecimal, of course—of the beginning of each line, starting at 0 at the beginning of the file. Each of the middle four columns of numbers is the hexadecimal representation of four characters, two digits per character. The 16 columns at the right show the ASCII characters that correspond to the hexadecimal digits in the line.

Press F9 again, and the display returns to the ASCII view. Press Esc to return to the File List display.

Deleting a File

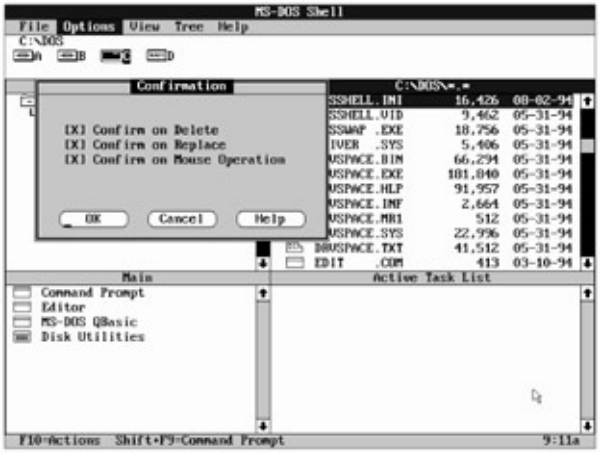
Using the Shell, you can delete one file, several files, all files in a directory, or any directory except the root (\) simply by selecting the file, files, or empty directory you want to delete, and then pressing Del or selecting Delete from the File menu. Because it's so easy to

change directories and select files in the Shell, using the Shell to delete files you no longer need is a snap compared to typing comparable commands at the MS-DOS system prompt.

Simply because the Shell makes it so quick and easy to delete files, however, it's a good idea to give yourself some protection by making sure that the Shell asks you to confirm each deletion. With the Shell, in fact, you can protect yourself not only from inadvertently deleting a file with the Delete command, but also against inadvertently replacing a file by copying another in its place.

Protecting Yourself

To protect yourself against inadvertently deleting a file, select the Options menu, then choose Confirmation. The Shell displays the dialog box shown here:



The brackets to the left of each option contain an X if the option is active. Selecting an active option turns it off; selecting an inactive option turns it on.

Confirm On Delete determines whether the Shell prompts you for confirmation before deleting a file. Confirm On Replace protects you against a less obvious way of losing a file: It determines whether the Shell prompts you for confirmation before copying over an existing file. Confirm On Mouse Operation determines whether the Shell warns you before carrying out a mouse operation that moves or copies a file.

All three options should be on (contain an X between the brackets). If one is off, tab to it and press the Spacebar to turn it on. When you're finished, press Enter to close the dialog box.

Now to delete a file. Some time ago you created README.NEW by copying README.TXT. You don't need README.NEW, so select it (don't select README.TXT by mistake), then press Del or choose Delete from the File menu. The Shell displays a dialog box headed *Delete File Confirmation.*

This is how the Shell gives you a second chance. The dialog box shows the name of the file to be deleted, and the buttons at the bottom ask you to choose Yes, No, or Cancel. Press Enter to choose Yes. The Shell deletes README.NEW and returns you to the File List

display.

You can delete an entire directory and its contents in two quick steps with the Shell. Because the Shell refuses to delete a directory that contains files, you first select the directory in the Directory Tree, and then select and delete all files in it with the Select All and Delete commands on the File menu. Once the files are gone, you can use Delete again, this time to remove the directory itself. Because it's so easy, make sure you do, indeed, want to erase both the directory and everything in it—a few valuable files might be hidden in a welter of unnecessary ones.

Whether or not you have set Confirm On Delete, you must delete all files in a directory before you can delete the directory, just as when you use the MS-DOS Remove Directory command. Likewise, you cannot delete any directory that contains subdirectories, nor can you delete (or rename) the root directory, because each disk must contain this directory.

Using the Options and View Menus

You just used the Confirmation command from the Options menu to protect against inadvertently deleting or copying over a file. Other choices on the Options and View menus let you display file names in a particular order; display information about the selected file, directory, and disk; display the files in two directories at once; display all the files on the selected disk; display only the File List; or display the Program List and File List at the same time, as the Shell is doing now.

Controlling the Order in Which File Names Are Displayed

When you first start the Shell, the File List displays the file names in alphabetic order. You can change this order to display the files arranged (sorted) by extension, date, size, or even the order in which they are stored on the disk. To display the files arranged by extension, choose File Display Options from the Options menu. The Shell displays the File Display Options dialog box, which you can see at the top of the next page.

In addition to specifying the order in which files are to be displayed, this dialog box lets you tell the Shell whether to display hidden and system files and whether to display files in

ascending (A to Z) or descending (Z to A) order.

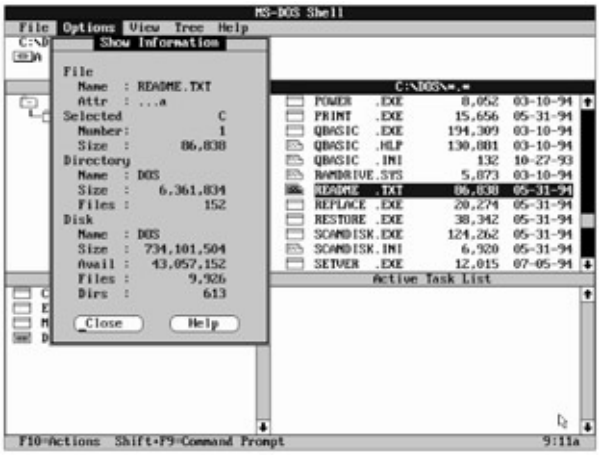
To change Display Hidden/System Files or Descending Order with the keyboard, you tab to the option and press the Spacebar. With the mouse, you simply click anywhere on the option. Either way, if the option is turned on, an X appears in the brackets; if the option is off, as it should be now, the brackets are empty.

The different ways you can sort files are in the list at the right side of the dialog box. These options are mutually exclusive, so only one can be active at any time. The active choice is indicated by a dark dot inside the circle (a dot between parentheses in text mode).

To try changing one of the settings, press Tab to move to the list and press the Down arrow key once to select *Extension.* Then press Enter or click on OK to close the dialog box. When the full window reappears, the files are listed alphabetically by extension—BAS files before COM files, and so on—rather than by name as they were before. Finding files arranged alphabetically by name is generally easier than if they're arranged by extension, so use the File Display Options command again and return to sorting by name.

Displaying File, Directory, and Disk Information

The Show Information command on the Options menu displays information about the currently selected file, directory, and disk. Highlight README.TXT (or any other TXT file. The file-specific data will be different, but the layout and interpretation will be the same), then choose the Show Information command. The screen should look like this:



There's nothing in this window for you to type or select; it simply shows you the following information:

- *File* shows the name and attributes of the selected file.
- *Selected* shows the letter of the selected disk drive and the number and total size of all selected files. If you're displaying directories from two disks, it shows both disk letters and the number and total size of all selected files on both disks.
- *Directory* shows the name of the directory that contains the file named under *File,*

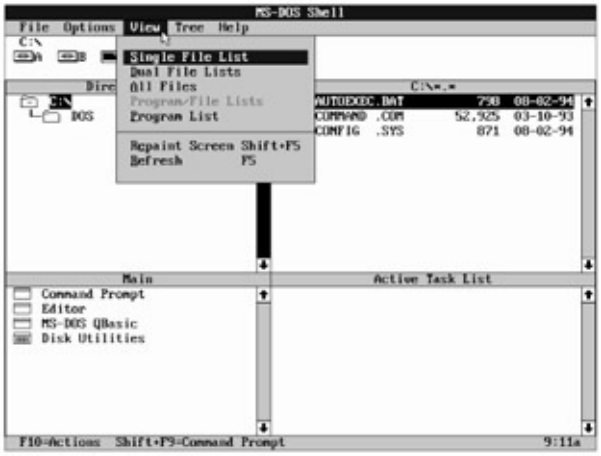
the number of bytes taken up by all the files in the directory, and the number of files in the directory. Knowing the size of a directory can be quite useful in certain operations, such as copying a directory from a hard disk to a floppy disk.

- *Disk* shows the name (volume label), capacity, available space, number of files, and number of directories on the disk that contains *File.*

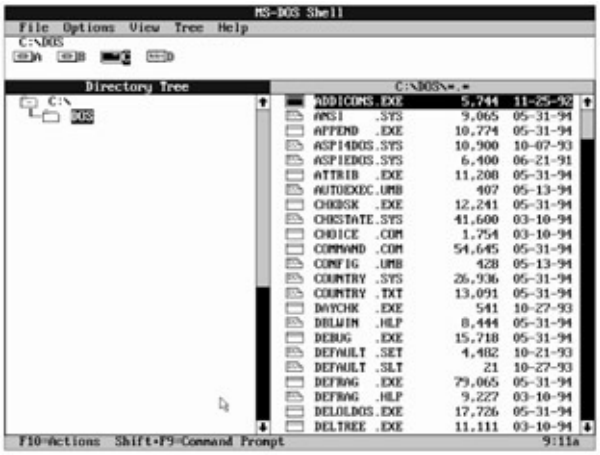
You can press Esc or click on Close to remove the Show Information window from the screen.

Changing the View

The Shell lets you display the File List, the Program List, or both (as you're doing now). You choose what you want to see with the View menu. To change the view, open the View menu. Your screen should look like the screen at the top of the next page.



The first choice (Single File List) is highlighted; the item Program/File Lists is grayed because that's what the Shell is displaying now, so you can't choose it. Press Enter to select the highlighted choice. Now the File List takes up the entire window:



Displaying Two Directories at Once

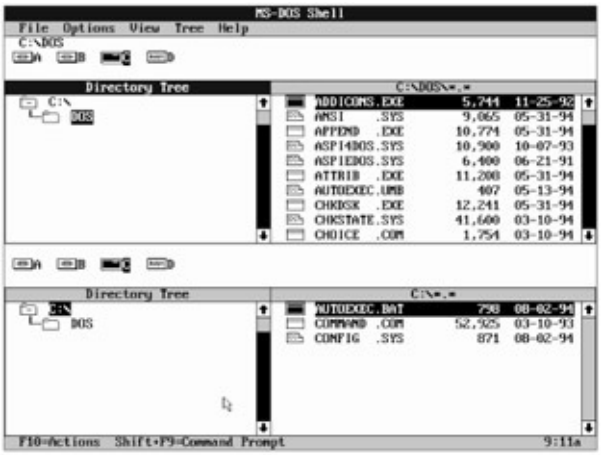
The Shell lets you display the files in two directories, on the same or on different drives. To

see this, choose Dual File Lists from the View menu. The Shell divides the middle portion of the screen horizontally into two areas, one above the other. The areas are identical: Each includes the disk-drive icons, the Directory Tree, and the names of the files in the selected directory.

In the example shown below, the upper file area lists the names of the files in the \DOS directory on drive C, and the lower file area lists the names of the files in the root directory.

When you're viewing a dual file list, you can select files in either list for copying, deleting, moving, and so on. You can select files in more than one directory by displaying two directories and activating Select Across Directories on the Options menu, and you can even view directories on two disks by selecting different drives for the two areas.

Choose the Single File List command again to return the display to one directory.



Finding That File

The bigger the disk, the more you can store on it. That's the good part. The more you store, the harder it can be to find a particular file. That's the bad part. Eventually, everyone who uses a computer for a while confronts the same frustrating question: Where did I save that file?

If you use the Shell in MS-DOS version 5 or later, that question becomes, if not negligible, at least less irritating. Rather than search through each directory, you can tell the Shell to do the work with the Search command on the File menu. Suppose, for example, that you wrote a bid proposal last year, and you think you saved it as PROPOSAL.DOC. You're pretty sure the proposal is still on your hard disk somewhere, and now you'd like to resurrect the document, change it a bit, and submit it as a different bid. With one command, you can tell the Shell to go look for what you want. When you choose Search from the File menu, you see a dialog box like the one on the next page.

Notice that the Shell initially proposes to search the entire disk for all files (*.*). That approach can be useful in some situations, particularly because the Shell's report on what it finds includes the path to each file. In this example, however, you're interested in a particular file name and extension, so you'd type *proposal.doc* in the field labeled *Search For.* (To find

a set of files, you could use wildcard characters—type, for example, **.doc* to see a list of all files on the disk with the extension DOC.) To search the entire disk, leave the X next to the *Search entire disk* option. To limit the search to the current directory and its subdirectories, you turn off this option.

To try this command, tell the Shell to search the entire disk for the file UNDELETE.EXE. When you press Enter or click on OK to start the search, the Shell might become still for a short time. Don't be concerned; it's working, just as MS-DOS is working when you use Check Disk or a similar command. When the search is complete, a new window appears, displaying a list of all files that match the path (if any), file name, and extension you specified. Press Esc to clear the window.

Displaying All the File Names on a Disk

Whereas the Search command gives you a way to find one or more needles in a haystack, the All Files command on the View menu lets you see the whole pile of hay—every file in every subdirectory on a disk. Choose the All Files command from the View menu.

When you choose the All Files command, the Shell combines two types of displays in one. At the right side of the screen, it lists the names of all files on the disk, showing the same directory information it normally shows in the File List area. Along the left side of the screen, it provides the same information you see when you choose the Show Information command from the Options menu. This combined display not only lets you see the directory information for each file on the disk, it also provides you with the directory name, attributes, directory size, and other details displayed by the Show Information command. As you select different file names, the Shell updates the information on the left side of the screen to tell you about the currently selected file.

Return to displaying the directory tree by choosing Program/File Lists from the View menu.

Viewing the Directory Tree

When you use directories and subdirectories to organize your files, you can create a directory tree with many levels. The Tree menu lets you control how many directory levels

are displayed in the Directory Tree area of the File List. If you want, you can choose to see only the root directory of a disk. On the other hand, you can tell the Shell to outline the entire tree, including every subdirectory of every directory on the disk. Between these extremes, you can selectively collapse and expand single directories or subdirectories according to the level of detail that interests you at the time.

You can use the various commands on the Tree menu to expand and collapse directories, but the Shell also recognizes certain keys that make your work much easier. Here are the commands on the Tree menu and what they do, as well as the keys you use:

- *Expand One Level* expands the selected directory to display the next directory level. Speed key: + (plus).
- *Expand Branch* expands the selected directory to show all directory levels under it. Speed key: * (asterisk).
- *Expand All* expands the entire directory tree to show all subdirectories under all directories. Speed key: Ctrl-* (hold down Ctrl and press asterisk).
- *Collapse Branch* collapses the selected directory. Speed key: - (minus).

The Tree menu does not include a command or a key combination that collapses the entire directory tree once you've expanded all levels. You don't have to collapse the branches one by one, however. To return to the Shell's normal display of the directory tree, select the root directory and press the minus key to collapse the entire tree. Once that's done, press the plus key to expand the root directory one level.

Beyond this description, all you need is a little bit of experimentation with your own directory tree in order to become comfortable with the Shell's ability to show you different levels of your directory structure from the root directory to the entire tree. Notice, as you select different directories, that the file list at the right side of the File List area changes to show the files in the selected directory.

Although the Shell isn't suited for everything you'll do with MS-DOS, you can see that it makes many routine file-management tasks much simpler and quicker.

The next part of this chapter deals with a different side of the Shell—the Program List. If you want to stop temporarily, you can leave the Shell and return to MS-DOS by pressing F3 or choosing Exit from the File menu.

The Program List

If you think of the File List area as the part of the Shell window that provides information, you can think of the Program List area as the part that provides functionality. The Program List area can contain two different types of items: programs and program groups. Although both represent applications and other programs you can run directly from the Shell, programs are stand-alone items that run as soon as you activate them. Program groups are umbrella items that include several programs. (In graphics mode, a program group has a different icon from a program. In text mode, a program group is enclosed in square brackets [].)

When you start the Shell, the Program List area displays a list of programs and the name of one program group. Typically, the programs and the program group are the following:

- *Command Prompt,* which temporarily stops the Shell and displays the MS-DOS system prompt so that you can type MS-DOS commands. You return to the Shell by typing *exit.*
- *Editor*, which starts the versions 5 and later MS-DOS Editor program that you meet briefly here and learn about in more detail in the next chapter.
- *MS-DOS QBasic,* which starts the QBasic program, a version of the Microsoft QuickBasic programming language.
- *Disk Utilities,* which is a program group that displays another list of choices when you select it. In this case, Disk Utilities groups a set of commonly used MS-DOS disk and file operations.

You'll experiment with these and add programs and a group of your own. Before you start, however, prepare for one of the later examples by copying one file to a floppy disk in drive A. Put a formatted floppy disk in drive A, then highlight the file named DOSSHELL.INI, which you viewed earlier, in your \DOS directory. Now press F8 or choose Copy from the File menu to copy DOSSHELL.INI to A:\. You won't need the File List area for now, so clear the window a bit by choosing Program List from the View menu.

Starting a Program

To start a program or display the items in a program group, you simply choose from the Program List. For example, Command Prompt should be highlighted now. Although you wouldn't necessarily consider this the name of a program, it does in fact start a program that causes MS-DOS to temporarily leave the Shell and display the system prompt. Press Enter or double-click on Command Prompt, and in a few moments the screen clears, MS-DOS displays an opening message, and you see the MS-DOS prompt.

To return to the Shell, type this:

```
C:\DOS>exit
```

Unless you're running a program that has its own quit command, this is the command you use to return to the Shell.

Switching Among Programs

Have you ever, while using one program (a word processor, for example), needed to use another program (a spreadsheet, say, or a graphics program)? If so, you had to leave the first program, start the second, finish with the second program, leave it, and start the first program again. Sometimes this can happen fairly often, making you spend too much time just starting and stopping programs.

The Shell includes a feature called the Task Swapper, which lets you start one or more programs and then switch from the Shell to each program with just a few keystrokes. You'll test this with the MS-DOS Editor and QBasic, both of which come with MS-DOS.

Note If you're interested in trying the Task Swapper with your own application programs, you can follow the instructions given here by substituting the names of your application programs when you're told to activate the Editor and QBasic. A word processing program, such as Microsoft Word, will do. But try not to use a memory-resident ("pop-up") program, such as Doskey; you might see a message telling you to quit the program before returning to the Shell.

To start the Task Swapper, open the Options menu. The fifth item is Enable Task Swapper, and this choice is turned either on or off. If it's on, it's preceded by a dot. If it's off, select it and press Enter to turn it on, otherwise press Esc to close the Options menu.

Now the Program List area should be divided into two parts: The left half should be titled *Main*; the right half should be titled *Active Task List.* The right side displays the names of all programs that you have started and left running; you haven't left any running, so the list should be blank.

Start the Editor by choosing Editor from the Main window. When the Editor dialog box asks you which file to edit, type *edfile* and press Enter or click on OK. When the Editor starts, it displays a blank window. So that your applications will be easy to recognize, type the following line:

This is my Editor file.

Now return to the Shell without stopping the Editor. To do this, press Ctrl-Esc. In a few moments the Shell displays the Program List screen again. This time, Editor appears in the Active Task List to tell you that you can rejoin the Editor any time you like. To do so, you simply select it from the Active Task List.

Start a second program by choosing MS-DOS QBasic from the Main window. When the program asks you the name of the file to edit, type *basfile* and press Enter or click on OK. When QBasic starts, type this:

This is my QBasic file.

Again, press Ctrl-Esc to return to the Shell. Now the Active Task List shows two entries, MS-DOS QBasic and Editor.

To switch to the Editor, select it from the Active Task List. There's the Editor again, just as when you left it. Now suppose you were using the Editor and remembered that you wanted to check something with QBasic. Hold down the Alt key and press Tab until the title bar at the top of the screen displays *MS-DOS QBasic.* Release the Alt key; you're back to your other program. Press Ctrl-Esc to return to the Shell.

You can switch among your programs and the Shell at will by pressing Alt-Tab. When you press Alt-Tab, the Shell cycles through the name of each active program (including itself) in round-robin fashion. Unless you have started several programs, Alt-Tab is the only key combination you need to switch back and forth. If you'd like to see what other key combinations you can use, choose Keyboard from the Help menu, then choose Active Task List Keys.

If you want to start a program but won't use it right away, you can add it to the Active Task List without ever leaving the Shell. Highlight Command Prompt, but press Shift-Enter instead of just Enter. Now the Active Task List should contain Command Prompt, MS-DOS QBasic, and MS-DOS Editor.

When you're through using a program, just end it as you normally would; the Shell displays the Program List screen again, but now the name of the program you left doesn't appear in the Active Task List.

To stop the three tasks you've started, do the following: Choose Command Prompt from the Active Task List, and type *exit* when MS-DOS displays the system prompt. When the Shell reappears, Command Prompt is gone from the list of tasks. Next, switch to QBasic, and choose Exit from the File menu. Choose No when QBasic asks if you want to save the sample file. When you return to the Shell, switch to the Editor, choose Exit from the File menu, and again choose No when asked if you want to save the file. This time when you return to the Shell, there are no entries in the Active Task List.

You'll find that switching among programs like this lets you make much better use of your time at the computer.

Selecting a Program Group

The symbol or brackets associated with Disk Utilities in the Main area tells you that selecting this item displays a different list of choices. To see the list, choose Disk Utilities.

The area of the screen is now titled *Disk Utilities*—the name of the program group you selected—rather than *Main,* and you see a different list of choices: Main (which returns you to the Main Program List screen), Disk Copy, Backup Fixed Disk, Restore Fixed Disk, Quick Format, Format, and Undelete. As you'll see shortly, you can customize this list of choices by adding your own program items and program groups. Choose Main to return to the Main screen.

When the Program List area is active, the menus aren't the same as when the File List is active. The Tree menu disappears because it isn't needed here. The Options and Help menus remain the same, and the View menu remains almost the same. (Refresh is not available from the Program List area.) The File menu, however, contains a number of different choices.

The Program List File Menu

Rather than offering choices that deal with files, the Program List's File menu lets you manage its list of program items and program groups. Display the File menu; the choices include these:

- *New,* which adds a new program item or program group to the Program List.
- *Open,* which starts the highlighted program item or displays the items in a selected program group. (It has the same effect as highlighting an item and pressing Enter.)
- *Copy,* which copies a program item to a program group you specify.
- *Delete,* which deletes a program item or a program group (from which you've already deleted all program items).
- *Properties,* which lets you specify the title, startup command, and other definitions of a program item or program group.
- *Reorder,* which lets you change the order in which the program items and program groups are displayed.
- *Run,* which lets you start any program, whether or not it is displayed in the list of program items and program groups.
- *Exit,* which leaves the Shell.

Adding a Program Item

Adding a program item to the Shell can be simple. You can even add a program that requires more than one command simply by separating the commands with semicolons.

Suppose you wanted to add an entry called Memory Check that would carry out the MS-DOS Mem command. Choose New from the File menu. The Shell displays a dialog box that gives you the choice of adding a Program Group or a Program Item. The Shell assumes you want to add a Program Item, which you do, so press Enter or click on OK.

Now the Shell displays the Add Program dialog box:

The Shell requires that you fill out only the Program Title and Commands fields to define a new entry. The Program Title field specifies how the Program List item will be displayed; type *Memory Check.* Press Tab to move to the Commands field, which specifies the command (or commands) to be carried out; type *mem* and press Enter. The Add Program dialog box disappears, and the item Memory Check is added to the list of programs.

Choose the new item by highlighting it and pressing Enter or by double-clicking on it; the screen clears, and in a few moments MS-DOS displays the report of the Mem command, followed by the message *Press any key to return to MS-DOS Shell.* Press any key to return to the Shell.

Changing a Program Item

You can change any of the characteristics of a program item—the title, commands to be carried out, and so forth. The Shell refers to these characteristics as *properties;* to change them, you highlight the program item to be changed and select Properties from the File menu. Highlight Memory Check, and then select Properties from the File menu. The Shell displays the Program Item Properties dialog box:

This dialog box contains the same fields as the Add Program dialog box. Because it lets you change an existing definition, however, fields to which values have been assigned—in this case, Program Title and Commands—are filled in.

To change one of the properties or add a new one, you simply enter the new value in the

appropriate field. Suppose, for example, you decide that you want the program item to be named Mem Command instead of Memory Check. You also want to assign it the password IQ.

Type *Mem Command* in the Program Title field, type *IQ* (notice the capitals) in the Password field, and press Enter to carry out the change. Try the command again. This time, a dialog box appears, asking for the password. Type *IQ*—be sure to type in capital letters, because the Shell can tell the difference—then press Enter. Once you've entered the password, the dialog box disappears, the screen clears, and the memory report is displayed. Your program item is the same command, but you've now given it a new definition.

Deleting a Program Item

This Mem Command example has served its purpose, so you'll delete it. Make sure that Mem Command is highlighted, and select Delete from the File menu. The Shell displays the Delete Item dialog box shown on the next page.

Like the Delete File dialog box, the Delete Item dialog box lets you confirm the name of the item to be deleted. Because you've assigned a password to Mem Command, the dialog box reminds you of the fact. The dialog box offers two choices, Delete and Do Not Delete. The Delete option is highlighted, and you are asked to choose OK, Cancel, or Help. You want to delete the item, so press Enter. The dialog box disappears, and the Shell continues to display the Main Program List screen, but Mem Command is gone.

Customizing the Program List

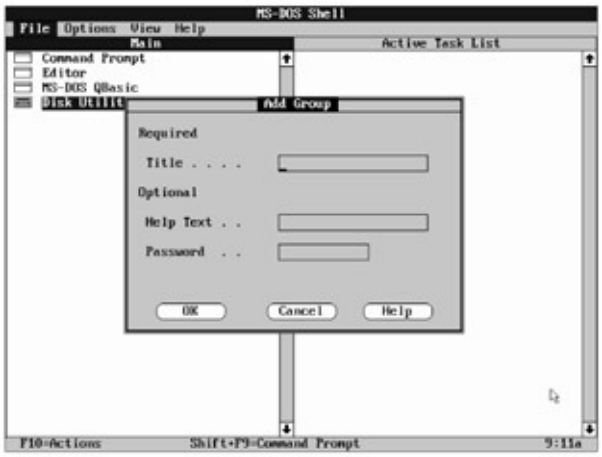
You can customize the list of program items and program groups that the Shell displays, tailoring the Shell to reflect the programs you use. You might add a program group named Word Processing, for example, to include your word processor and associated programs, or you might add a program group named Financial for your spreadsheet or accounting programs. Once you create the groups, you will add program items to them. The program items run the programs.

Adding a Program Group

Suppose you wanted to create a separate program group named Text Files that would let you edit, display, and print files from the Shell. You'll add the program group and then add the MS-DOS Editor to it.

Note If you want, you can try adding your word processor as the program item. Assuming that your word processor accepts a file name as part of the startup command, the following instructions should work for you. Just remember to use the command that starts your program instead of the Edit command that starts the MS-DOS Editor.

To create the program group, choose New from the File menu just as when you added the Memory Check program item. As before, the Shell displays the New Program Object dialog box. This time, however, you don't want to accept the Shell's choice of the Program Item option, so select Program Group instead and then press Enter (or click on OK). The Shell displays the Add Group dialog box:



As the dialog box shows, the only field you must complete is Title, which specifies the group name you want in the Program List area. If you wish, you can also specify the Help text that the Shell displays if F1 is pressed when the group name is highlighted, and (as before) you can designate a password that must be entered in order to use the group. For this program group, you'll enter a title and some help text.

First type *Text Files* in the Title field, then tab to the Help Text field. Type *This group lets you edit text files,* and press Enter to save these entries; the Shell displays the Program List area again, with the added group Text Files.

Now activate the Text Files group by highlighting it and pressing Enter or by double-clicking on it. The title line at the top of the display changes from *Main* to *Text Files,* and even though you haven't added any program items yet, the list contains one choice: Main. The Shell adds this program item for you so you can return to the Main Program List screen. (You can also return by pressing Esc.)

Specifying a Command Parameter

The first program item you'll add to the Text Files program group will run the MS-DOS Editor or the alternative you've chosen. (Yes, you can already run the Editor from the Main list, but it's used as an example here because most computer users have a text-handling program of some type.)

Choose the New command from the File menu. The Shell displays the New Program Object dialog box, just as it did when you added Memory Check and Text Files. Press Enter or click on OK to create a program item. The Shell displays the Add Program dialog box. Type *Text Editor* and tab to the Commands field.

When you specified the Memory command in the earlier example, you entered just the command name in the Commands field. A text editor, however, normally lets you specify the name of a file you want to work on. For example, if you include a file name with the command that starts the MS-DOS Editor, MS-DOS opens the file if it exists or creates the file if it doesn't exist; if you don't include a file name, MS-DOS starts the Editor without opening or creating a file.

To tell the Shell that you want a program item to accept one parameter, such as a file name, you include a percent sign followed by the numeral 1 (%1) in the Commands field. For the MS-DOS Editor, type *edit %1*. (For a different editor or a word processor, type the appropriate startup command. For example, type *word %1* if you are using Microsoft Word.) This process of creating a parameter is explained on the next page in "Designing Your Own Dialog Box." Now tab to the Startup Directory field.

Specifying a Path

If you routinely use directories in your work, you probably organize your data files in different directories, depending on what they contain or what type of application you use them with. Your word processing files, for example, might be categorized by type (letters, memos, reports), client, department, or in any of a number of other ways.

In setting up a program to run from the Shell, you can use the Startup Directory field in the Add Program dialog box to change to a particular drive or directory whenever you start the associated program. If you write a lot of letters, for example, you might change to your LETTERS directory when you start your word processor.

To see how the startup directory works, tell the Shell to use the root directory of drive A whenever it starts by typing *a:* in the Startup Directory field. Now tab to the Application Shortcut Key field.

Specifying a Shortcut Key

Just as some Shell menu selections have keyboard shortcuts—F8 for Copy in the File Menu of the File List, for example—you can specify a shortcut key combination for a program item or program group. The shortcut key must be Shift, Alt, or Ctrl, combined with another key on the keyboard. To make Alt-E the shortcut key for the Editor, press Alt plus the E key. *Alt-E* appears in the Application Shortcut Key field. Finally, tab to the Pause After Exit field.

Controlling the Pause Message

When you tried the Memory Check examples, you saw that the Shell displayed *Press any key to return to MS-DOS Shell* and waited to let you read the message and press a key. This pause isn't required, however, and you can control it with the Pause After Exit field. There's an X between the brackets now, to show that the option is turned on. Press the Spacebar (or click on the X) to turn the option off.

Chances are you won't want to keep anyone from using the Editor, so there's no need for a password. Besides, you've already seen how the password option works, so press Enter to tell the Shell you've finished with the dialog box. Instead of returning to the Program List, however, the Shell displays another dialog box titled Add Program.

Designing Your Own Dialog Box

The command you entered for the Editor was *edit %1,* telling the Shell that you wanted the command to accept one parameter (a file name) of your choice. The Shell will have to prompt for this parameter, and the dialog box you're looking at now lets you specify the contents of the dialog box that displays the prompt.

The cursor is at the beginning of the Window Title field; this field specifies the title displayed in the bar at the top of the dialog box (*Add Program* in the dialog box you're working with now). Type *File To Edit* and tab to the Program Information field.

You use the Program Information field to specify the instructional or explanatory text that appears below the title but still at the top of the dialog box. (In the box you're working in now, the text reads *Fill in information for %1 prompt dialog*.) Type *Enter the name of the file to be edited (or press Enter to start the Editor with no file).* Tab to the Prompt Message field.

The Prompt Message field specifies the text that appears to the left of the field to be filled in. (Here it's *Prompt Message* for the field you're working in now.) The Edit command parameter must be a file name, so type *File name:* and tab to the Default Parameters field.

The Default Parameters field lets you specify a value that the Shell fills in for the command parameter. (You'll see how this works shortly.) Type *fred,* then press Enter to complete the definition of the Text Editor program item. The Shell displays the Text Files program list again, this time with two items: Main and Text Editor.

Testing the New Program Item

To try out your new program item, check that the floppy disk with DOSSHELL.INI is in drive A. Highlight *Text Editor* and press Enter to start the program. First, the Shell displays the File To Edit dialog box that you just designed:

Check the title of the dialog box, the instruction line at the top, and the prompt to the left of the entry field. They should match the text you entered in the Window Title, Program Information, and Prompt Message fields a moment ago.

The entry field contains *fred,* the default parameter you specified in the Default Parameters field. If you were to press Enter now, the Shell would tell MS-DOS to run the Edit command, using FRED as the name of the file to edit. Because there is no such file, first MS-DOS would create it, and then the Editor would show you a blank screen.

Instead, request the DOSSHELL.INI file that you copied to the floppy disk in drive A earlier in the chapter. The Shell erases the default file name as soon as you start typing, so just type *dosshell.ini.*

Now press Enter. Drive A becomes active, and soon the screen fills with the lines of DOSSHELL.INI.

Right now you're just verifying that the Text Editor program item works, so select Exit from the Editor's File menu (or use the appropriate Quit command if you're using a different program). The Editor quits, and the Shell screen returns, showing your Text Files group and its two choices.

The Disk Utilities Program Group

You can learn still more about defining program items and groups by selecting a program item from the Disk Utilities group and seeing how the fields are filled out. Just don't change anything. For example, return to the Main screen, and select Format in the Disk Utilities group. The screen looks like this:

Notice that the Parameters field is filled in with *a:*, suggesting that the floppy disk in drive A should be formatted. Now press Esc (not Enter) to return to the Program List and, with Format still highlighted, choose Properties from the File menu to see how the Format program item is defined. The screen should look like this:

One parameter, %1, is specified in this dialog box. To see how this parameter is defined, press Enter or click on OK. The Shell displays the Program Item Properties dialog box for %1:

Compare the fields of this dialog box with the dialog box itself, as illustrated earlier; the text in the fields matches the text that defines the title, instructions, and prompt of the Format

dialog box. Press Enter or click on OK to clear the screen.

The Editor example showed the degree to which you can tailor the Shell to your own use. Although the example prepared only one command for MS-DOS to carry out (the Edit command with a file name as a parameter), remember that you can specify more than one command, if necessary, by separating the commands with a semicolon (surround each with blank spaces).

You've also seen how to specify and define a parameter for the Edit command. In your own work, you can specify more than one parameter, each of which has its own dialog box with separate instructions, prompts, and default values. Together, multiple commands and multiple parameters should let you add any program you like to the Shell Program List, providing all the startup information the program might need.

You added only one program item to the Text Files group, but you could add several more, using the same method you used to add the Editor. If you want, you can experiment with programs that are part of MS-DOS. You could, for example, add an item named Display File that runs a Type command.

Using the same techniques, you could add program groups such as Desktop Publishing or Financial Management to run all the application programs you use. With the Task Swapper, you could then start several of your application programs and switch among them without having to exit from one to start another. With the help of the Associate command on the File List's File menu, you can even set up your system so that the Shell starts a particular program and opens the specified file whenever you choose a file name with a specified extension.

Just as you can structure your file system to match your work, you can structure the program choices of the Shell to run the programs that you use.

Deleting a Program Group

When you're through experimenting with the MS-DOS Editor program item, you can delete the program item from the Text Files group, and you can delete the Text Files group itself. A short while ago, you deleted the Mem Command program item. Deleting a program group is just as easy; the only catch is that you cannot delete a group that contains any program items. In this respect, deleting a program group is like using the Remove Directory command to delete a directory: It works only if the group or directory is empty.

To return the Program List to its original state, return to the Main Program List screen if necessary. Next, activate your Text Files group and highlight the Text Editor program item. Choose Delete from the File menu, and press Enter or click on OK when the Shell displays the Delete Item dialog box. (If you've added other program items to this group, delete them, too.)

Once you've deleted all of the program items in a group, you can delete the group itself. It's the same procedure you just followed. First, return to the Main Program List screen. Next,

highlight Text Files, choose Delete from the File menu, and press Enter or click on OK to delete the group. When you're finished, the Shell displays the Main Program List screen again, this time without the Text Files group.

This chapter hasn't covered all the features of the Shell, but it has shown you the most common ones. Because the operating techniques are fairly consistent throughout the Shell and because the Help information is quite thorough, you should be able to learn what isn't covered by selecting the menu items and using the Help feature to guide you.

PREVIOUS NEXT

Chapter 12: Creating and Editing Files of Text

Overview

Although computers were once thought of primarily as machines for mathematics—you still occasionally hear them referred to as "number crunchers"—word processing is the most common use of personal computers today. Word processing programs offer a stunning array of features, accommodating not only the usual memos and reports, but even book-length documents that include multiple columns, a table of contents, an index, and many graphics.

But this array of capabilities is a mixed blessing because a high-end word processor is a large program requiring a significant commitment from both the computer (memory and disk space) and the person using it (learning time).

Beginning with version 5, MS-DOS includes a simple menu-based text editor that offers a good alternative to a word processor for smaller jobs. The official name for this program is the *MS-DOS Editor*, but this chapter will use the name *Edit* instead, both because it's simpler and because that's the name of the command that starts the MS-DOS Editor.

Edit lacks many of the capabilities of a word processor: You must press Enter at the end of each line, for example, and you can't control the capabilities of your printer much beyond the simple printing of text. These very limitations, however, make Edit small, fast, and easy to learn. It's admirably suited for writing short memos and lists and for creating sets of MS-DOS commands known as *batch files*. If you have used a word processor, especially one with drop-down menus, Edit's basic operation should feel familiar.

This chapter shows you around Edit by using an example situation that could occur in any office: You're in charge of a project, and your team has completed several spreadsheets, a 10-page proposal, and a cover letter. You've copied these files to a floppy disk and want the team to review the results one last time before the presentation. You're going to send copies of the floppy disk to the team members, and you need a short memo to tell them what's on it.

Note If you don't have version 5 or later, you have a different editor, called Edlin, which is not described in this book. For a description of how to use Edlin, see the manual that came with your computer.

PREVIOUS NEXT

Using the Keyboard and the Mouse

Like many application programs, Edit responds to either the keyboard or a mouse for most operations. On screen, the keyboard cursor is represented, as usual, by a flashing underline; the mouse is represented by a rectangular block (the mouse pointer). You use the keyboard and the mouse much as you do in the Shell, so if you haven't yet tried out the Shell, turn to the heading "Using the Keyboard and the Mouse" near the beginning of Chapter 11 for a description of how to display a menu and select a menu item or a file name. The text you find there, plus Figure 11-2, summarizes the techniques and defines possibly unfamiliar mouse terms, such as *click, double-click, drag,* and *select*. The first few examples in this chapter give instructions for both keyboard and mouse, but the remaining examples simply ask you to open a menu or choose a menu command.

In addition to choosing from menus, however, you'll frequently use two techniques with Edit that you don't need in the Shell: positioning the cursor where you want to edit text and selecting a block of text. The following descriptions provide some needed background information. Examples later in the chapter provide specifics.

Positioning the Cursor

You can use either the keyboard or the mouse to position the cursor. You'll probably find yourself using both at different times, depending upon where your hands are positioned when you have to move the cursor. Figure 12-1 shows the main cursor-movement keys and their effect.

Key	Moves the Cursor to
Right arrow	Next character
Left arrow	Previous character
Up arrow	Previous line
Down arrow	Next line
PgUp	Previous screen
PgDn	Next screen
End	Last character (including spaces) in the line
Home	First nonspace character in the line
Ctrl-Right arrow	Beginning of the next word
Ctrl-Left arrow	Beginning of the previous word
Ctrl-Enter	Beginning of the next line
Ctrl-End	End of the document

Ctrl-Home	Beginning of the document

Figure 12-1: Edit's cursor-movement keys.

To position the cursor with the mouse, move the mouse pointer to the text location you want, and then click the left mouse button.

Selecting a Block of Text

Some edit operations, such as moving and copying text, require you to select the block of text to be affected. Other operations, such as printing, let you select a portion of the document if you wish; otherwise, the action applies to the entire document.

To select a block of text with the keyboard, position the cursor at the first character to be selected, hold down the Shift key, and then use any of the cursor-movement keys described in Figure 12-1 to choose the text you want. Edit highlights the text you select as you move the cursor. When you release the Shift key, the block remains selected until you change the position of the cursor.

To select a block of text with the mouse, position the mouse pointer on the first character to be selected, press and hold the left mouse button, and move the mouse pointer to the last character of the block. The selected text is highlighted as you move the mouse. Just as when you select text with the keyboard, the text remains selected until you move the cursor again or click the left mouse button.

PREVIOUS NEXT

Starting Edit

To start Edit from the MS-DOS system prompt, type its name:

```
C:\>edit
```

MS-DOS copies the Edit program into memory, and after a moment Edit displays its opening screen.

The screen looks substantially different from a normal MS-DOS display and is noticeably different from the opening Shell screen too. If you take a closer look, however, you can see that the Edit screen has the same essential features as the Shell: a large window in which you work, a menu bar across the top, scroll bars along the edges, and a bar across the bottom telling you how to carry out basic operations. For the most part, you can think of the differences between Edit and the Shell as comparable to the differences between a sedan and a station wagon: They don't look the same, but if you can drive one, you can drive the other. And, as you'll soon see, Edit is easy to work with.

PREVIOUS NEXT

Help

Right now, you should be looking at a large dialog box in the middle of the screen. As in the Shell, dialog boxes are Edit's means of displaying and requesting information. The flashing cursor suggests that you <*Press Enter to see the Survival Guide*>, so press Enter. The screen changes to this:

The Survival Guide is the introductory screen to Edit's online help. (*Online* is a venerable computer term meaning that a program or a device is available; hence, online help.) Most of the information you need for operating Edit is available from the online help whenever Edit is running.

As in the Shell, Edit's help feature explains what you can do at any point in using the program. And, as in the Shell, you can request two types of help: general information and specific instructions (known as *context-sensitive* help) for using a particular command or dialog box. The examples in this chapter direct you to display both types of help, but you can also request assistance any time you want by pressing F1 or by choosing a command from the Help menu. Because Edit's Help differs substantially from the Shell's, the next few pages describe Edit's Help in some detail.

Some help screens include several topics from which you can choose. These topics are enclosed in left-pointing and right-pointing arrowheads. The cursor right now, for example, is under the *G* in the topic *Getting Started*. To see Edit's own description of how to use it and its help feature, select this topic by pressing Enter or by moving the mouse pointer anywhere between the arrowheads and clicking the right mouse button. Edit responds by displaying a set of topics on *Getting Started*.

At the top of the Help window, Edit displays *HELP: Getting Started* to remind you of the topic you chose. Below this are three choices: *Getting Started, Keyboard,* and *Back. Getting Started* is highlighted to show that it is the current topic; its choices are displayed in the lower part of the Help window. *Keyboard* displays additional help, and *Back* returns you to the previous help screen.

To see a little more help, select *Using Help*. Either press the Tab key until the cursor is under the *U* and then press Enter, or move the mouse pointer anywhere between the arrowheads and press the right mouse button. Edit now displays the *Using Help* screen:

This screen describes the help capabilities and how you use them. After you have read the description, press Enter or select *Back* to return to the *Getting Started* screen.

The bottom line of the screen displays some additional help information—in this case, the effect of pressing the function key and other special keys. No matter what you're doing with Edit, the bottom line displays information specific to what you're doing. Right now, the bottom line tells you what happens if you press the following keys:

Key	Description	Effect of Pressing
F1	Help	Displays the help screen of the selected item.
F6	Window	Switches windows. In this case, moves the cursor to the Edit window (now labeled *Untitled*); if the cursor were in the Edit window, F6 would move it to the Help window.
Esc	Cancel	Closes the Help window, and returns to the Edit window.
Ctrl-F1	Next	Displays the next help topic. (Steps through all the help topics, whether or not they apply to what you're doing right now.)
Alt-F1	Back	Displays the previous help topic.

Customizing Help

Right now, the Edit window where you create documents is between the Help window and the bottom line. If you're using a color display, the background color of the Edit window probably isn't the same as that of the Help window. The Edit window is labeled *Untitled* because you didn't name a file when you started Edit.

When you're using Help, you can leave all or part of its current display visible and move between the Edit window and the Help window by pressing F6 or by placing the mouse pointer in the window you want and clicking the left button. That way, you can work on a file and yet refer to help whenever you need it. As you may already have noticed, however, the help screens vary in size. Some, like the one you're looking at now, take about half the screen. Others take only a line or two, and some, like the *Using Help* screen, take almost all the available space.

To help you control your workspace, Edit lets you increase or decrease the size of the active window—the one that contains the cursor. Using the keyboard, press Alt-Plus (hold down Alt and press the plus sign) to make the active window larger; press Alt-Minus to make the active window smaller. With the mouse, simply place the pointer at the top of the Edit window, hold down either mouse button, and move the window border up or down until the Help and Edit windows are the size you want.

If you resize windows and find that either window is too small to display an entire help topic or the file you're working on, you can either change the size again or use the PgUp, PgDn, and arrow keys to scroll through the text with the keyboard. You can also scroll with the mouse by using the scroll bar at the right edge of the active window.

Viewing both windows can be helpful when you're exploring Edit on your own, but for now press Esc to clear the Help window.

PREVIOUS NEXT

Entering Lines

It's time to enter the first few lines of the sample memo. If you make a typing error, you can backspace and correct it before pressing Enter. But don't worry if you press Enter before you realize a line contains errors; as you'll learn, it's easy to correct mistakes. Type the following lines, pressing Enter at the end of each line (press Enter without typing anything where there's a line space):

This floppy disk has 5 files on it.
Please check the spreadsheets and print the
documents to make sure they agree with our
assumptions.

You can review the documents on the
screen. To check the proposal, type this:

TYPE A:PROPOSAL.DOC œ MORE

This displays one screen at a time.
Press the Spacebar to display the
next screen, or press Ctrl-Break to stop.

Let's do this quickly; it's due Thursday.

Tom

The screen shows the lines you have entered.

Adding Text to a File

You insert text in a file by positioning the cursor where the text is to be inserted and typing the new text. For example, to insert the phrase *and recommendations* after the word *assumptions,* use the arrow keys or the mouse to position the cursor under the period that follows *assumptions,* and type *and recommendations* (but don't press Enter).

Now suppose you also decide to add a title to your memo. Use the mouse or press Ctrl-Home to move the cursor to the beginning of the file, the *T* in the word *This*. Type the following lines (press Enter twice after *team* to add a line space):

Final Project Review - 01/05/95
To: Project team

Oops. You also wanted a line space between the title lines. Move the cursor to the *T* in *To* and press Enter.

Now you decide you want to include a list of file names after the first sentence. Position the cursor at the beginning of the line that starts *Please check*. Press Enter to insert a line space, press Tab to indent the line, and then type the following (don't forget to press Enter at the end of the line):

FORECAST.WKS

Look at the cursor. It moved down to the next line, but it isn't at the left margin. It's indented so that it's directly below the beginning of the line you just inserted. What's going on?

Where's the Margin?

If you indent a line with spaces or tabs (which Edit converts to spaces), Edit assumes that you want the next line to start with the same indent. This assumption can be quite a convenience when you're writing an outline or other type of document that has many indented lines, and that's why pressing the Home key moves the cursor to the first nonspace character in a line, not to the left margin.

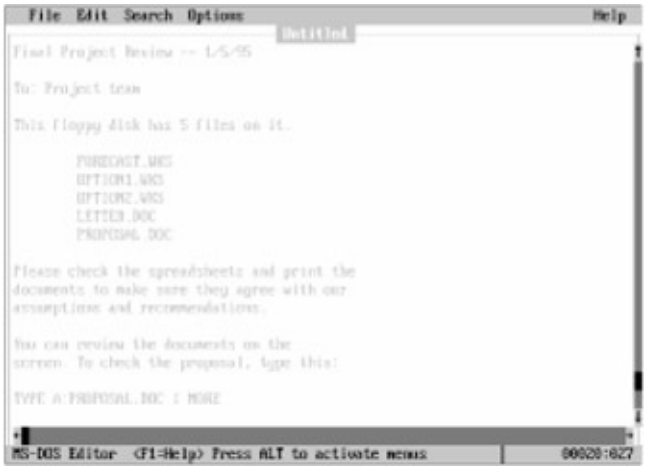
Type the following lines to enter four more file names in the indented list, pressing Enter at the end of each:

OPTION1.WKS
OPTION2.WKS
LETTER.DOC
PROPOSAL.DOC

The assumption that you want to continue an indent isn't always true, of course, so sometimes you'll have to erase the spaces Edit inserts at the beginning of a line. To reduce the inconvenience, Edit lets you erase all the spaces at the beginning of a line by pressing Backspace once when the cursor is under the first nonspace character in an indented line.

Try it. Press Enter once to put a blank line between the last file name in the list and the first line of the following paragraph. Check that the cursor is under the *P* in *Please,* which should be indented eight spaces. Press Backspace; Edit erases the indent and moves the cursor and the line beginning with *Please* to the left margin.

This completes the first draft of the memo. The following illustration shows how your screen should look.



Printing a File

A file stored on disk is perfect for distribution but is otherwise of limited value. In most cases, you want a printed copy of the documents you create. To print this version of the memo, make sure your printer is turned on, and then choose Print from the File menu using one of the two methods described below:

- Using the keyboard, press Alt, F, and then P.
- Using the mouse, click on File in the menu bar at the top of the screen, then click on Print.

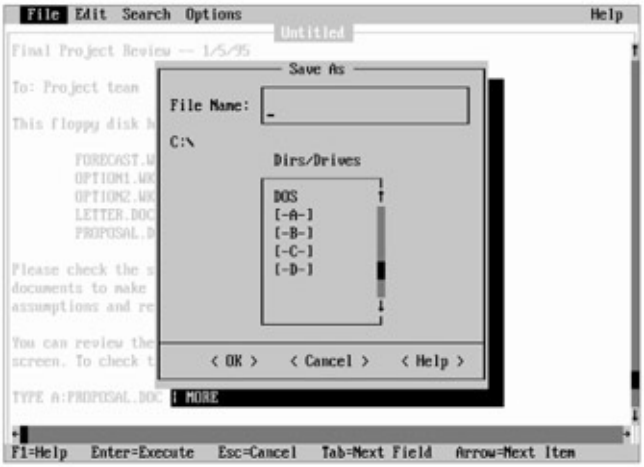
> **Note** From now on, use keystroke or mouse sequences like this whenever you're asked to choose a command from a particular menu.

Edit responds with a dialog box asking whether you want to print the entire document or just the part that is selected. The small dot in parentheses tells you that Edit proposes to print the complete document. This is what you want, so press Enter or click on OK; Edit should print the file. If your printer isn't ready, Edit displays the message *Device fault*. Check the printer, and try again by pressing the Enter key.

PREVIOUS NEXT

Saving a File

Your memo so far is stored only in the computer's memory, not on disk. If you turned the computer off, the document would be lost. To save the file on disk, choose Save As from the File menu. Edit displays a dialog box that asks you to name the file:



The cursor is in the text box titled *File Name* because Edit is waiting for you to type a file name. Below File Name is the name of the current directory (it should read C:\). The box that follows, titled *Dirs/Drives,* lets you save the file in a different directory or even on a disk in a different drive.

Edit can save a file to any directory or drive on your system. For this example, you'll simply type a drive letter and a file name in the File Name text box. Make sure that a formatted floppy disk is in drive A and type this:

```
a:memo.txt
```

Now press Enter, or choose OK at the bottom of the dialog box. The file is saved, as you can see from the title block at the top of the screen, which has changed from *Untitled* to *MEMO.TXT*.

More About Directories and Drives

As you saw in Chapters 8 and 9, directories are invaluable tools. Before you move on, choose the Save As and Open commands from the File menu, and take a few moments to examine the dialog boxes they display a little more closely. (Press Esc or choose Cancel to clear the screen when you're through.)

To help you find or save a file in any directory on any drive, the Dirs/Drives text box in these dialog boxes lists the names of all subdirectories of the current directory and ends with the names of all the drives on your system. The drives are easy to recognize because they are enclosed in square brackets like this: [-A-].

To use the Dirs/Drives text box, you move the cursor to the text box with the Tab key, use the arrow keys to highlight the directory or drive you want, and press Enter (or double-click

with the mouse) to change the current drive/directory. To move farther down the directory tree, you repeat the same steps.

Once you've chosen a new drive or directory in the Save As dialog box, the highlight returns to the File Name text box so that you can type the name you want to give the file. If you're using the Open dialog box, you can either type a file name or choose a drive or directory to display a list of files in the Files box. The list will contain all of the files from that drive or directory that match the pattern in the File Name box, such as **.txt*. You can then choose a file from this list. The Dirs/Drives text box is particularly useful when you want to scan your drives and directories, either to store a file or to load one whose location you're not sure of.

PREVIOUS NEXT

Deleting Text

To delete the character at the location of the cursor, press Del. To delete more than one character, select them all and either press Del or choose the Clear command from the Edit menu. Using either the keyboard or the mouse, you can select any amount of text, from a single character to the entire document, as described early in this chapter.

To practice deleting text, you'll return to the file MEMO.TXT Begin by deleting the words *on it* at the end of the line that precedes the list of file names. Position the cursor under the period at the end of the line. Although it seems as if you're starting at the wrong end, remember that you can extend the selection either to the left or to the right. Hold down the Shift key, and press the Left arrow key until *on it* and the preceding space are highlighted. Now simply press Del or choose Clear from the Edit menu. The words are gone.

To delete an entire line and not leave any odd line spacing in your document, you delete everything: text, spaces, even the invisible carriage return that Edit uses to mark the end of a line. Suppose, for example, you decide not to include the file named LETTER.DOC and want to delete the entire line from the memo. Simply highlighting the file name won't do the job because the carriage return will still be there, creating a blank line. Instead, start by positioning the cursor. If you're using the keyboard, hold down Ctrl and press Enter several times to move the cursor to the *L* in *LETTER*. Press Backspace to delete the indent. Now hold down the Shift key and press the Down arrow key once. The entire line is selected.

With the mouse it's even easier: Position the cursor at the left edge of the line, press and hold the left mouse button, and move the mouse slightly to drag the mouse pointer one row down.

When the line is selected, press Del or choose the Clear command to delete the line.

PREVIOUS NEXT

Ending an Editing Session

To leave Edit and return to MS-DOS, choose Exit from the File menu. You have changed the file since the last time you saved it, so Edit asks you whether it should save the file first; select Yes. Edit saves the file, then ends and returns to MS-DOS, which displays the system prompt.

If you make some changes to a file, then decide you don't want to change the file after all, you can cancel the editing session without saving the changes by selecting No when Edit asks whether it should save the file.

PREVIOUS NEXT

Editing an Existing Text File

When you type *edit* followed by the name of a file, Edit checks to see whether the file you named exists. If the file exists, Edit copies the file into memory; if the file doesn't exist, Edit shows you an empty window but uses the file name as a title and remembers the file name when you save the file.

To work with the file you just created, type the following:

```
C:\>edit a:memo.txt
```

Edit starts and displays the memo.

Copying and Moving Text

You can copy or move text from one place in a document to another by using a special area of Edit's memory called the *Clipboard*. Copying, of course, leaves the text in the original location unchanged, while moving deletes the text from its original location.

To copy text, you first copy it to the Clipboard, then copy it from the Clipboard to the cursor location. (When you insert your text in the new location, you *paste* it, from the phrase *cut and paste*.) Suppose you wanted to copy the list of file names in MEMO.TXT to the end of the memo, following the signature line. First, select the file names. To use the keyboard, position the cursor under the *F* in *FORECAST.WKS*, hold down either Shift key, and press the Down arrow key four times; to use the mouse, move the mouse pointer to the *F*, hold down the left mouse button, and move the mouse down until all four lines are highlighted.

Next, open the Edit menu. All the choices show a highlighted letter; the second choice is Copy, with a highlighted C (notice that the shortcut key for copying to the Clipboard is Ctrl-Ins); click on Copy or press C. The Edit menu disappears. There's no indication that anything has happened, but Edit has copied the four lines to the Clipboard.

Now move the cursor down to the line that follows *Tom,* and open the Edit menu again. This time, only Paste shows a highlighted letter, meaning that it's the only valid choice from the menu (its shortcut key is Shift-Ins). Paste is an option only when something is on the Clipboard, so the dark characters and the highlighted P tell you that the lines were indeed copied to the Clipboard. Press P, and the four file names appear in the new location.

Moving text is almost the same as copying; the only difference is that you select Cut instead of Copy from the Edit menu. To move the new set of file names back immediately following the original list, select the lines you just pasted and choose Cut from the Edit menu (the shortcut key for Cut is Shift-Del). The lines disappear from the screen, just as if you had deleted them. But move the cursor up to the blank line that follows the last file name in the original location and press Shift-Ins (the shortcut key for Paste). The second list of file names appears immediately below the first.

Pasting from the Clipboard doesn't erase its contents. Press Shift-Ins again, and the four

lines of file names are inserted again. You should now see 12 file names.

Copying, cutting, and pasting make it easy to rearrange a document. You don't want to keep this version of the memo, so select Open from the File menu, type *a:memo.txt* in the File Name text box, and select *No* when Edit asks if you want to save the loaded file. The changed version of the file is replaced by the original.

PREVIOUS NEXT

Searching for a Group of Characters

As a file gets longer, it takes you longer to find a particular word or line, and there's more chance that you'll miss it. Edit eliminates this problem by searching for any character or group of characters you specify.

This memo isn't long enough to require that sort of help, but you can still see how it works. Suppose you wanted to check the names of all the files whose extension is DOC. Choose Find from the Search menu. Edit asks you what you want to find and proposes whatever word the cursor is currently beneath. Type *doc* in the Find What text box and press Enter.

Almost immediately, Edit has highlighted *DOC* in *PROPOSAL.DOC*. Notice that although you typed *doc,* Edit found *DOC* too. As you'll see in a moment, you can tell Edit whether to distinguish between uppercase and lowercase letters.

If you want to find the same characters again, you don't have to repeat the whole process: Just press F3, which tells Edit to repeat the last search. This time it highlights *doc* in the word *documents*. That's not quite what you had in mind, but it's perfectly normal behavior for Edit: It searches for the characters you specify in either uppercase or lowercase, anywhere within a word, unless you specify otherwise.

Choose Find from the Search menu again (it should show *doc* in the Find What text box). First, type *DOC* to replace *doc.* Next, notice the option labeled Match Upper/Lowercase. Select the option by pressing Tab and then pressing the Spacebar or by clicking the mouse anywhere in the option. An X appears inside the brackets.

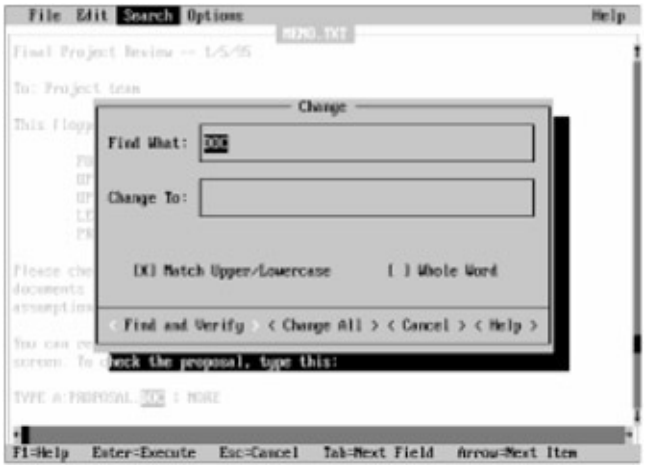
Now press Enter or choose OK to search again. Edit jumps down and highlights *DOC* in the line beginning *TYPE A:PROPOSAL.DOC.* Press F3 to repeat the search, and the highlight moves up to the preceding PROPOSAL.DOC. Now press F3 again; instead of highlighting *doc* in *documents,* Edit again highlights *DOC* in the line beginning *TYPE.* You told Edit you didn't want to find *doc,* just *DOC*.

PREVIOUS NEXT

Replacing One Group of Characters with Another

One of the most common reasons you'll want to find a particular word or group of characters in a document is so that you can change it. Edit will not only find the characters for you, as you just saw, it will also make the change for you. Suppose you want to change the file extension WKS to XLS wherever it occurs in the memo.

Choose Change from the Search menu. Edit displays the Change dialog box, as shown below.



In addition to the Find What text box you saw in the Find dialog box, there's a Change To text box. The Find What text box should contain *DOC;* you want to change every occurrence of WKS to XLS, so type *WKS*. Move to the Change To text box and type *XLS*. Leave the Match Upper/Lowercase option turned on so that Edit will search only for *WKS* and not *wks*. Find And Verify is highlighted at the bottom of the dialog box as the action to take, so press Enter to start the search.

Edit highlights the first occurrence of *WKS* (in *FORECAST.WKS*) and asks you whether to change or skip this occurrence. Press Enter (or click on Change) to change it; Edit displays the next occurrence of *WKS* and repeats the question. Change this occurrence too. When Edit displays the last occurrence of *WKS* (in *OPTION2.WKS*), change it as well. Press Enter or click OK when Edit tells you *Change complete*. The screen shows that all three file extensions have changed from WKS to XLS.

To tell Edit to change all occurrences without prompting for verification, you would select Change All instead of Find And Verify in the Change dialog box.

PREVIOUS NEXT

Inserting and Overstriking Text

You probably used the Backspace key to correct typing errors while you were entering the memo. After writing a memo or a report, however, you'll usually want to make some changes after you read through the first draft. To change text after you have entered it, you start by positioning the cursor where you want to make the change. At that point you can delete incorrect characters, insert new ones, or type correct characters in place of the incorrect ones (the latter is called *overstriking*). The next few examples show you how to make typical changes in a document.

To change the word *Project* to *Product* in the first line of the memo, move the cursor to the *j* and press the Insert key. The cursor changes from an underline to a flashing block. Each time you press Insert, you alternate back and forth between inserting characters (signified by the flashing underline) and replacing characters (signified by the flashing block). Now, with the flashing block showing, whatever you type won't be inserted; it will replace what is on the screen. Type *du;* the new characters replace the old, changing *Project* to *Product*.

Press Insert again to change the cursor back to an underline so that characters you type are inserted.

Insert the word *New* before *Project* in the second title line by moving the cursor to the *P* and typing *New* followed by a space (but don't press Enter, because you don't want to enter a carriage return character). The remainder of the line moves over to make room for the new characters.

Next, change the 5 to a 4 in the third line, and change the period at the end of the third line to a colon. Position the cursor under the 5, press Insert to change to overstrike, and type *4*. Then move the flashing block to the period, and type a colon. Press Insert to return to inserting rather than overstriking characters.

Finally, add a file description to the second file name (OPTION1.XLS). Move the cursor to the space after *XLS,* press the Spacebar three times, and type *Higher sales.* Do the same to add the file description *No staff increase* three spaces after the third file name (OPTION2.XLS).

PREVIOUS NEXT

Copying from Another File

Suppose you had another file that contained the mailing address of the project team members, and you wanted to include that list at the beginning of the memo. You wouldn't have to type the list; you could simply copy it from the other file. You can try this feature by creating another file of addresses and copying it into MEMO.TXT.

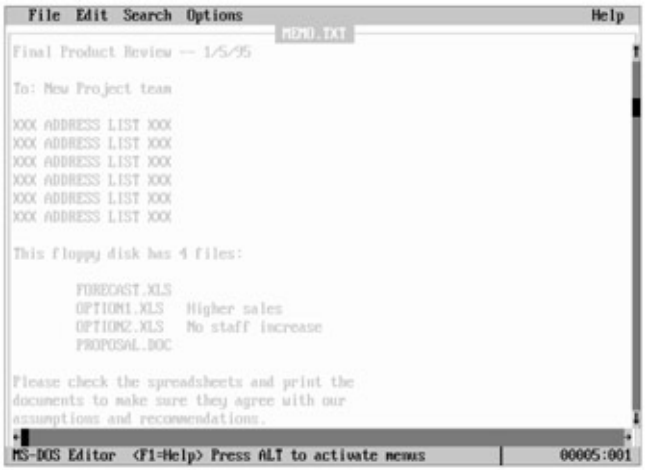
You'll create a file named ADDRESS.TXT that contains six dummy address lines. First, save MEMO.TXT by choosing Save from the File menu, and then remove the file from memory by choosing New from the File menu. Edit clears the screen to show you that you're starting a new file.

Next, type the following line:

XXX ADDRESS LIST XXX

Now make five more copies of the line. To do this, select the line (make sure that the entire line, from the left margin to the right edge of the screen, is highlighted), press Ctrl-Ins to copy it to the Clipboard, press Home to remove the highlight, and press Shift-Ins five times to copy the lines. You should see six identical lines. Copy all six of the lines to the Clipboard, and then save the file by choosing Save As from the File menu, typing *a:address.txt,* and pressing Enter.

Now copy the lines from the Clipboard into a different file. Load the MEMO.TXT file by choosing *Open* from the File menu, typing *a:memo.txt* in the File Name text box, and pressing Enter. You want to paste the address list just after *To: New Project team,* so position the cursor at the beginning of the blank line below *To.* Press Enter to add an extra blank line for spacing. Now press Shift-Ins to insert the six lines from the Clipboard. This is the final version of the memo; the screen should look like this:



Save this final version of the memo. Now for something a bit less tedious before you leave Edit.

Changing the Screen Display

If you have a color monitor, Edit probably displayed white text against a blue background when you first started it. You can change this color scheme to any combination of foreground (text) and background (screen) color your display and MS-DOS are capable of showing. Select Display from the Options menu. The screen should look like this:

Before you start customizing your display, turn to Edit's Help again, this time to see how Edit can advise you on the Display dialog box. Press F1, and Edit displays the Help window that tells you how to use the Display dialog box. Read the explanation, and then press Esc to remove the Help window.

The cursor is now in the text box that lists possible foreground colors. Using the arrow keys or the mouse, select different colors (you can move the cursor between the foreground and background color boxes by pressing Tab and Shift-Tab). The text block to the left of the color boxes changes to show the effect of each color selection.

When you find a combination you like, press Enter to return to Edit. It will use these colors the next time you start it. (If you prefer the standard combination, press Esc to cancel the dialog box; Edit leaves the colors unchanged.) Leave Edit by choosing Exit from the File menu.

PREVIOUS NEXT

Chapter Summary

This chapter showed you the basics of using Edit. You're ready to use it for many of the short text-editing jobs that don't require a word processor.

There's more to Edit, such as different methods of cutting and pasting text and shortcut keys based on the WordStar word processor. You can explore further on your own, using Help as your guide: Display a help screen—the Keyboard help screens are a good place to start—and use the F6 key to switch back and forth between the Help and Edit windows as you try out new features. You'll find that Help is the most patient instructor you've ever had.

PREVIOUS NEXT

Chapter 13: Taking Control of Your System

Overview

Up until now in this book, you have used the MS-DOS commands in their standard form. MS-DOS, however, gives you a great deal of flexibility in controlling the way some commands do their work for you. This chapter describes two ways in which you can take control of your system.

The beginning of this chapter introduces the concept of input and output *redirection.* Essentially, redirection is a form of traffic control that allows you to route input and output between devices and files you specify. You'll also see in this chapter how to use redirection with MS-DOS commands that let you control both the form and the content of command input and output.

If you have version 5 or later of MS-DOS, the second part of this chapter takes you further into the Doskey program introduced in Chapter 3, "Getting Your Bearings." Using Doskey, you'll see not only how to edit commands you've already used, but also how to enter and carry out more than one command at a time.

PREVIOUS NEXT

Redirecting Command Output

It is easy to visualize what happens with command output, so even though it might seem odd to discuss results before discussing causes, this section on command output gives you a foundation for understanding command input as well.

The result, or output, of most commands is some action, such as copying a file (with the Copy command) or controlling the operation of a device (with the Mode command). The output of some commands, however, such as Directory, Check Disk, and Tree, is a report. Up to now, you have used these reports primarily as displays—MS-DOS has sent them to the *standard output device,* the console. (Recall from Chapter 7, "Managing Your Devices," that MS-DOS uses the name CON, or console, for both the keyboard, which is input only, and the display, which is output only.)

As you'll see in this chapter, MS-DOS lets you send reports and other output to a different device, such as a printer, or to a file. This is called *redirecting* the output of the command; you did it a few times in previous chapters with such commands as *dir > prn.* The technique is simple: To redirect the output of a command that normally sends its results to standard output (the display), you type the command name followed by > and the name of the device or file to which the output is to be sent. The > looks something like an arrowhead pointing toward the alternate output device or file. If you redirect output to a file that doesn't exist, MS-DOS will create the file.

You can easily redirect output to print a copy of the directory as you did in Chapter 4, "A Look at Files and Floppy Disks," for example. To repeat that example, but this time with an understanding of what happens, be sure the printer is turned on and type the following:

```
C:\>dir > prn
```

The > tells MS-DOS to redirect the output of the Directory command, and PRN tells MS-DOS where to send it: to the printer. The directory should be printing now. If it's long and you want to cancel the printing, press Ctrl-Break.

PREVIOUS NEXT

Redirecting Command Input

You've seen how quickly and easily you can redirect output to a device or a file. You can just as easily redirect input, in effect telling certain MS-DOS commands to get their data from a source other than the one (often called *standard input device)* they would normally use. To redirect input, follow the command name with the < character and the name or the device from which the input is coming.

Together, redirected input and output are known as *redirection.* Although redirection might sound complicated, it is easy to understand, as the examples in this chapter show.

Three MS-DOS commands, known collectively as *filter commands,* make particularly effective use of redirection.

PREVIOUS NEXT

Filter Commands

Filter commands take input from a device or a file, change the input in some way, and send the result to an output device or file. They are called filter commands because they work much like a filter in a water system, which takes incoming water, changes it in some way, and sends it along the system.

MS-DOS includes three filter commands:

- *Sort* arranges lines in ascending or descending order.
- *Find* searches for a string of characters.
- *More* temporarily halts the display after one screenful (to give you a chance to read the lines).

You can redirect both the input and the output of a filter command. The filter commands aren't really intended to be used with keyboard input. Rather, they are designed to get their input from a file or even from the output of another command. This chapter shows you how to use and combine the filter commands to create your own powerful, specialized commands.

PREVIOUS NEXT

Preparing for the Examples

Redirection and filter commands give you the elements of a simple file-management program. While they won't replace a file manager, they let you use MS-DOS to search and sort simple lists without spending extra money or time on another program.

The examples in this chapter use a sample file that almost everyone needs: a list of names and telephone numbers. Too often, files of telephone numbers and business cards are out of date, incomplete, or in the other office. Questions arise: "Was that number in the telephone index or the business-card file?" Or "Did I file the number under Jones or under Accountants?" This example shows you how to let MS-DOS keep track of your phone list and eliminate these questions.

Change the current directory to \RUNDOS by typing the following:

```
C:\>cd \rundos
```

Entering the Sample File

The sample file is named PH.TXT. You can use Edit to create it from MS-DOS, or you can use your own word processor if it can save documents as unformatted text files—that is, files that don't contain any of the program's own formatting codes.

Each line of the file contains six items of data: last name, first name, area code, telephone number, a key word that identifies a category, and a short description. The key words are: *cust* for customer, *cons* for consultant, and *vend* for vendor. You'll enter the items in specific columns so you can sort the list by any item.

To use Edit to create the file, start Edit and name the file in a single command by typing the following:

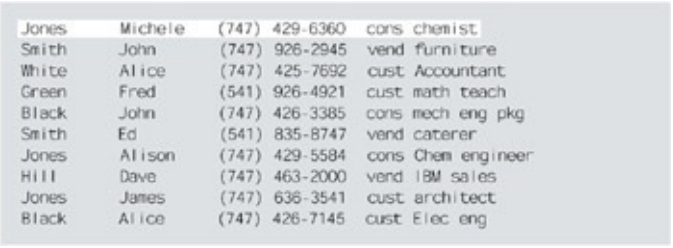
```
C:\RUNDOS>edit ph.txt
```

Now you can begin to enter the items in the file. To help you get the items in the correct columns, the following entry shows the first line, with a period marking each space. Type the line as shown, pressing the Spacebar once for each period shown:

```
Jones.....Michele...(747).429-6360..cons.chemist
```

End the line by pressing Enter.

Enter the remaining lines in the sample file, shown in Figure 13-1, using the spacing of the first line as a guide. When you've typed all the lines, save the file and quit the editor.



```
Jones      Michele   (747) 429-6360  cons chemist
Smith      John      (747) 926-2945  vend furniture
White      Alice     (747) 425-7692  cust Accountant
Green      Fred      (541) 926-4921  cust math teach
Black      John      (747) 426-3385  cons mech eng pkg
Smith      Ed        (541) 835-8747  vend caterer
Jones      Alison    (747) 429-5584  cons Chem engineer
Hill       Dave      (747) 463-2000  vend IBM sales
Jones      James     (747) 636-3541  cust architect
Black      Alice     (747) 426-7145  cust Elec eng
```

Figure 13-1: Telephone and business-card list.

Now you're ready to use the file. To see the entry for Alice White, type the following (notice the capital *W*):

```
C:\RUNDOS>find "Wh" ph.txt

---------- PH.TXT
White      Alice    (747) 425-7692  cust Accountant
```

That's fast, but it's just the beginning.

PREVIOUS NEXT

The Sort Filter Command

The Sort filter command arranges, or sorts, lines of input and sends them to standard output (the display) unless you redirect the output—for example, to the printer. If you enter the command with no options, it sorts the lines of input in ascending order (alphabetically from A to Z, or numerically from lowest to highest number), starting the sort on the character in the first column.

The Sort command has two parameters:

sort /R /+<column>

/R (*Reverse*) sorts the lines in reverse order (Z to A, or highest to lowest number).

/ + <column> sorts the lines starting at the specified column, rather than starting in the first column.

To sort a particular file, you can redirect the input of the Sort command by following the command name with < and the name of the file to be sorted. If you don't have version 4 or later of MS-DOS, be sure to use a space both before and after the <. If you don't redirect the input, the Sort command sorts lines that you type at the keyboard (standard input).

Sort Command Examples

Figure 13-2 shows the column number of each of the six items in the telephone list: last name, first name, area code, telephone number, key word, and description. You'll use these column numbers to sort the file in different ways.

Figure 13-2: Column numbers of items in the telephone list.

The simplest way to sort the file is in ascending order, starting in the first column. (In the sample file, this sorts the entries by last name.) Type this:

```
C:\RUNDOS>sort < ph.txt
```

MS-DOS quickly displays the sorted result:

```
Black     Alice      (747) 426-7145  cust Elec eng
Black     John       (747) 426-3385  cons mech eng pkg
Green     Fred       (541) 926-4921  cust math teach
Hill      Dave       (747) 463-2000  vend IBM sales
Jones     Alison     (747) 429-5584  cons Chem engineer
Jones     James      (747) 636-3541  cust architect
Jones     Michele    (747) 429-6360  cons chemist
Smith     Ed         (541) 835-8747  vend caterer
```

```
Smith     John       (747) 926-2945  vend furniture
White     Alice      (747) 425-7692  cust Accountant
```

The file itself hasn't changed; what you see is simply the result of MS-DOS reading, sorting, and displaying the lines of the file.

To sort the file in reverse order, use the /R option:

```
C:\RUNDOS>sort /r < ph.txt
```

It doesn't take MS-DOS any longer to sort backward:

```
White     Alice      (747) 425-7692  cust Accountant
Smith     John       (747) 926-2945  vend furniture
Smith     Ed         (541) 835-8747  vend caterer
Jones     Michele    (747) 429-6360  cons chemist
Jones     James      (747) 636-3541  cust architect
Jones     Alison     (747) 429-5584  cons Chem engineer
Hill      Dave       (747) 463-2000  vend IBM sales
Green     Fred       (541) 926-4921  cust math teach
Black     John       (747) 426-3385  cons mech eng pkg
Black     Alice      (747) 426-7145  cust Elec eng
```

Suppose you wanted to arrange the list by the key word—first the consultants, then the customers, then the vendors. The first letter of the key word is in column 37, so use the column option:

```
C:\RUNDOS>sort /+37 < ph.txt
```

Now it's easy to pick out the different categories:

```
Jones     Alison     (747) 429-5584  cons Chem engineer
Jones     Michele    (747) 429-6360  cons chemist
Black     John       (747) 426-3385  cons mech eng pkg
White     Alice      (747) 425-7692  cust Accountant
Jones     James      (747) 636-3541  cust architect
Black     Alice      (747) 426-7145  cust Elec eng
Green     Fred       (541) 926-4921  cust math teach
Smith     Ed         (541) 835-8747  vend caterer
Smith     John       (747) 926-2945  vend furniture
Hill      Dave       (747) 463-2000  vend IBM sales
```

Sorting is fast, easy, and useful.

The Find Filter Command

The Find filter command searches lines of input for a string of characters you specify. If you enter the command with no parameters other than the string and the file name, the Find command displays all lines that contain the string.

The Find command has six parameters:

find /V /C /N /I "<string>" <filename>

/V displays all lines that do not contain the string.

/C (*Count*) displays just the number of lines found, not the lines themselves.

/N (*Number*) displays the input line number with each line found.

/I (*Ignore*), in versions 5 and later, causes the Find command to ignore differences between uppercase and lowercase—for example, to treat *a* and *A* as the same letter.

"<string>" is the string of characters you want to find; it must be enclosed in quotation marks. Keep in mind that unless you specify the /I parameter, the Find command distinguishes between uppercase and lowercase letters, so "cons" and "CONS", for example, are different strings.

<filename> is the name of the file to be searched. If you omit <filename>, the Find command searches standard input. You can include several different file names in a single Find command simply by separating the file names with spaces. If some of the files are in a different directory or on a disk in a different drive, precede the file name with the appropriate path name or drive letter.

Find Command Examples

To display the entries for all consultants in the file named PH.TXT, type the following:

```
C:\RUNDOS>find cons" ph.txt
```

The first line of output identifies the input file, PH.TXT. Each line that contains the string *cons* is displayed immediately after:

```
----------  PH.TXT
Jones      Michele    (747) 429-6360  cons chemist
Black      John       (747) 426-3385  cons mech eng pkg
Jones      Alison     (747) 429-5584  cons Chem engineer
```

To see how the Find command works with more than one file, type the following to make a duplicate copy of the phone list:

```
C:\RUNDOS>copy ph.txt ph1.txt
```

Now type the Find command you used in the preceding example, but this time include both files:

```
C:\RUNDOS>find "cons" ph.txt ph1.txt
```

MS-DOS displays the name of each file as it finds *cons* in PH.TXT and PH1.TXT:

```
---------- PH.TXT
Jones     Michele    (747) 429-6360  cons chemist
Black     John       (747) 426-3385  cons mech eng pkg
Jones     Alison     (747) 429-5584  cons Chem engineer

---------- PH1.TXT
Jones     Michele    (747) 429-6360  cons chemist
Black     John       (747) 426-3385  cons mech eng pkg
Jones     Alison     (747) 429-5584  cons Chem engineer
```

Obviously, using the Find command to search for a character string in several files is more productive when the contents of the files differ, but this example shows the method, if not the full capability, of the command. You don't need PH1.TXT anymore, so type *del ph1.txt* to keep your disk uncluttered.

Return now to the original phone list, PH.TXT. If you just want to know how many consultants are in the list, use the /C option:

```
C:\RUNDOS>find /c cons" ph.txt
```

This time, the line that identifies the input file also shows the number of lines that contain the string:

```
---------- PH.TXT: 3
```

Three lines in the file contain *cons*.

As often happens in a real telephone index, the sample file uses different words or abbreviations to mean the same thing. Both *engineer* and *eng* are used, for example, to describe an engineer. Both words contain *eng,* however, so you can find all the engineers by typing the following:

```
C:\RUNDOS>find "eng" ph.txt

---------- PH.TXT
Black     John       (747) 426-3385  cons mech eng pkg
Jones     Alison     (747) 429-5584  cons Chem engineer
Black     Alice      (747) 426-7145  cust Elec eng
```

As also happens in real life, capitalization in the sample file is not consistent. One entry, for example, contains the notation *Chem engineer,* and another contains the all-lowercase *chemist*. You could avoid the inconsistency by specifying the string as *hem* rather than as *chem* or *Chem*. But if you have version 5 or later, you can use the /I parameter instead to

tell the Find command to ignore uppercase/lowercase differences. If you have version 5 or later, type this:

```
C:\RUNDOS>find /i "chem" ph.txt
```

MS-DOS quickly responds:

```
---------- PH.TXT
Jones     Michele     (747) 429-6360  cons chemist
Jones     Alison      (747) 429-5584  cons Chem engineer
```

Finding Lines That Don't Contain the String

With any version of MS-DOS, you can also use the /V parameter to display lines that *don't* contain a string. For example, to display the entries *not* in the 747 area code, typ e this:

```
C:\RUNDOS>find /v "(747" ph.txt

---------- PH.TXT
Green     Fred      (541) 926-4921  cust math teach
Smith     Ed        (541) 835-8747  vend caterer
```

Including the left parenthesis with 747 distinguishes between entries with an area code of 747 and entries that might contain 747 in the phone number. Try the example without the left parenthesis:

```
C:\RUNDOS>find /v "747" ph.txt

---------- PH.TXT
Green     Fred      (541) 926-4921  cust math teach
```

Ed Smith's telephone number is 835-8747, so his entry wasn't displayed even though his area code is 541. When you enter the characters to find, be sure to include enough information to specify what you're looking for. In the sample file, for example, typing "(*7*" would be enough to specify the 747 area code; it wouldn't be enough, however, if the file contained another area code beginning with 7. And, as you saw, you must include the left parenthesis to distinguish an area code from some other set of numbers.

Including Line Numbers with the Output

To display the entries for people named Smith and to include the line numbers of the entries, use the /N option:

```
C:\RUNDOS>find /n "Smith" ph.txt

---------- PH.TXT
[2]Smith     John      (747) 926-2945  vend furniture
[6]Smith     Ed        (541) 835-8747  vend caterer
```

The two entries displayed are the second and sixth lines of the sample file.

Combining Find Command Options

You can combine Find command options. For example, to display the entries not in the 747 area code and to include their line numbers, use both the /V and /N options:

```
C:\RUNDOS>find /v /n "(7" ph.txt

---------- PH.TXT
[4]Green     Fred       (541) 926-4921  cust math teach
[6]Smith     Ed         (541) 835-8747  vend caterer
```

(If you used Edit to create the sample file, you might also see *[11]* appear at the end of this report. That's probably because you pressed Enter when you finished typing the last entry and, in doing so, created a blank line that Find has included in its search. The blank line is listed because you told Find to show the *numbers* of all lines that *don't* contain the 747 area code.)

PREVIOUS NEXT

More on Redirecting

Earlier in this chapter, you redirected input to the Sort command by specifying the file PH.TXT. You can also redirect the output of a filter command. To print the entries for all vendors (that is, to redirect output from the display to the printer), type the following:

```
C:\RUNDOS>find "vend" ph.txt > prn
```

The entries are printed. If your phone list has two or three hundred entries, this technique of using options and redirecting output is a quick way to print a copy showing a selected group of entries.

Redirecting Both Input and Output

You can redirect both input and output by following the command name with < and the name of the input file or device and then > followed by the name of the output file or device. (Be sure to include spaces before and after both < and > if you don't have version 4 or later of MS-DOS.)

For example, to print the alphabetized version of PH.TXT, check that your printer is on and then type this:

```
C:\RUNDOS>sort < ph.txt > prn
```

The input for the Sort command comes from PH.TXT, and the output is redirected to the printer.

MS-DOS allows you to redirect both the input and the output of a single command from and to the same file. That is, MS-DOS does not report any error if, for example, you type the command *sort < ph.txt > ph.txt.* Don't do this, however; you might destroy your data. If you want to sort a file and keep the same file name, do the following:

1. Redirect the output to a temporary file, such as PH-TMP.TXT. To sort PH.TXT in reverse order, you would type *sort /r < ph.txt > ph-tmp.txt*.

2. If you're certain you don't need it, delete the original file (*del ph.txt*). If you want to keep the original, give it a different name, such as OLDPH.TXT, using the Rename command (*ren ph.txt oldph.txt*).

3. Then use the Rename command to give the temporary file the original file name (*ren ph-tmp.txt ph.txt*).

4. The new version of PH.TXT would contain the telephone list sorted in reverse order. You could verify the contents by displaying PH.TXT with the Type command.

Adding Redirected Output to a File

When you redirect output to an existing file, the redirected output replaces the original file.

But you can also *add* redirected output to the end of an existing file by using >> instead of >. If the file doesn't already exist, it is created, just as when you use >.

PREVIOUS NEXT

Connecting Commands with a Pipe

A powerful way of using a filter command is to redirect the output of some other command to the input of the filter command. In effect, the two commands are connected, with the output of the first command feeding directly into the filter command. Continuing the analogy to a water system, this connection is called a *pipe.*

You tell MS-DOS to pipe the output of one command to the input of another by typing | between the names of the two commands; the | provides the connection between the two commands. The More filter command provides a simple example.

The More Filter Command

The More filter command displays one screenful (24 lines unless you've specified otherwise with the Mode command) followed by the line -- *More* --, and then it pauses. When you press any key, More displays the next screenful and pauses again if necessary, continuing in the same way until all the input has been displayed.

For the following example, you need a disk or a directory that contains more than one screenful of files. Your \DOS directory will do; type the following:

```
C:\RUNDOS>dir \dos | more
```

This command tells MS-DOS to redirect the output of the Directory command to the input of the More command. The More command displays the first screenful of the directory and -- *More* -- at the bottom of the screen. Press any key to see the rest of the directory, or press Ctrl-Break to cancel the More command and return to the system prompt. The More command lets you review a long output sequence or file without having to press Pause or Ctrl-Num Lock to start and stop the display.

Combining Filter Commands

You can pipe the output of one Find command to the input of another Find command to make a more specific search. A real-life list like the sample phone list, for example, might include several dozen customers. Suppose you want to display the names of customers in the 747 area code only. To do this, pipe the output of a Find command that searches for *cust* to another Find command that searches for (*7.* Type this:

```
C:\RUNDOS>find cust" ph.txt | find  "(7"
White      Alice    (747) 425-7692  cust Accountant
Jones      James    (747) 636-3541  cust architect
Black      Alice    (747) 426-7145  cust Elec eng
```

If you check an earlier list of the file, you'll see that Fred Green is a customer, but his area code is 541, so the second Find command eliminated the entry for his name. Notice that the line that identifies the file (----------PH.TXT) isn't displayed. The first Find command pipes ----------PH.TXT as part of its output to the second Find command, but because the line does

not contain the string (*7,* it is not included as part of the output of the second Find command.

You can also pipe the output of the Find command to the Sort command. To see all the consultants sorted by last name, type the following:

```
C:\RUNDOS>find "cons" ph.txt | sort

---------- PH.TXT
Black     John      (747) 426-3385  cons mech eng pkg
Jones     Alison    (747) 429-5584  cons Chem engineer
Jones     Michele   (747) 429-6360  cons chemist
```

You can combine as many commands as you like. Suppose you want to print a list of all customers in the 747 area code, sorted by telephone number. You can search PH.TXT for *cust*, pipe that output to a Find command that searches for (*7,* pipe that output to a Sort command that sorts at column 27 (the telephone number), and redirect the output to the printer. Try this by typing the following:

```
C:\RUNDOS>find "cust" ph.txt | find "(7" | sort /+27 > prn
```

The printed output includes this:

```
White     Alice     (747) 425-7692  cust Accountant
Black     Alice     (747) 426-7145  cust Elec eng
Jones     James     (747) 636-3541  cust architect
```

If your list included several dozen customers, this could be a handy way to organize a calling campaign.

The Difference Between > and |

Sometimes the distinction between > and | isn't readily apparent. > redirects output to a file or device; | redirects output to another command. This difference is easy to demonstrate. Sort is a filter command. To make the output of the Directory command the input to the Sort command, type the following:

```
C:\RUNDOS>dir c:\ | sort
```

If you're using MS-DOS 6.2 or later, add /-p /-w after *dir.* (The /-p ensures that the Directory command does not pause after one screenful of information and the /-w ensures that the directory entries are displayed in a single column.)

As you would expect, MS-DOS displays the directory sorted in alphabetic order. If the directory includes two files whose names look like 072F2321 or %PIPE1.$$$, don't be alarmed. These are temporary files MS-DOS creates in order to pipe the output of one command to the input of another; MS-DOS deletes the files automatically.

Now type this:

```
C:\RUNDOS>dir c:\ > sort
```

If you're using MS-DOS 6.2 or later, add /-p /-w after *dir.* (The /-p ensures that the Directory command does not pause after one screenful of information and the /-w ensures that the directory entries are displayed in a single column.)

This time you don't see anything on the screen because you told MS-DOS to redirect the output of the Directory command to a file named SORT. Confirm this by displaying the file with the Type command:

```
C:\RUNDOS>type sort
```

The file contains the directory, which is not sorted. You created the file when you redirected the output of the Directory command. You don't need this file, so delete it by typing *del sort.*

PREVIOUS NEXT

Editing an MS-DOS Command with Doskey

As you've seen, commands that redirect and filter input, output, or both can become long and complex, especially if you also include subdirectory names, file names, and extensions. By now you've also used many MS-DOS commands over and over again—to the point where you probably have no idea, other than "a lot," of the number of times you've typed *dir* followed by a drive letter, path name, or file name.

If you have version 5 or later, you can repeat (or edit) any recent command by using the small program named Doskey that you first encountered in Chapter 3, "Getting Your Bearings." (If you don't have version 5 or later, you don't have Doskey, so you can go on to Chapter 14, "Creating Your Own Commands.")

Typing *doskey* for the first time during a session with your computer causes MS-DOS to load the Doskey program into memory. Once in memory, Doskey keeps track of the commands you type and allows you to go back and review or repeat them by pressing the arrow and function keys.

But even though Doskey lets you reuse a command you typed earlier, you might want to change it slightly, to specify a different drive or file name. When you've recalled a previous command, Doskey lets you use the keys described in Figure 13-3 to edit the command.

Key	Action
Home	Moves the cursor to the beginning of the command
Ctrl-Home	Deletes from the cursor location to the beginning of the command
End	Moves the cursor to the end of the command
Ctrl-End	Deletes from the cursor location to the end of the command
Left arrow	Moves the cursor one character left
Ctrl-Left arrow	Moves the cursor one "word" (group of characters without spaces) left
Right arrow	Moves the cursor one character right
Ctrl-Right arrow	Moves the cursor one "word" (group of characters without spaces) right
	Adds typed characters at the location of the cursor the first time you press it;

Ins	overstrikes existing characters the next time you press it; toggles between these two actions
Del	Deletes the character at the location of the cursor; does not move the cursor
Esc	Erases the displayed command

Figure 13-3: The Doskey editing keys. A number of function keys are also available for editing; see the online Help Doskey.

To see how to edit commands with Doskey, start by loading the program into memory. Type this:

```
C:\RUNDOS>doskey
DOSKey installed.
```

Now type a command and press Enter to carry it out:

```
C:\RUNDOS>find "Smith" ph.txt | sort /+11
```

What if you wanted to type another command almost like that one? Instead of having to retype the whole thing, Doskey lets you recall the command to the screen, make any changes you like, then press Enter to carry out the command again. You start by using the Up arrow key to display the command you want to reuse, so press the Up arrow key. MS-DOS dutifully redisplays the Find command you just typed.

Your first edit will change *Smith* to *Black.* Press Home, which moves the cursor to the beginning of the command. Press Ctrl-Right arrow once to place the cursor under the quotation mark preceding *Smith,* and then press the Right arrow key to move the cursor under the *S* in *Smith:*

```
C:\RUNDOS>find "Smith" ph.txt | sort /+11
```

Now type *Black.* Notice that the characters you type replace the characters that were there. To insert characters rather than replace them, you would press Ins before beginning to type. To delete characters, you would press Del once for each character you wanted to remove.

Your command should look like this:

```
C:\RUNDOS>find "Black" ph.txt | sort /+11
```

Press Enter, and this time MS-DOS lists all people named Black, alphabetizing them by their first names. Press the Up arrow key again to display the last command. Now try moving the cursor around.

Press Home to move the cursor to the beginning of the command. Next, press Ctrl-Right arrow three times to move the cursor right three words:

```
C:\RUNDOS>find "Black" ph.txt | sort /+11
```

As a final exercise, try deleting parts of the command. Press Ctrl-End, and all characters from

the cursor to the end of the line vanish:

```
C:\RUNDOS>find "Black" ph.txt
```

Press Ctrl-Left arrow to move the cursor to the *p* in *ph.txt:*

```
C:\RUNDOS>find "Black" ph.txt
```

Press Ctrl-Home. This time, you delete all characters between the cursor and the beginning of the command:

```
C:\RUNDOS>ph.txt
```

What's left isn't very meaningful, so clean up by pressing Esc to erase the entire line:

```
C:\RUNDOS>_
```


PREVIOUS NEXT

Entering Multiple Commands with Doskey

In this chapter, you've seen many ways to use the redirection (< and >) and pipe (|) symbols to carry out more than one command at the same time. But what about other commands? While using MS-DOS you might sometimes have thought it would be nice to type two or more related (or unrelated) commands and have MS-DOS carry them out one after the other. With Doskey, you can type as many commands as you want, up to a maximum of 128 characters. To tell Doskey where one command ends and the next begins, you press Ctrl-T, which is displayed as a paragraph mark (¶).

To see how this feature works, go through the following example, which clears the screen, creates a new directory named TEST, makes TEST the current directory, and displays the directory listing. You'll enter all four commands on the same line, separating them by pressing Ctrl-T:

```
C:\RUNDOS>cls <Ctrl-T> md test <Ctrl-T> cd test <Ctrl-T> dir
```

When you press Enter, the screen clears and the commands are carried out one at a time. It happens quickly, but each command is displayed as it is carried out, so you can see the results:

```
C:\RUNDOS> md test

C:\RUNDOS> cd test

C:\RUNDOS\TEST> dir

 Volume in drive C is HARD DISK
 Volume Serial Number is 1608-5A30
 Directory of C:\RUNDOS\TEST

              01-05-95  11:27a
              01-05-95  11:27a
        2 file(s)        0 bytes
                  14069807 bytes free
```

The next example uses the Copy command to create a sample file by copying from the console (CON). Type this:

```
C:\RUNDOS\TEST>copy con example <Ctrl-T> type example
```

When you press Enter this time, the system pauses after the first command is displayed. MS-DOS is waiting for you to create the sample file, so type the following:

```
This is a sample file named EXAMPLE.
<Ctrl-Z>
```

When you press Enter, you see the familiar *1 file(s) copied* message followed by your

second command:

```
C:\RUNDOS\TEST> type example
This is a sample file named EXAMPLE.
```

This time, you created a file and displayed it.

Finally, clean up your directory by typing the following set of commands:

```
C:\RUNDOS\TEST>del example <Ctrl-T> cd .. <Ctrl-T> rd test
```


PREVIOUS NEXT

Chapter Summary

As you've seen, redirection, filter commands, and pipes let you create powerful, specific commands. When you consider these—and all the other MS-DOS commands you've encountered—you can begin to see ways in which you can adapt MS-DOS to your own needs and circumstances.

This chapter ends the part of the book that deals with MS-DOS as it is provided on disk. The next three chapters show you how to combine MS-DOS commands in sets known as *batch files* to put more of the power of MS-DOS to work. You have the building blocks. Now it's time to use them to customize MS-DOS.

PREVIOUS NEXT

Chapter 14: Creating Your Own Commands

Overview

As the preceding chapters show, MS-DOS gives you a great deal of control over your computer system. But MS-DOS is necessarily general purpose, because many people use it for many different tasks. So that you can adapt the computer to your work, MS-DOS lets you combine existing MS-DOS commands to create your own special-purpose commands.

The technique is simple: To make your own command, you create a text file that contains MS-DOS commands. And you can give such a file—called a *batch file*—any valid name except the name of an existing command; the extension of the file must be BAT. To use your command, simply type the name of the batch file; MS-DOS carries out the commands contained in the file as if you had typed each of them separately. Commands that you create in this way are called *batch commands.*

This chapter describes how to create batch files and batch commands. It also describes the Remark command, which is intended for batch files, and it shows you how to modify the MS-DOS startup procedure if there are certain commands you always want MS-DOS to carry out when you start your system.

PREVIOUS NEXT

A Batch of What?

The term *batch* has its origins in the early days of large computers, when most work was done by submitting a deck of punched cards to the data-processing department. The punched cards had to contain all the instructions required for the program to run correctly. There was no chance to interact with the system. The data-processing personnel ran these jobs in batches and delivered the output.

In effect, you do the same thing when you use a batch command because a batch file contains all the instructions needed to carry out a job. *Batch,* then, is used to describe a set of commands that are run one after the other. You can use batch files to automate frequently used command sequences and to make the system more accessible to colleagues who use application programs but might not know MS-DOS as well as you do.

PREVIOUS NEXT

How MS-DOS Searches for a Command

If you type something when MS-DOS is displaying the system prompt, MS-DOS assumes you have typed a command name. It then follows a particular sequence in trying to carry out the command:

1. It checks to see whether you typed the name of a built-in command, such as *dir* or *copy.* If you did, MS-DOS carries out that command.

2. If what you typed isn't the name of a built-in command, MS-DOS checks to see if you typed the name of a file with the extension COM or EXE (a command file). If you did, MS-DOS searches the current directory for the file and, if it finds the file, loads the program contained in the file and runs it.

3. If what you typed isn't the name of a command file, MS-DOS checks to see if you typed the name of a file with the extension BAT (a batch file). If you did, MS-DOS searches the current directory for the file and, if it finds the file, carries out the commands in the batch file.

4. If MS-DOS doesn't find the file in the current directory, it performs steps 2 and 3 of this sequence in each of the directories specified in the Path command.

The sequence is important because it explains why MS-DOS won't carry out a command file with the same name as a built-in command and why it won't carry out a batch file that has the same name as either a built-in command or a command file.

PREVIOUS NEXT

Creating the Sample Files

Change to the \RUNDOS directory:

```
C:\>cd \rundos
```

Because you'll be creating your own commands in this chapter, the system prompt will be shown as a simple C> to keep it unobtrusive. If your system prompt normally shows the current directory, you can simplify it temporarily by typing this:

```
C:\RUNDOS>prompt
```

When you next start or restart your computer, the system prompt will revert to showing both the current drive and the current directory.

You'll use three sample files in this chapter: LETR1.DOC, LETR2.DOC, and LETR3.DOC. Type the text shown below to create the sample files. (Remember, press either F6 or Ctrl-Z, and then press Enter where you see ^Z.)

```
C>copy con letr1.doc
This is the sample file.
^Z
        1 file(s) copied

C>copy letr1.doc letr2.*
        1 file(s) copied

C>copy letr1.doc letr3.*
        1 file(s) copied

C>_
```


PREVIOUS NEXT

Creating a Batch File

A batch file is simply a text file, whose extension is BAT, that contains MS-DOS commands. There are several ways to create a batch file. If the batch file is short and you're confident that it will work correctly, you can simply copy from the console to a file. If you think you might want to tinker with the file before saving it, you can use Edit or a word processor that can store files without inserting its own formatting codes. If you want to see how the commands work together, you can use Doskey for a test-as-you-go approach as described later in this chapter under the heading "Using Doskey to Create a Batch File."

Because the examples in this chapter are short and have already been checked for usability, you'll copy from the console to create your first batch files. You can't go back to correct an error after you press Enter at the end of a line, but if you make a typing error you can recover easily by pressing Ctrl-Break and reentering the batch file.

Suppose one of the application programs you use is a word processor, and you name the files that contain letters LETR1.DOC, LETR2.DOC, LETR3.DOC, and so forth. You use the Directory command fairly often to display the names of those particular files. Instead of typing *dir letr*.doc* each time, you could put the Directory command in a batch file named DIRLET.BAT.

Type the following to create the batch file:

```
C>copy con dirlet.bat
dir letr*.doc
^Z
       1 file(s) copied

C>_
```

The first line you typed names the batch file; the second line contains the command MS-DOS carries out. Test your batch command by typing its name:

```
C>dirlet

C>dir letr*.doc

 Volume in drive C is HARD DISK
 Volume Serial Number is 1608-5A30
 Directory of C:\RUNDOS

LETR1   DOC       26 01-05-95   3:13p
LETR2   DOC       26 01-05-95   3:13p
LETR3   DOC       26 01-05-95   3:13p
      3 file(s)         78 bytes
                 13277184 bytes free
```

It's possible that you'll see a double prompt like the following when your batch file finishes running:

```
C>
C>_
```

Don't worry about this prompt.

The first line displayed after you type your batch command is the Directory command that you entered into your batch file. MS-DOS displays the commands in a batch file as they are carried out, as if you had typed the commands yourself.

You could make the batch command even easier to type by naming the batch file simply LDIR.BAT. In the long run, however, it's usually better to make the name long enough to give a good hint of what the command does, especially if you create a large number of batch files.

Displaying Messages from a Batch File

The Remark (rem) command doesn't cause MS-DOS to do anything, but it is a valid command. You can include a message with the Remark command. The command form is this:

rem <message>

Although this command isn't especially useful at the MS-DOS command level, it lets you insert a message into a batch file. To see how the Remark command works, create another version of DIRLET.BAT that contains a descriptive message; type the following:

```
C>copy con dirlet.bat
rem DIRECTORY OF LETTERS
dir  letr*.doc
^Z
        1 file(s) copied
```

The new version of DIRLET.BAT replaces the first version you created a few minutes ago. (Your version of MS-DOS might ask whether you want to overwrite the file; choose *yes.*) Test this new version by typing this:

```
C>dirlet
```

The Remark command you included causes MS-DOS to display the message before it displays the directory shown on the following page.

```
C>rem DIRECTORY OF LETTERS

C>dir letr*.doc

 Volume in drive C is HARD DISK
 Volume Serial Number is 1608-5A30
```

```
 Directory of C:\RUNDOS

LETR1    DOC       26 01-05-95   3:13p
LETR2    DOC       26 01-05-95   3:13p
LETR3    DOC       26 01-05-95   3:13p
      3 file(s)         78 bytes
                  13277184 bytes free
C>_
```

Carrying Out the Same Batch Command with Different Data

You have seen that most MS-DOS commands include one or more parameters that you can use to make your instructions more specific. When you enter a Directory command, for example, you can specify a file name to display some portion of the files on a disk and the /W option to display the wide form of the directory. The Copy command is another example. It requires two parameters: the name of the file to be copied and the name to be given to the new copy.

Parameters let you use the same MS-DOS command with different data. You can give a batch file the same capability with a feature called a *replaceable parameter.* A replaceable parameter is a special symbol you put in a batch file. When you use the batch file, MS-DOS replaces the symbol with a parameter you include when you type the batch command. The symbol consists of a percent sign followed by a one-digit number, such as %1. You can use the numbers 1 through 9 in replaceable parameters, and you can include more than one replaceable parameter in a batch file. (MS-DOS also recognizes %0 as a replaceable parameter but reserves this parameter to mean the drive, path, and file name of the batch file itself.)

The number of the symbol identifies which parameter replaces the symbol. If a batch command takes two parameters, for example, MS-DOS replaces %1 wherever it occurs in the batch file with the first parameter you type with the batch command, and it replaces %2 with the second parameter you type. Replaceable parameters can be used anywhere in a batch command.

For example, suppose you wanted a batch command that would print a file by copying it to the printer. (You already have an MS-DOS Print command, but go through the example anyway; it illustrates the use of replaceable parameters.) All the batch file needs is a Copy command and one replaceable parameter that identifies the file to be printed. The batch command is called *Prnt* to avoid confusion with the Print command. Type the following:

```
C>copy con prnt.bat
copy %1 prn
^Z
      1 file(s) copied
```

To test your Prnt batch command, make certain the printer is turned on and type the following:

```
C>prnt letr1.doc
```

MS-DOS displays the command after replacing %1 with the batch-command parameter, LETR1.DOC, and prints the file:

```
C>copy letr1.doc prn
      1 file(s) copied
```

Figure 14-1 shows several versions of PRNT.BAT that you might create for printing other documents. Each version contains at least one replaceable parameter; the last version contains two. To the left of each version is an example of how the batch command would be typed, and to the right are the corresponding commands that would be carried out after MS-DOS replaced the replaceable parameters. The batch-command parameters, the replaceable parameters in each version of the batch file, and the result after MS-DOS replaces them with the batch-command parameters are shown in *italics.*

Batch Command You Would Type	Contents of PRNT.BAT	Commands That Would Be Carried Out
C>prnt *memo.doc*	copy *%1* prn	copy *memo.doc* prn
C>prnt *memo*	copy *%1*.doc prn	copy *memo*.doc prn
C>prnt *memo rept*	copy *%1*.doc prn copy *%2*.doc prn	copy *memo*.doc prn copy *rept*.doc prn

Figure 14-1: Replaceable parameters in a batch file.

Replaceable parameters make batch files much more flexible. Your batch commands needn't be limited to handling the same files or devices all the time—they can be used just like MS-DOS commands to operate with any file or device.

Canceling a Batch Command

As with other MS-DOS commands, you press Ctrl-Break to cancel a batch command. But when you cancel a batch command, MS-DOS prompts you to confirm. To see this, create a short new batch file named DIRS.BAT that first displays the entries in your \DOS directory and then displays the entries in the current directory (\RUNDOS).

Create the DIRS.BAT file by typing the following:

```
C>copy con dirs.bat
dir \dos
dir
^Z
      1 file(s) copied
```

The directory displays should be long enough to give you time to press Ctrl-Break. Type the

name of the Dirs batch command, then press Ctrl-Break as soon as MS-DOS starts displaying the file names:

```
C>dirs

C>dir \dos

 Volume in drive C is HARD DISK
 Volume Serial Number is 1608-5A30
 Directory of C:\DOS

.          <DIR>        01-15-94   6:05p
..         <DIR>        01-15-94   6:05p
EGA     SYS       4885   02-01-94  12:00a
DISPLAY  SYS      15682   02-01-94  12:00a
FORMAT   COM      32285   02-01-94  12:00a
PACKING  LST       3492   02-01-94^C

Terminate batch job (Y/N)?_
```

If you respond *n,* the command being carried out is canceled, but MS-DOS continues with the next command in the batch file. Type *n*; MS-DOS carries out the next command, which displays the current directory:

```
C>dir

 Volume in drive C is HARD DISK
 Volume Serial Number is 1608-5A30
 Directory of C:\RUNDOS

.          <DIR>        01-05-95   8:21a
..         <DIR>        01-05-95   8:21a
PH       TXT       526      01-05-95   1:43p
LETR1    DOC        26      01-05-95   3:13p
LETR2    DOC        26      01-05-95   3:13p
LETR3    DOC        26      01-05-95   3:13p
DIRLET   BAT        41      01-05-95   3:17p
PRNT     BAT        13      01-05-95   3:24p
DIRS     BAT        15      01-05-95   3:30p
       9 file(s)          673 bytes
                      13265422 bytes free
```

If you respond *y* to the "Terminate?" question, MS-DOS cancels the entire batch command and displays the system prompt. Type the Dirs batch command, and cancel it again, but this time respond *y*:

```
C>dirs
```

```
C>dir \dos

 Volume in drive C is HARD DISK
 Volume Serial Number is 1608-5A30
 Directory of C:\DOS

.          <DIR>        01-15-94   6:05p
..         <DIR>        01-15-94   6:05p
EGA      SYS     4885     02-01-94  12:00a
DISPLAY  SYS    15682     02-01-94  12:00a
FORMAT   COM    32285     02-01-94  12:00a
PACKING  LST     3492     02-01-94  12:00a
ANSI     SYS     8868     02-01-94^C

Terminate batch job (Y/N)?y
```

MS-DOS returns to command level without completing the batch command.

PREVIOUS NEXT

Developing Your Own Startup Procedure

Each time you start or restart the system, MS-DOS goes through a startup procedure that includes searching the root directory of the startup disk for a special batch file named AUTOEXEC.BAT. If it finds the file, MS-DOS carries out whatever commands the file contains. Typically, AUTOEXEC.BAT is used to hold the commands you don't want to have to type each time you start or restart the system—a Path command, for example, that tells MS-DOS where to find command files on a hard disk and a Prompt command that sets the system prompt to show the current directory.

Versions 4 and later of MS-DOS create AUTOEXEC.BAT (or modify the existing version) as part of the installation procedure. Earlier versions of MS-DOS don't create the file automatically, but because AUTOEXEC.BAT is so useful, most systems include one.

Although MS-DOS gives special treatment to AUTOEXEC.BAT, that doesn't mean the file is untouchable. You can add commands to it at any time, but you should always be careful not to change or delete any existing commands, especially those you don't clearly understand. (You *can* encounter such commands in AUTOEXEC.BAT, particularly on systems that connect to a network and those that have been set up by someone else.)

The following examples show you some commands you might want to include in AUTOEXEC.BAT to tailor MS-DOS and your computer more closely to your needs or preferences. If you follow these examples, however, you'll be replacing the AUTOEXEC.BAT file you have now, so before you start, you'll check for, and protect, your existing file.

A Safety Check

It's easy to check for AUTOEXEC.BAT. Remember, it must always be in the root directory of the system disk, so a simple Directory command does the job. The current directory should still be C:\RUNDOS, so type the following command to check the root directory:

```
C>dir \autoexec.bat
```

If MS-DOS responds *File not found,* you don't have an AUTOEXEC.BAT file, and you won't hurt a thing by trying the following examples. If, however, MS-DOS responds by showing an entry for AUTOEXEC.BAT, type the following to rename your existing file so that you won't lose it:

```
C>ren \autoexec.bat autoexec.sav
```

Type the Directory command again to search for AUTOEXEC.BAT in the root directory. This time, MS-DOS should respond *File not found*.

You'll restore your original AUTOEXEC.BAT later.

Creating an AUTOEXEC.BAT File

Depending on how your system is set up and what you want to do with it, an AUTOEXEC.BAT file can contain anywhere from a few to many commands. As mentioned earlier, however, two that are usually included are a Path command that tells MS-DOS where to find command files and a Prompt command that sets the system prompt to display the current directory.

Although your normal AUTOEXEC.BAT file might well include more than these two commands, the following example shows you how to create just such a simple AUTOEXEC.BAT file. The file is short, so copy from the console to create it. Type the following:

```
C>copy con \autoexec.bat
rem SAMPLE STARTUP PROCEDURE
path  c:\;c:\dos;c:\rundos
prompt $p$g
^Z
```

You created the file in the root directory of the system disk because that's where MS-DOS always looks for AUTOEXEC.BAT. To test your startup procedure, you must restart the system.

Note Be sure to open the latch or remove any floppy disk in the floppy disk drive so that MS-DOS restarts from the hard disk.

Now restart the system by pressing Ctrl-Alt-Del. MS-DOS might display some messages before carrying out the commands in your new AUTOEXEC.BAT file, but regardless of your version of MS-DOS, the end of the startup screen looks like this:

```
C>rem SAMPLE STARTUP PROCEDURE

C>path c:\;c:\dos;c:\rundos

C>prompt $p$g

C:\>_
```

So far so good, but, as you've probably realized, a simple batch file like this just begins to tap the power of AUTOEXEC.BAT. Suppose, now, that you want MS-DOS to clear the screen and display its version number each time you start your computer. You also want MS-DOS to change to a particular directory and display a directory listing. Here's a new AUTOEXEC.BAT file that does what you want. You can use Edit or a word processor if you want to try your editing skills. Otherwise, simply copy from the console again (as shown in the example) to replace the sample AUTOEXEC.BAT file with a new version.

Type the following:

```
C:\>copy con autoexec.bat
rem SAMPLE STARTUP PROCEDURE
```

```
path c:\;c:\dos;c:\rundos
prompt $p$g
cls
ver
cd \rundos
dir
^Z
```

Again, restart the system to test your new AUTOEXEC.BAT file. This time, you see (though briefly) this:

```
C>rem SAMPLE STARTUP PROCEDURE

C>path c:\;c:\dos;c:\rundos

C>prompt $p$g

C>cls
```

The screen clears, and then MS-DOS displays this:

```
C:\>ver

MS-DOS Version 6.0

C:\>cd \rundos

C:\RUNDOS>dir

 Volume in drive C is HARD DISK
 Volume Serial Number is 1608-5A30
 Directory of C:\RUNDOS

.          <DIR>        01-05-95   8:21a
..         <DIR>        01-05-95   8:21a
PH       TXT       526     01-05-95   1:43p
LETR1    DOC        26     01-05-95   3:13p
LETR2    DOC        26     01-05-95   3:13p
LETR3    DOC        26     01-05-95   3:13p
DIRLET   BAT        41     01-05-95   3:17p
PRNT     BAT        13     01-05-95   3:24p
DIRS     BAT        15     01-05-95   3:30p
       9 file(s)          673 bytes
                      13265422 bytes free
```

Your new startup procedure not only tells MS-DOS where to find the files it needs, it also

clears the screen, displays the version number, changes the current directory, and then displays the entries in the current directory.

You can also use AUTOEXEC.BAT to handle special startup requirements. For example, if your system has equipment options that require special setup instructions with the Mode command, put the commands in AUTOEXEC.BAT so that you needn't type them each time you start or restart the system. Or, if you have a device attached to your system that requires a special program to operate it (such as a CD-ROM drive or a scanner), you can put the command in your AUTOEXEC.BAT file so that you don't have to enter the command each time you start the system. And by all means, put the command to load Doskey in AUTOEXEC.BAT if it isn't there already.

If you renamed your existing AUTOEXEC.BAT file at the beginning of this example, type the following to save the example as AUTOEXEC.TST and restore the original:

```
C:\RUNDOS>ren \autoexec.bat  autoexec.tst

C:\RUNDOS>ren \autoexec.sav  autoexec.bat
```

Now your real AUTOEXEC.BAT will be executed the next time you start up your system. The version you created in the example is still in the root directory as AUTOEXEC.TST, in case you want to experiment with it further.

If you didn't have an AUTOEXEC.BAT file to begin with, type the following so that MS-DOS won't carry out your example file the next time you start your system:

```
C:\RUNDOS>ren autoexec.bat autoexec.tst
```

The version you created in the example is still present in the root directory as AUTOEXEC.TST, in case you want to experiment with it further.

Finally, to rid your system of a possibly invalid Path command, restart your computer by pressing Ctrl-Alt-Del.

PREVIOUS NEXT

Some Useful Commands

The following examples describe a few batch commands you might find useful; they might also give you some ideas for other commands you could create. Each topic includes a description of what the command does, the contents of the batch file, and one or two examples of its use.

These examples are illustrative; they are not hands-on exercises because you might not have the necessary files or devices to use them. But remember that they're here; you'll probably find a situation in which they can be helpful.

Printing a File

Earlier in the chapter you created the Prnt batch command, which prints a file by copying it to the printer. PRNT.BAT contains *copy %1 prn*. To use it, you type the name of the batch file followed by the name of the file to be printed. (For example, to print a file named REPORT.DOC, you would type *prnt report.doc*.)

As the examples in Figure 14-1 showed, you can make batch commands more specific by including common parts of a file name or extension in the batch file. For example, if you frequently print files that have the extension DOC, you could enter the command in the batch file as *copy %1.doc,* to print the file named REPORT.DOC, you would only have to type *prnt report*.

Printing a File in Small Type

If your printer is compatible with Epson or IBM dot-matrix printers, you can use the Mode command to print 132 characters on a line. This type size is handy for wide reports or spreadsheets, and putting the Mode and Copy commands in a batch file named SMALL.BAT makes them easy to use. SMALL.BAT contains:

```
mode lpt1:  132
copy %1  lpt1:
mode lpt1:  80
```

If your printer is attached to a different port, change *lpt1* to *lpt2* or *lpt3,* as necessary. The Small command takes one parameter, the name of the file to be printed in small type. To use the Small command to print the file REPORT.DOC, you would type

```
C:\>small  report.doc
```

MS-DOS would display the Mode and Copy commands and print the file in small type:

```
C:\>mode lpt1: 132

LPT1: not rerouted

LPT1: set for 132
```

```
No retry on parallel printer time-out

C:\>copy report.doc  lpt1:
        1 file(s) copied

C:\>mode lpt1: 80

LPT1: not rerouted

LPT1: set for 80

No retry on parallel printer time-out
```

The second Mode command resets the printer to normal type.

Eliminating Old BAK Files

Because many word processors create a backup file with an extension of BAK each time you edit a file, your disks can get crowded with files you might not need. The two batch files here work together to let you find all the BAK files on your disk and erase the ones you don't need.

First, FINDBAK.BAT consists of one Directory command that uses the /S parameter to look in all subdirectories of the root directory and the /B parameter to display only the path and file name of each file it finds:

```
dir \*.bak /s /b
```

The backslash before **.bak* starts the search in the root directory and, because all other subdirectories are contained in the root, the /S parameter makes the command find every BAK file on the disk. Use the Findbak command to locate the files you want to delete.

Next, CLEAN.BAT consists of one Delete command that erases all files in the specified directory that have the extension BAK:

```
del %1 *.bak
```

The Clean command takes one parameter, the directory name followed by a backslash. If you omit the parameter, the command cleans up the current directory. You can clean up the disk in a different drive by preceding the path name with the drive letter and a colon. For example, after using FINDBAK.BAT to find your BAK files, you would erase all files whose extension is BAK in the current directory, simply by typing *clean.* If you wanted to erase all the BAK files in the directory \MKT\WP, you would type *clean \mkt\wp\.* To erase all BAK files in the directory \LASTYEAR\RPT on the disk in drive A, you would type *clean a:\lastyear\rpt\.*

Creating Commands with Doskey

Note Doskey is included with versions 5 and later of MS-DOS. If you have an earlier version of MS-DOS, go on to Chapter 15, "Creating Smart Commands."

You've seen how batch files can help you customize MS-DOS by combining commands to perform special tasks. To create a batch file, you can copy from the console, as in the preceding examples, or use a text editor or a word processor. If you have version 5 or later, you can also use Doskey.

Chapters 3 and 13 showed how Doskey can help you edit and repeat MS-DOS commands you've already used. This section introduces several Doskey parameters that help you not only view but save commands in either of two forms: as batch files or as keyboard shortcuts called *macros.*

Batch Files and Macros

You know that batch files are sets of MS-DOS commands that you save in a text file and carry out by typing the name of the batch file. A macro is a similar type of work saver. You create a macro by assigning a descriptive name to one or more commands. Once you've defined a macro in this way, you simply type the name to carry out the commands. You benefit by saving time and keystrokes, especially with often-used, long, or complicated sets of commands.

In some respects, macros are very much like batch files. Both are your creations; both cause MS-DOS to carry out one or more commands to do a specific job; and both let you start the command sequence by typing a short, descriptive name.

Doskey Command Parameters

In earlier chapters, you've typed *doskey* to start the Doskey program, and you've used function and editing keys to retrieve, display, and edit commands. Doskey also includes several parameters you can use to create and view commands and macros. Three important ones are these:

doskey /history <macro>=<command> /macros

Doskey is the name of the program. When you type *doskey* by itself, MS-DOS loads the program into memory, where it remains until you turn the system off or restart MS-DOS.

/history, which you can abbreviate as */H,* tells Doskey to show you a list of all commands currently in memory. You can use the */history* parameter to create a batch file by redirecting the commands from memory to a file.

<macro>=<command> tells Doskey to create a macro. As already mentioned, you assign a name (shown here as *macro*) to one or more commands (*command*). An equal sign must follow the macro name.

/macros, which you can abbreviate as */M,* tells Doskey to display the macros currently in memory. You can use the */macros* parameter to save macros by redirecting them from the computer's memory, where they will be lost when you shut down the system, to a file.

Doskey includes a few other parameters you can use to control its behavior. You don't need these parameters here; the details are in Appendix C, "MS-DOS Command Reference."

Using Doskey to Create a Batch File

When you use Doskey, you can retrieve and edit commands you typed earlier because the commands remain in a special portion of your computer's memory until you turn off the computer or restart Doskey, or until you've typed so many commands that the oldest ones are discarded to make room for newer ones.

Because commands remain in memory, you can use Doskey as an alternative to a text editor for creating some batch files. The following example shows how to redirect Doskey's list of commands to a batch file. Begin by starting Doskey. Type this:

```
C:\>doskey
```

(If Doskey is already running, press Alt-F7. This clears the list of commands Doskey has recorded and assures you of a fresh start.)

Now type some commands:

```
C:\>cd \rundos

C:\RUNDOS>dir /oe > prn

C:\RUNDOS>cd \
```

These three commands change the current directory, print a directory listing sorted by extension, and return to the root directory of drive C. These commands could form a small but useful batch file for keeping track of the files you create with your applications, especially if you can specify the directory you want to list and print. You can create the file in a two-step process.

First, use the Doskey /H parameter to redirect the commands to a file. This example uses the extension BAT, but you could just as easily save the commands as a standard text file with an extension such as TXT or DOC. Type this:

```
C:\>doskey /h > list.bat
```

That's really all there is to using Doskey to save a sequence of commands you've typed. If you want to save the commands as a usable batch file, however, your work isn't quite done, so use Edit to add a few finishing touches. Type this:

```
C:\>edit list.bat
```

When Edit displays the file, change \RUNDOS to a replaceable parameter so you can specify any directory you choose. Change this:

```
cd \rundos
```

to this:

```
cd %1
```

Notice, too, that Doskey has included the command you typed to create the batch file. You have to omit this command to avoid accidentally redirecting some future list of unrelated Doskey commands to your batch file, so delete the line that contains *doskey /h > list.bat.* Save the file, and you're done. From now on, whenever you want to print the directory entries of the files in a particular directory, sorted by extension, just type *list* followed by the name of the directory.

Using Doskey in this way can be especially useful when you're developing a batch file and aren't certain the commands you're putting together will do the job you have in mind. Start Doskey, and type each command you intend to use. Because each command is carried out immediately, you can see exactly what it does, and you can see how all the commands work together before you save them as a batch file.

Using Doskey to Create a Macro

Creating a macro with Doskey is just as easy as creating a batch file, perhaps even easier. To define a macro to Doskey, you type a macro name, an equal sign, and the command or commands you want the macro to carry out. You can use replaceable parameters as you do in batch files, but instead of %1, %2, %3, and so on, you use $1, $2, $3, and so on through $9.

The following examples use the Directory and Mode commands to create some macros you might want to use.

In versions 5 and later, the /O parameter of the Directory command allows you to sort a directory in a number of ways: by name, extension, size, or date and time, or with subdirectories grouped at the beginning of the listing. Depending on what you're doing, you might choose to look at a directory in any of these ways. To keep typing to a minimum, you can create a macro for each type of listing and add the /P parameter so that MS-DOS will pause after each screenful.

If you haven't turned off or restarted your computer, clear Doskey's memory by pressing Alt-F7. If you have turned off your computer, turn it back on, and type *doskey* to load the program. Now create the macros by typing the following:

```
C:\>doskey dname=dir $1 /on /p

C:\>doskey dext=dir $1 /oe /p
```

```
C:\>doskey dsize=dir $1 /os /p

C:\>doskey ddate=dir $1 /od /p

C:\>doskey dsubs=dir $1 /og /p
```

Now try them out. Using your \DOS directory as an example, type *dname \dos* to list the entries alphabetically by file name, *dext \dos* to list them by extension, *dsize \dos* to list them by size, and *ddate \dos* to list them by date and time. If you want, type *dsubs c:* to list the subdirectories in the current directory on drive C before the files.

If you have a monitor that can change between a normal display (25 lines) and a condensed display (43 or 50 lines) with the Mode command, you can create macros that switch you quickly back and forth. You might, for example, prefer a condensed display for directory listings but a regular 25-line display for most other work. Type the following:

```
C:\>doskey big=mode con lines=25

C:\>doskey little=mode con lines=43
```

Now, when you want a condensed display, simply type *little.* To switch back to 25 lines, type *big.* (If you try these examples and MS-DOS responds *ANSI.SYS must be installed to perform requested function,* you need to identify a file named ANSI.SYS to MS-DOS as part of your startup procedure. Chapter 17, "Tailoring Your System," tells you how.)

You can also combine the screen macros with the directory-display macros by taking advantage of Doskey's ability to accept more than one command on a line. At the system prompt, you separate commands by pressing Ctrl-T. In a macro, you do the same thing by typing *$T* (or $t). For example, the following macro changes the screen to a condensed display, lists by size the entries in the directory you specify, waits for you to press a key, and then returns to a 25-line display. Note the *$t* separating the Mode, Pause, and Directory commands. Although the macro is shown on two lines here, don't press Enter until you reach the end:

```
doskey dir43=mode con lines=43 $t dir $1 /os /p
$t pause $t mode con lines=25<enter>
```

To see a condensed listing of your \DOS directory, you would type *dir43 \dos.*

Saving Macros for Later Use

Although macros act much like batch files, they're stored in your computer's memory, not on disk; they aren't saved when you turn off or restart the computer. You can save the macros you create by using either the /H or the /M parameter to redirect the macros into a file. The difference between the two parameters is important:

- If the macro-definition commands are in the list of commands Doskey can retrieve, /H saves the *commands* that create the macros.

- If the macros are currently in your computer's memory, /M saves the macros.

To see how these parameters work, first use the /H parameter to save the macro definitions currently in your computer's memory. Type this:

```
C:\>doskey /h > macros1.txt
```

Now, save the macros themselves by typing this:

```
C:\>doskey /m > macros2.txt
```

Use the Type command to view the two files you just saved. The commands saved with /H look something like this (you probably won't see them all if you've been trying out the macros because other commands will take up space in the list):

```
doskey dname=dir $1 /on /p
doskey dext=dir $1 /oe /p
doskey dsize=dir $1 /os /p
doskey ddate=dir $1 /od /p
doskey dsubs=dir $1 /og /p
doskey big=mode con lines=25
doskey little=mode con lines=43
doskey dir43=mode con lines=43 $t dir $1 /os /p
  $t pause $t mode con lines=25
doskey /h > macros1.txt
```

The file saved with /M looks like this:

```
DNAME=dir $1 /on /p
DEXT=dir $1 /oe /p
DSIZE=dir $1 /os /p
DDATE=dir $1 /od /p
DSUBS=dir $1 /og /p
BIG=mode con lines=25
LITTLE=mode con lines=43
DIR43=mode con lines=43 $t dir $1 /os /p
  $t pause $t mode con lines=25
```

The first file contains the *commands* that created the macros. You can easily turn such a file into a batch file by entering the macro-definition commands, giving the file a BAT extension, and deleting the last line (which will be something like *doskey /h > macros1.txt*).

The second file contains the *macros* you created. It forms a useful record of your macros, but in order to run the macros again in your next session with MS-DOS, you must edit the file, adding *doskey* in front of each one to turn the macro into the command that creates it. You can then turn the file of macro-definition commands into a batch file by giving the file a BAT extension.

Thus, although the /M parameter might appear to be more useful because it saves the actual macros, the reverse is true. Each time you start MS-DOS, you must also define any macros you want to use. If you're going to save a macro, you'll want to save the macro definition because you must run the command that creates the macro before you can use the macro itself. You can put a macro-definition command in a batch file and run it without any problems. You cannot, however, run the macro itself from a batch file, even if you first define it in the same batch file. Macro names can be typed only at the MS-DOS system prompt.

PREVIOUS NEXT

Chapter 15: Creating Smart Commands

Overview

The previous two chapters showed how redirection, filter commands, pipes, batch files, and macros let you build your own commands or change the way that MS-DOS commands work. This chapter shows how MS-DOS gives you more control over the way it carries out the commands you build into a batch file. The techniques in this chapter help you create powerful commands tailored to your needs.

You can make your commands display their own instructions or warning messages. Or you can specify the circumstances under which MS-DOS carries out one command—or a different sequence of commands altogether. You can even make the system pause until you tell it to proceed.

This chapter shows you how to develop a batch file that uses most of these capabilities. The next chapter extends the example, describes two additional batch commands, and shows several useful commands you can create. When you complete these two chapters, you'll be ready to apply the full power of MS-DOS to your needs.

PREVIOUS NEXT

Preparing for the Examples

For these examples, you'll need a formatted floppy disk and some sample files. If you completed the examples in Chapter 13, "Taking Control of Your System," use the \RUNDOS directory, which contains the phone-list file. Put the formatted floppy disk in drive A, and change the current directory to \RUNDOS by typing this:

```
C:\>cd \rundos
```

If you want, change the prompt to C>, as shown in the following examples, by typing *prompt*. You'll use six sample files in this chapter and the next chapter:

P.DOC Q.DOC

P.BAK Q.BAK

P.OLD Q.OLD

Type the following to create the files (as usual, press F6 or Ctrl-Z and Enter where you see ^Z):

```
C>copy con p.doc
This is a sample file.
^Z
        1 file(s) copied

C>copy p.doc p.bak
        1 file(s) copied

C>copy p.doc p.old
        1 file(s) copied

C>copy p.* q.*
P.DOC
P.BAK
P.OLD
        3 file(s) copied
```

This completes the preparation for the examples.

Creating an Archive Command

Disk files proliferate as you use your computer. You'll probably archive files from time to time, copying them to long-term-storage floppy disks and then erasing them from your working disks, just as you occasionally remove documents from your paper files and put them in long-term storage.

Although MS-DOS version 6 includes the MSbackup command and earlier versions include the Backup and Restore commands to help you archive old files, this chapter shows you how to use the MS-DOS batch commands to develop an Archive command that all users can use, whether or not they're comfortable with MS-DOS commands.

You start with a batch file that contains a single Copy command, and then you expand the batch file to display instructions, provide a safeguard against inadvertently erasing a previously archived file, and erase the file after it is archived.

The floppy disk in drive A is the archive floppy disk, the one you store in a safe place. Your hard disk is the working disk that contains files you want to archive.

Your Archive command will be a file named ARCHIVE.BAT. The initial version contains only a Copy command. To create the file, type the following:

```
C>copy con archive.bat
copy %1 a:%1
^Z
        1 file(s) copied
```

This file is the starting point for your Archive command. It requires only one parameter, the name of the file to be archived. The Copy command copies this file (*%1* in the Copy command) to the floppy disk in drive A, giving it the same name (*a:%1*) as the original. To see how the batch file works, make certain the archive floppy disk is in drive A; archive P.DOC by typing this:

```
C>archive p.doc
```

MS-DOS responds by displaying and carrying out the Copy command in the batch file:

```
C>copy p.doc a:p.doc
        1 file(s) copied
```

This isn't much, but then it's only a starting point. You'll add significantly to this simple command as you go through the examples in this chapter.

PREVIOUS NEXT

Modifying the Sample Batch File

This chapter describes four batch commands: Echo, Pause, If, and Goto. Each command description explains the purpose of the batch command, then adds it to ARCHIVE.BAT. The modified version of the batch file is shown with the changed or added lines shaded. Either use the MS-DOS editor (Edit) to make the indicated changes or enter each updated version by copying from the console.

PREVIOUS NEXT

Controlling System Messages

The Echo command controls whether commands in a batch file are displayed, and it lets you display your own messages. It has three parameters:

echo on off <message>

on causes commands to be displayed as they are carried out. (This is called "turning echo on.") Echo is on unless you turn it off.

off causes commands not to be displayed as they are carried out. (This is called "turning echo off.") Eliminating the commands from the display can make a batch file easier to use by reducing clutter on the screen.

<message> is a string of characters, such as a reminder or a warning, that you want to display. <message> is displayed whether echo is turned on or off.

You can include only one parameter with an Echo command. If you omit all parameters (just type *echo*), MS-DOS displays the status of echo (either *ECHO is on* or *ECHO is off*).

The first changes to your Archive command add an Echo command at the beginning to turn echo off and another Echo command after it to display a title message.

Here is the modified version of ARCHIVE.BAT:

```
echo off
echo Archive Procedure
copy %1 a:%1
```

To archive P.DOC with your new Archive command, type the name of the batch file and the name of the file to be archived:

```
C>archive p.doc
```

Because echo is always on when MS-DOS starts carrying out the commands in a batch file, MS-DOS displays the first Echo command, which turns echo off. Then it carries out the second Echo command, which displays the title *Archive Procedure*. Finally, it carries out the Copy command from the original ARCHIVE.BAT and copies the file:

```
C>echo off
Archive Procedure
      1 file(s) copied
```

Messages produced by a command itself (such as *1 file(s) copied* in the preceding example) are displayed whether echo is on or off.

Starting with version 3.3, you can prevent any individual command from being echoed by preceding it with the @ symbol. This means that you can eliminate even that first *echo off* message by changing the first line of ARCHIVE.BAT to *@echo off*. If you're using 3.3 or a

later version, edit the file again to add the @ symbol at the beginning of line 1. From now on, all examples will show the @ symbol; if you're not using 3.3 or a later version, simply ignore it.

Your Archive command is starting to take shape. It's time to add some instructions for the person who uses it.

PREVIOUS NEXT

Making the System Pause

Some MS-DOS commands, such as Format and Diskcopy, display a message and wait for you to respond, giving you a chance to confirm your intention or to complete preparation by inserting a floppy disk or by turning on the printer. You can have your batch files do the same by using the Pause command, which displays the message *Press any key to continue...* and makes the system wait until you press any key. (The message is *Strike a key when ready...* in versions prior to 4.)

The format of the Pause command is as follows:

pause <message>

You'll add a Pause command to ARCHIVE.BAT now. You'll also add a message—a reminder to make certain that the archive floppy disk is in drive A before the file is copied.

The modified version of ARCHIVE.BAT is this:

```
@echo off
echo Archive Procedure
echo Make sure archive floppy disk is in drive A
pause
copy %1 a:%1
```

Remember, if you're not using version 3.3 or later, ignore the @ in line 1. Test this version by typing the following:

```
C>archive p.doc
Archive Procedure
Make sure archive floppy disk is in drive A
Press any key to continue . . .

_
```

The Echo command displays the reminder, the Pause command displays its message telling you to press a key, and the system waits. Complete the command by pressing any key. MS-DOS copies the file and acknowledges this:

```
1 file(s) copied
```

Controlling Which Commands Are Carried Out

Besides carrying out MS-DOS commands as though you had typed them individually, batch files let you specify that a command should be carried out only if some condition is true—for example, only if a specific file exists. This capability makes your batch files more flexible, letting you adapt them to a variety of situations.

The If command specifies the condition to be checked and the MS-DOS command to be carried out. It has three parameters:

if not <condition> <command>

not reverses the meaning of the If command so that <command> is carried out only if <condition> is not true.

<condition> is the condition to check. It has two commonly used forms:

- *exist* <filename> checks whether the named file exists. You can include a path name, if necessary. If <filename> exists, the condition is true.
- <string1> = = <string2> compares the two character strings you specify. If they are identical, the condition is true. Note that there are two equal signs.

<command> is any MS-DOS command.

You'll add an If command to ARCHIVE.BAT to control when it displays a warning message.

Adding Protection to Your Archive Command

MS-DOS 6.2 and later warn you if you attempt to copy a file onto another with the same name. For earlier versions of MS-DOS, the following exercise can provide a valuable tool. For users of all versions, it is a good learning exercise.

When you copy a file with the Copy command, you tell MS-DOS the name of the original and the name to be given to the new copy. MS-DOS checks to see whether there is already a file with the same name as the copy on the target disk; if there is, MS-DOS replaces the existing file with the copy. What if you didn't realize that a file with the same name existed on the disk? You might inadvertently lose a valuable file.

To protect yourself against such an oversight, you'll include two more commands in ARCHIVE.BAT. First, you'll add an If command that checks to see whether the file to be archived already exists on the archive floppy disk in drive A. If it does, an Echo command in the second part of the If command displays a warning message telling you that you can cancel the command by pressing Ctrl-Break. You'll also add a Pause command to give you time to read the warning message and, if necessary, cancel the Archive command by pressing Ctrl-Break.

Here is the modified version of ARCHIVE.BAT. Type the fifth and sixth lines on one line, even

though they are shown on two lines in the text:

```
@echo off
echo Archive Procedure
echo Make sure archive floppy disk is in drive A
pause
if exist a:%1 echo a:%1 exists. Press
CTRL-BREAK to cancel, or
pause
copy %1 a:%1
```

Test this version of your Archive command by typing the following:

```
C>archive p.doc
Archive Procedure
Make sure archive floppy disk is in drive A
Press any key to continue . . .
_
```

Press any key to continue the command:

```
a:p.doc exists. Press CTRL-BREAK to cancel, or
Press any key to continue . . .
_
```

Don't press any key yet.

You might not want to replace the copy of the file in drive A. Press Ctrl-Break to cancel the command. MS-DOS asks whether you really mean it:

```
Terminate batch job (Y/N)?_
```

Type *y* to confirm. MS-DOS cancels the rest of your Archive command and displays the system prompt without copying the file. If you had not pressed Ctrl-Break, the file would have been copied as before, erasing the previously archived version of P.DOC.

Smoothing a Rough Edge

Your Archive command is getting more useful; you've protected yourself from inadvertently erasing an existing file. But there's a problem now. See what happens if the name of the file to be archived isn't the name of a file already on the floppy disk in drive A. Delete P.DOC from drive A, and archive it again:

```
C>del a:p.doc

C>archive p.doc
Archive Procedure
Make sure archive floppy disk is in drive A
Press any key to continue . . .
```

_

Fine so far. Press the Spacebar to continue:

```
Press any key to continue . . .
_
```

That's not too good. The system comes right back and pauses again, even though everything is OK, because the Pause command that follows the If command is always carried out. Press the Spacebar again:

```
1 file(s) copied
```

MS-DOS copies the file as it should. Your Archive command is working properly, but the two pauses could be confusing, especially if someone else uses your batch command. How can you fix this?

You could delete the first Pause command, but if you did, you wouldn't have a chance to make certain that the correct floppy disk had been placed in drive A. That's not a very good solution.

Or you could delete the second Pause command and change the If command, making the "file exists" warning a Pause message instead of an Echo message, as it is now. But then, recall that you would have to delete the command *@echo off* at the beginning of the batch file so that the message following the new Pause command would be displayed. And that would mean all your commands in the file would be displayed. This solution would make the response to your Archive command cluttered and confusing, so it isn't very good either.

There is, however, a way to change your Archive command so that the second Pause command is carried out only if the file to be archived is already on the floppy disk in drive A. This solution requires using another command for batch files—the Goto command.

Changing the Sequence of Commands

The batch files you have created up to now carry out the MS-DOS commands they contain in the order in which the commands appear. Your batch commands would be more flexible if you could control the order in which the commands are carried out. The Goto command gives you this control by telling MS-DOS to go to a specific line in the command file, rather than to the next command in the sequence.

You tell MS-DOS where to go in the batch file by specifying a *label*. A label, which is located on a line in a batch file, consists of a colon (:) immediately followed by a string of characters (for example, :*start*). A label is not a command. It merely identifies a location in a batch file. When MS-DOS goes to a label, it carries out whatever commands follow the line on which the label appears.

The Goto command has one parameter:

goto <label>

<label> is the label that identifies the line in the batch file where MS-DOS is to go.

The Goto command is often used as part of an If command. For example, the If command checks some condition. If the condition is not true, MS-DOS carries out the next command; if the condition is true, MS-DOS carries out the Goto command and moves to some other part of the batch file.

Remember the problem with your Archive command? If the file to be archived is already on the floppy disk in drive A, you want MS-DOS to display a warning message and pause; otherwise, you just want to copy the file. This situation is tailor-made for an If command that includes a Goto command.

The modified version of ARCHIVE.BAT is this:

```
@echo off
echo Archive Procedure
echo Make sure archive floppy disk is in drive A
pause
if not exist a:%1 goto safe
echo a:%1 exists. Press CTRL-BREAK to cancel, or
pause
:safe
copy %1 a:%1
```

These changes warrant a bit more explanation:

- The If command still checks whether the file to be archived exists on the floppy disk in drive A. Now, however, the command includes the parameter *not*, which means that the command in the second part of the If command is carried out only if the reverse of the condition is true (that is, if the file does *not* exist on the floppy disk in

drive A).

- The second part of the If command—the command to be carried out if the condition is true—is changed to a Goto command that tells MS-DOS to skip to a label called :*safe*.

- If the file to be archived doesn't exist on the floppy disk in drive A, MS-DOS carries out the Goto command and then jumps to the label :*safe*, skipping the Echo and Pause commands. The Copy command following :*safe* copies the file.

- If the file to be archived does exist on the floppy disk in drive A, the Goto command is not carried out and MS-DOS continues with the Echo and Pause commands. If you don't cancel by pressing Ctrl-Break, MS-DOS carries out the Copy command. In this instance, MS-DOS ignores the line that contains the label :*safe* because the only purpose of the label is to identify a location in a batch file.

P.DOC is on the floppy disk in drive A (you archived it again a bit earlier), so first see if this version of your Archive command warns you that the file exists. Type the code on the following page.

```
C>archive p.doc
Archive Procedure
Make sure archive floppy disk is in drive A
Press any key to continue . . .

_
```

Press any key to continue the command:

```
a:p.doc exists. Press CTRL-BREAK to cancel, or
Press any key to continue . . .

_
```

There's the warning. Press any key to complete the command:

```
1 file(s) copied
```

But the problem came up when the file wasn't on the archive floppy disk: You got the second pause anyway. Test your revised Archive command in this situation by archiving P.BAK, which hasn't yet been archived:

```
C>archive p.bak
Archive Procedure
Make sure archive floppy disk is in drive A
Press any key to continue . . .

_
```

Press a key:

```
1 file(s) copied
```

Problem solved. Because P.BAK wasn't on the archive floppy disk in drive A, the Goto command was carried out; the Goto command caused MS-DOS to skip over the intervening Echo and Pause commands and to copy the file.

Your Archive command works properly. You had to do a little more work to avoid the double pause, but now the command is less confusing and easier to use. You'll probably encounter this kind of circumstance fairly often as you create batch commands. Just remember: It takes a little more time to make a command easy to use, but the investment is usually worthwhile, especially if other people will be using the command.

Using Wildcard Characters with a Batch File

You can use wildcard characters to archive a series of files with your Archive command. Type the following command to archive all the files named P:

```
C>archive p.*
Archive Procedure
Make sure archive floppy disk is in drive A
Press any key to continue . . .
_
```

Press any key:

```
a:p.* exists. Press CTRL-BREAK to cancel, or
Press any key to continue . . .
_
```

You archived P.DOC and P.BAK in earlier examples, so they're on the archive floppy disk. The Archive command doesn't identify the specific file that exists, but it does give you a chance to cancel if you don't want to overwrite a file.

Press Ctrl-Break to cancel the rest of the command, and respond to the MS-DOS prompt for confirmation with *y*:

```
Terminate batch job (Y/N)?y
```

If you had pressed any key other than Ctrl-Break, all the files named P would have been copied to the floppy disk in drive A, replacing any that were already there.

Erasing the Original of an Archived File

Archiving involves not only copying a file to an archive floppy disk but also means deleting the original. Your Archive command only copies; now it's time to make it delete too. But just as it's prudent to check whether the file exists on the archive floppy disk before you make the copy, it's also prudent to check again before deleting the original, just to be sure the file was copied to the archive floppy disk.

You need only one additional If command to check whether the file to be archived is on the archive floppy disk and, if it is, to delete the original from the working disk. Here is this modified version of ARCHIVE.BAT:

```
@echo off
echo Archive Procedure
echo Make sure archive floppy disk is in drive A
pause
if not exist a:%1 goto safe
echo a:%1 exists. Press CTRL-BREAK to cancel, or
pause
:safe
copy %1 a:%1
if exist a:%1 del %1
```

The screen responses of this version of your Archive command are the same as in the previous version, but the new command deletes the original file after copying it. Test it by archiving P.DOC again:

```
C>archive p.doc
Archive Procedure
Make sure archive floppy disk is in drive A
Press any key to continue . . .

_
```

Press any key:

```
a:p.doc exists. Press CTRL-BREAK to cancel, or
Press any key to continue . . .

_
```

You want to copy the file to the archive floppy disk, so press any key:

```
1 file(s) copied
```

See if the original version of P.DOC still exists by typing this:

```
C>dir p.doc

 Volume in drive C is HARD DISK
 Volume Serial Number is 1608-5A30
 Directory of C:\RUNDOS

File not found
```

The original is gone. Your Archive command does copy the specified file to the archive floppy disk and then deletes the original.

Functionally, the command is complete.

If you are using MS-DOS version 6.0 or later, the Move command can be used, eliminating the need to delete the original file after a successful copy. Users of MS-DOS versions 6.2 and later will find that the Move command will warn them if the file name already exists on the archive floppy disk, eliminating the need for the *if (not) exist* command.

PREVIOUS NEXT

Dressing Up Your Archive Command

The value of your batch files depends not only on what they do but also on how easy it is to use them correctly. This ease of use is particularly important if you use a batch file only occasionally or if someone else uses it; it's vital if the batch file deletes other files, as your Archive command does.

That's why, for example, your Archive command starts by turning echo off: An uncluttered screen helps to make the responses less confusing. You can also do some other things to make a batch file easy to use:

- Clear the screen.
- Use the Echo command to display messages that report on progress or results.
- Use spaces or insert tabs so you can position messages displayed by the Echo command where they are prominent on the screen.
- Use the Echo command to include line spaces that further improve the readability of the screen.

You can also use the Remark command to put notes to yourself in the batch file. As long as echo is off, these remarks aren't displayed. Remarks can help you remember how a batch file works, in case you have to change it or you want to create a similar command using the same technique. Remarks are especially useful if the batch file is long or if it's one you may not look at very often.

Echoing a Line Space

Up to now, you have used the Echo command either to turn echo off or to display a message. You can also use the Echo command to display a line space—something you'll probably want to do fairly often to improve the appearance of the display.

You've seen that you turn echo on or off by following *echo* with either *on* or *off*. If you follow *echo* with some other words, they are displayed as a message, and if you simply type *echo*, MS-DOS tells you whether echo is on or off. How do you tell MS-DOS to echo a line space? Beginning with version 3.1, typing *echo*, (*echo* and a period, with no space between) causes MS-DOS to display a line space. If this does not work with your version of MS-DOS, you can also type *echo*, press the Spacebar once, hold down the Alt key, and use the numeric keypad (not the numbers in the top row of the keyboard) to type *255*. When you release the Alt key, the cursor moves over one column. You don't see any characters displayed, but when MS-DOS carries out your Echo command, it will echo a line space on the screen.

The following example modifies ARCHIVE.BAT. If your version of MS-DOS doesn't respond to *echo.*, substitute Alt-255 in the appropriate lines. The changes in this batch file don't add any capability to your Archive command, but they do make it easier for someone to

understand what's happening. The changes are described in more detail in a moment. Here is the modified version of ARCHIVE.BAT:

```
@echo off
cls
Rem Three tabs in the following echo command
echo<tab><tab><tab>***ARCHIVE  PROCEDURE***
echo.
echo Make sure archive floppy disk is in drive A
pause
Rem Branch around warning if file not archived
if not exist a: %1 goto safe
echo.
echo a:%1 exists. Press CTRL-BREAK to cancel, or
pause
:safe
copy %1 a:%1
if exist a:%1 del %1
echo.
echo %1 archived
```

The purpose of each change is as follows:

- In line 2, the Clear Screen command starts your Archive command off with a blank screen, giving you complete control over what is displayed.
- In line 3, the Remark command reminds you that the space at the beginning of the Echo command in line 4 is created by pressing the Tab key three times.
- In line 4, the title (***ARCHIVE PROCEDURE***) is made more prominent and is displayed in the center of the screen by the insertion of three tabs.
- In line 5, an Echo command displays a line space below the title.
- In line 8, the Remark command explains the purpose of the Goto command included in line 9.
- In line 10, another Echo command displays a line space to make the warning message in line 11 more visible.
- In line 16, the Echo command displays another line space to make the message in line 17 more visible.
- In line 17, the Echo command's message tells which file has been archived.

Make the necessary changes, and compare your batch file with the modified form of ARCHIVE.BAT shown earlier. If there are any differences, correct them before saving the

revised version.

Although you haven't changed anything your Archive command does, its screen responses are quite different now. Test the new version by archiving P.OLD (which isn't on the archive floppy disk):

```
C>archive p.old
```

Because ARCHIVE.BAT now starts by clearing the screen, everything you see from this point on is displayed by your Archive command. When the command prompts you to check the floppy disk in drive A, press any key. The screen now looks like this:

```
***ARCHIVE PROCEDURE***

Make sure archive floppy disk is in drive A
Press any key to continue . . .

        1 file(s) copied

p.old archived
```

Test ARCHIVE.BAT again to see how it responds when the file already exists on the archive floppy disk. P.DOC and P.OLD are no longer on the disk in drive C (you archived them after making the change that erases the file), so archive P.BAK. Type the following, pressing any key to continue the command after each pause:

```
C>archive p.bak
```

The screen looks like this:

```
***ARCHIVE PROCEDURE***

Make sure archive floppy disk is in drive A
Press any key to continue . . .

a:p.bak exists. Press CTRL-BREAK to cancel, or
Press any key to continue . . .

        1 file(s) copied

p.bak  archived
```

Notice, though, that line 15 was added to ensure the copy was successful before the original was deleted, but lines 16 and 17 will be run even if the copy had not been successful. To solve this problem, again use some inverse logic and modify ARCHIVE.BAT, starting at line 15, to read this way:

```
@echo off
cls
```

```
Rem Three tabs in the following echo command
echo<tab><tab><tab>***ARCHIVE PROCEDURE***
echo.
echo Make Sure archive floppy disk is in drive A
pause
Rem Branch around warning if file not archived
if not exist a:%1 goto safe
echo.
echo a:%1 exists. Press CTRL-BREAK to cancel, or
pause
:safe
copy %1 a:%1
if not exist a:%1 goto nocopy
del %1
echo.
echo %1 archived
goto end
:nocopy
echo.
echo %1 NOT archived
echo Check floppy disk in drive A before preceding
:end
```

ARCHIVE.BAT is quite a bit longer than it was when you began, but your Archive command doesn't look much like a homemade command anymore. Although you needn't always go to such lengths when you create a command, it's nice to know that a little extra effort can make your work look professional. If others will use the batch files you create, the investment of your effort can quickly pay off in shorter training time, more efficient use of the system, and fewer mistakes.

PREVIOUS NEXT

Chapter Summary

This chapter covered a lot of ground. Experimenting is the best way to put what you learned here into practice. Just be certain to use floppy disks that don't contain files you need until you're sure your batch commands are working properly.

The next chapter describes two additional batch commands that let you create even more flexible batch files, and it shows you several useful batch files you can use to start your personal collection. You'll need the archive floppy disk in drive A. Remove this floppy disk, and label it ARCHIVED FILES.

PREVIOUS NEXT

Chapter 16: Creating More Smart Commands

The previous chapter showed you how to use the advanced capability of batch files. Knowing how to create batch files is only half the job, however; the other half is finding uses for them. This chapter shows you how to create some commands to search through the phone-list file you created in Chapter 13, "Taking Control of Your System"; it also describes four advanced batch commands and shows several useful batch files to give you some ideas for your own use.

Preparing for the Examples

For the first examples in this chapter, you need the phone-list file (PH.TXT) that you created in Chapter 13. If you haven't gone through the examples in Chapters 13, 14, and 15, you should do so before proceeding. PH.TXT should already be in the directory named \RUNDOS. Change the current directory to \RUNDOS by typing:

```
C>cd \rundos
```

If your prompt doesn't appear as C>, you can change it to match the following examples by typing *prompt*. This completes the preparation for the examples.

PREVIOUS NEXT

Commands for Searching Through a File

In Chapter 13, you created a file of names, addresses, and telephone numbers, and you used the Find and Sort commands to display entries. You can also put the Find and Sort commands in batch files to create your own search commands and achieve some of the capabilities of a simple record-management program.

For example, the simplest search uses the Find command to display all records that contain a particular string. Create a batch file named SHOW.BAT by typing the following:

```
C>copy con show.bat
@echo off
find "%1" ph.txt
^Z
        1 file(s) copied
```

This batch file gives you a Show command that displays all records from PH.TXT that contain the string you specify as the parameter to your Show command. To search the phone list, type *show* followed by the string. For example, to display the entries for all consultants (entries that contain the string *cons*), type this:

```
C>show cons
```

MS-DOS displays all entries for consultants:

```
-----       PH.TXT
Jones     Michele   (747) 429-6360  cons chemist
Black     John      (747) 426-3385  cons mech eng pkg
Jones     Alison    (747) 429-5584  cons Chem engineer
```

Note As it stands, your Show command requires that you specify the string in the same combination of uppercase and lowercase letters that appears in the file. If you have version 5 or later of MS-DOS, remember that you can include the /I parameter to tell the Find command to ignore differences in case. For example, if you entered the Find command as *find /i "%1" ph.txt*, you could type the Show command in the preceding example as *show cons, show Cons,* or *show CONS* and still be assured of finding the entries you wanted. For simplicity, the following examples omit the /I parameter of the Find command. If you want, however, you can include it to make your batch files even more flexible.

As you've seen, typing *show cons* is easier than typing *find "cons" ph.txt.* Read on, and you'll find that you can use similar batch files to conduct even more powerful searches just as easily.

Compound Searches

As the examples in Chapter 13, "Taking Control of Your System," showed, you can also

combine Find commands to search for records that contain various combinations of character strings—for example, all consultants named Jones or all entries that are outside the 747 area code. Putting these Find commands in a batch file saves even more typing than the Show command you just created.

For example, suppose you want to create a command to show all entries that contain both one string and another. This requires two Find commands, with the output of the first piped to the second. You could call such a command Showand. Create a file named SHOWAND.BAT by typing this:

```
C>copy con showand.bat
@echo off
find "%1" ph.txt | find "%2"
^Z
       1 file(s) copied
```

This batch command takes two parameters—the two strings the Find command searches for. Now you can search the phone list for entries that contain two strings as easily as you can search it for entries containing one string; just type *showand* followed by the two strings. For example, to display all consultants named Jones, type the command at the top of the next page.

```
C>showand cons Jones
```

MS-DOS displays the records that contain both strings:

```
Jones     Michele   (747) 429-6360  cons chemist
Jones     Alison    (747) 429-5584  cons Chem engineer
```

This is definitely easier than typing *find "cons" pb.txt | find "Jones".*

What if you want to create a command that shows all entries except those that contain a particular string? Use a Find command with the /V parameter. You could call this command Showxcpt; create SHOWXCPT.BAT by typing the following:

```
C>copy con showxcpt.bat
@echo off
find /v "%1" ph.txt
^Z
      1 file(s) copied
```

This batch command requires one parameter, the string you don't want to see. For example, to display all entries not in the 747 area code, type the following:

```
C>showxcpt (7
```

MS-DOS displays all lines from PH.TXT that don't contain *(7.*

```
-----      PH.TXT
Green      Fred      (541) 926-4921  cust math teach
Smith      Ed        (541) 835-8747  vend caterer
```

These three batch files—SHOW, SHOWAND, and SHOWXCPT—let you search a file quickly in several ways. You can combine all three searches into a single command just by changing SHOWBAT.

Chaining Batch Files to Create Powerful Commands

As your skill in creating batch files grows, you'll sometimes find that you want to use one batch file to carry out the commands in another batch file. One way to do this is to *chain* the batch files. Another, described later in this chapter, is to use the Call command.

When you chain batch files, you use the name of one batch file as a command in another. MS-DOS then carries out the commands in the second batch file as if you had typed its name; if the second batch file contains the name of a third batch file, MS-DOS carries out its commands, and so on. When the commands in the last chained batch file have been carried out, MS-DOS returns to the system prompt.

To see how this works, modify SHOW.BAT to cover all three types of searches you just performed by chaining it to either SHOWAND.BAT or SHOWXCPT.BAT. When you do this, the parameters you type with the revised Show command must specify the type of search as well as the string or strings to search for.

You're creating your own Show command with the following parameters:

show xcpt and <string1> <string2>

xcpt searches for entries that don't contain the specified string.

and searches for entries that contain two specified strings.

<string1> and <string2> are the strings to search for. If you include *and*, you must include both <string1> and <string2>; otherwise just include <string1>.

If you don't specify either *xcpt* or *and*, your Show command searches for all entries that contain <string1>.

This more powerful version of SHOW.BAT is still fairly short. Type the following:

```
C>copy con show.bat
@echo off
if %1==xcpt showxcpt %2
if %1==and showand %2 %3
find "%1" ph.txt
^Z
       1 file(s) copied
```

The first If command checks whether the first parameter (*%1*) typed with the Show command is *xcpt* (recall that the == compares two strings to see if they are identical). If *%1* is the same as *xcpt*, SHOWXCPT.BAT is carried out. The second parameter *(%2)* you type

with the Show command is the string to search for; it is the single parameter that SHOWXCPT.BAT requires.

The second If command checks whether the first parameter typed with the Show command is *and*. If it is, SHOWAND.BAT is carried out. The second and third parameters (*%2* and *%3*) typed with the Show command are the two strings to search for; they are the two parameters that SHOWAND.BAT requires.

If the first parameter (*%1*) typed with the Show command is neither *xcpt* nor *and*, the Find command is carried out to perform a simple search.

Except for the Echo command, only one of the commands in SHOW.BAT is carried out in any particular search. Figure 16-1 shows the contents of SHOW.BAT and an example of each type of search you can make. For each example, the figure lists the Show command as you would type it and substitutes values for the replaceable parameters in SHOW.BAT. The command in SHOW.BAT that is carried out is not shaded. An arrow to the right represents chaining to SHOWXCPT or SHOWAND; the contents of each chained batch file, with values substituted for its replaceable parameters, are shown below the chained batch file.

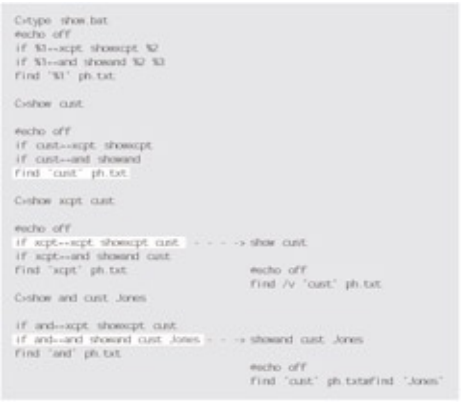


Figure 16-1: Chaining batch files.

Now you can perform any of the three types of searches with the Show command. The simple search works as it did in the earlier Show command. For example, to display all entries that contain *Jones*, type this:

```
C>show Jones

-----      PH.TXT
Jones      Michele   (747) 429-6360  cons chemist
Jones      Alison    (747) 429-5584  cons Chem engineer
Jones      James     (747) 636-3541  cust architect
```

To display all entries that don't contain a particular string, type *xcpt* as the first parameter. For example, to display all entries outside the 747 area code, type the command you see at the top of the next page.

```
C>show xcpt (7
```

```
-----     PH.TXT
Green     Fred       (541) 926-4921 cust math teach
Smith     Ed         (541) 835-8747 vend caterer
```

Or, to search for two strings, type *and* as the first parameter. For example, to see all engineers named Jones, type this:

```
C>show and Jones eng
Jones     Alison     (747) 429-5584  cons Chem engineer
```

If you put your telephone numbers and business cards in a text file like this, these three batch files put the contents of the file at your fingertips. Not only can you search for an entry quickly, you can easily display groups of related entries.

This application, like Edit, is another example of how MS-DOS can make your computer more useful without additional software.

PREVIOUS NEXT

Some Useful Batch Files

To give you some ideas about the sort of batch commands that might help you from day to day, several useful batch files are shown here. These are not step-by-step examples; each description gives the purpose of the batch file, shows its contents, explains how it works, and describes how you would use it.

An effective way to become familiar with batch commands is to experiment. You could enter the following batch files, for example, then experiment with them, making changes and seeing the effects of the changes. If your batch files include commands that can modify or delete files, be sure to test the batch files on disks that contain files you don't need.

You can create the batch files by copying from the console to a file or by using a text editor. Most of the batch files include one or more Echo commands to add a line space to the display (for readability); they are shown as *echo* followed by a period (*echo*.). If you don't have version 3.1 or later, echo a line space by typing *echo*, holding down the Alt key, and typing *255* using the numeric keypad.

Viewing a Long Directory Listing

Even though you back up old files and clear out unneeded ones regularly, a directory can soon contain more files than MS-DOS can list on one screen. And some often-used directories—word processing directories, for example, or collections of spreadsheets and charts—can become quite large.

If you have version 4 or later of MS-DOS and an EGA or VGA display, you can use the LONGDIR.BAT file described here to change the display to 43 lines per screen, display a long directory one screenful at a time, and then change the display back to 25 lines.

LONGDIR.BAT contains this:

```
@echo off
cls
mode con lines=43
dir %1 /p
pause
mode con lines=25
```

The first Mode command switches the display to 43 lines. The Directory command then finds the listings for the directory (*%1*) you specify as a parameter with the Longdir command, displays the entries 43 lines at a time, and waits for you to press a key. The Pause command displays *Press any key to continue* when the directory listing is complete. The second Mode command returns the screen display to 25 lines.

To use the command, you type *longdir* followed by the directory listing you want to see. You can include a drive letter, followed by a colon, a path name, and a file name including wildcard characters. For example, you can type the command as *longdir a:, longdir*

\clients\wp or *longdir \clients\wp*.doc.* If you don't include a drive, path, or file name (type just *longdir*), the command displays the current directory.

Your system could be set up so that LONGDIR.BAT lists the directory you want but doesn't change to 43 lines per screen after telling you that *ANSI.SYS must be installed to perform requested function*. If this happens, you need to identify a file named ANSI.SYS to MS-DOS. Chapter 17, "Tailoring Your System," tells how to do this.

Cleaning Up Disk Storage

Because many word processors and other application programs create a backup file with the extension BAK each time you change a file, your disks can get crowded with backup files you don't need. The CLEANUP.BAT batch file described here replaces one you created earlier; it displays the directory entries of all files with an extension of BAK and then pauses. At that point, you can cancel the command by pressing Ctrl-Break; if you press any key to proceed, the command erases the files.

CLEANUP.BAT contains this:

```
@echo off
cls
echo *** Will erase the following files: ***
dir %1*.bak
echo.
echo Press CTRL-BREAK to cancel, or
pause
del %1*.bak
```

Notice that there is no space between *%1* and **.bak* in the Directory and Delete commands; this lets you specify a drive letter or a path name as a parameter with the file name. Be sure to end a path name with a backslash (\).

To erase all BAK files from the current directory on the disk in the current drive, type *cleanup* and press any key when the Pause command prompts you.

To erase all BAK files from another directory, type the path name followed by \ as a parameter; you can also precede the path name with a drive letter and colon (for example, *cleanup a:*\ or *cleanup \mkt\wp*\).

Directory of Subdirectories

If a directory contains many subdirectories and files, sometimes you'd like to see just the subdirectories. The DIRSUB batch file described here lets you do just that; it uses different techniques depending on the version of MS-DOS you're using.

If you're using version 5 or later, DIRSUB.BAT contains this:

```
@echo off
```

```
echo *** Subdirectories in %1 ***
echo.
dir %1 /ad
```

The /AD (Attribute Directory) parameter tells MS-DOS to display just the subdirectories.

You can type one parameter (*%1*) with the Dirsub command to specify the directory whose entries are to be displayed. The parameter can include a drive letter and colon, a path name, or both. If you don't include a parameter, this Dirsub command displays the subdirectories in the current directory (but doesn't display the directory name because you didn't type any characters to replace %1 in the message line).

If you're using an earlier version of MS-DOS, you can take advantage of the fact that the Directory command displays <*DIR*> to identify a subdirectory. This version of DIRSUB.BAT contains the following:

```
@echo off
echo *** Subdirectories in %1 ***
echo.
dir %1 œ find "<"
```

In this file, the output of the Directory command is piped to a Find command that displays all directory lines that contain a less-than sign (<).

With either version of DIRSUB.BAT, to see the subdirectories in the current directory you would type *dirsub*. To see those in the current directory of drive A, you would type *dirsub a:*, and to see the subdirectories in \MKT\WP, you would type *dirsub \mkt\wp*.

The two special entries . and .., which the Directory command identifies as subdirectories, are in all directories except the root. You can modify DIRSUB.BAT so that it doesn't display these entries. To do this, pipe the output of the Directory command to a Find command that uses the /V parameter to eliminate all lines that contain a period. For version 5 or later, the command would be this:

```
dir %1 /ad œ find /v "."
```

For an earlier version of MS-DOS, the command would be this:

```
dir %1 œ find "<" œ find /v "."
```

The added Find command uses the /V parameter to eliminate all lines that contain a period. You would use this version of the Dirsub command just as you would use the earlier version.

If you created DIRSUB.BAT, it's a snap to reverse the action so that it displays only files instead of subdirectories. The DIRFIL batch file requires that you make only a simple change to the last line of DIRSUB.BAT.

If you're using version 5 or later, simply put a hyphen before the D in the /AD parameter to

reverse its meaning:

```
dir %1 /a-d
```

If you're using an earlier version, add the /V parameter to the Find command to reverse its meaning:

```
dir %1 œ find /v   "<"
```

You would use this command just as you used Dirsub.

Moving Files from One Directory to Another

In a tree-structured file system, you'll sometimes want to move files from one directory to another. Versions 6.0 and later of MS-DOS include the Move command, which makes it simple to move files. If you have an earlier version, you can move files only by copying the file to the new directory with the Copy command and then erasing the original with the Delete command. MOVE.BAT combines these in one command. If you are running MS-DOS 6.0 or later, and want to experiment with this file, save the file as MOVER.BAT. Given the Move command, MS-DOS version 6.0 or later will find MOVE.EXE before even looking for MOVE.BAT.

MOVE.BAT contains this:

```
@echo off
copy %1 %2
cls
echo Files in target directory:
echo.
dir %2 /p
echo.
echo If files to be moved are not in directory,
echo press CTRL-BREAK to cancel. Otherwise
pause
del %1
```

Here is how the batch file works, line by line:

- In line 1, the Echo command turns echo off.
- In line 2, the Copy command copies the files to the target directory.
- In line 3, the Clear Screen command clears the screen.
- In lines 4 and 5, Echo commands display a message and a line space.
- In line 6, the Directory command displays the contents of the target directory. The /P parameter is included to ensure time enough to view even a very long directory.

- In lines 7 through 9, Echo commands display a line space and a warning message.
- In line 10, the Pause command makes the system pause to let you cancel the Delete command in line 11 if the files you moved are not displayed in the listing of the target directory.

You have created a command with two parameters:

move <source> <target>

<source> is the name of the file to be moved (copied and then deleted). You can include a drive letter and path name. If you use wildcard characters in the file name, MS-DOS displays the name of each file it copies.

<target> is the name of the directory to which the <source> files are to be copied. If you omit <target>, the files are copied to the current directory.

For example, assume that the current directory is \MKT\WP. To move the file REPORT.DOC to the directory \ENG\WP, you would type *move report.doc \eng\wp*. To move the file BUDGET.JAN from \ENG\WP to \MKT\WP, you type the command as *move \eng\wp\budget.jan*. To move all files from \MKT\WP to \WORD\MKT, you would type *move *.* \word\mkt.*

PREVIOUS NEXT

Four Advanced Batch Commands

Note The batch commands you have worked with up to this point are sufficient for you to create useful batch files like the search commands for the phone-list file. Rather than continue with the rest of this chapter right now, you might want to put the book aside for awhile and experiment, and then return to learn the last four batch commands this book discusses.

The four remaining batch commands—Shift, For, Call, and Choice—give you even more control over how your batch files work. They're somewhat more complicated than the other batch commands, but the examples should make their use clear.

Preparing for the Advanced Examples

The remaining examples in the chapter require the P.* and Q.* files that should still be in your RUNDOS directory from the exercises you did in Chapter 15, "Creating Smart Commands." You also need to put the floppy disk you labeled ARCHIVED FILES in drive A.

The Shift Command: Shifting the List of Parameters

The Shift command moves the list of parameters you type with a batch command one position to the left. For example, suppose you type a Shift command with three parameters. After the Shift command is carried out once, what was *%3* becomes *%2* and what was *%2* becomes *%1*; what was *%1* is gone. After a second Shift command, what started out as *%3* is *%1*; what started as *%2* and *%1* are both gone.

This command lets a short batch file handle any number of parameters; the following sample batch file illustrates the technique. ARCH1.BAT archives any number of files. Here are its contents:

```
@echo off
:start
if "%1"==" " goto done
echo *** Archiving %1 ***
if not exist a:%1 copy %1 a:
shift
goto start
:done
```

Here's a description of how it works:

- As usual, the first line is an Echo command to turn echo off.
- The label *:start* marks the beginning of the commands that will be repeated for each parameter.
- The If command checks to see whether a parameter was entered for *%1*. It does

this by comparing "*%1*" with two quotation marks enclosing nothing at all (""). You're telling MS-DOS to compare the first parameter (if one was typed) with nothing, or no parameter. When the comparison is true, meaning that there are no more parameters, the Goto command sends MS-DOS to the label *:done* in line 8.

- The Echo command displays the name of the file that is being archived.
- The If command checks whether the file to be archived exists on drive A; if it doesn't, the Copy command copies the file from the current drive to drive A.
- The Shift command moves the list of parameters one position left.
- The Goto command sends MS-DOS back to the label *:start*.
- The label *:done* marks the end of the command file.

To use this batch file you type *arch1* followed by the names of the files to be archived. The command stops when "*%1*" equals " "—in other words, when there are no more file names to be substituted for *%1*.

To test the command, copy the file named P.OLD (which you archived in Chapter 15, "Creating Smart Commands") from the floppy disk in drive A to the current drive by typing this:

```
C>copy a:p.old
```

Now archive P.OLD, Q.DOC, and Q.OLD by typing the command shown here:

```
C>arch1 p.old q.doc q.old
*** Archiving p.old ***
*** Archiving q.doc ***
      1 file(s) copied
*** Archiving q.old ***
      1 file(s) copied
```

The confirming messages show that MS-DOS did not copy the file that was already on the floppy disk in drive A (P.OLD) but that it did copy the files that weren't on the floppy disk (both Q.DOC and Q.OLD).

This short archive command doesn't display instructions or warnings like the batch file you created in Chapter 15, "Creating Smart Commands," but it does show how you can use the Shift command to write a batch file that handles any number of parameters. The technique is simple: Do something with *%1*, shift the parameters, then use a Goto command to send MS-DOS back to the beginning to do it all over again. Just make sure you define a way to stop the process, or MS-DOS will carry out the command forever.

The For Command: Carrying Out a Command More than Once

Sometimes you might want MS-DOS to carry out a command more than once in a batch file

—for instance, to carry out a command for each file that matches a file name with wildcard characters. The For command does just that. Like the If command, the For command has two parts: The first part defines how often a command is to be carried out, and the second part is the command to be carried out.

The For command is somewhat more complex than the other batch commands. If you don't fully understand the description of its parameters, read on and try the examples. As with many other aspects of using a computer, it's easier to use the For command than it is to read about it.

The For command has three parameters:

for %%p in (<set>) do <command>

The words *in* and *do* are required in the command; they are not parameters.

%%p is a replaceable parameter used inside the For command. MS-DOS assigns to it, in turn, each value included in (<set>).

(<set>) is the list of possible values that can be given to *%%p*. You separate the values with spaces, and you must enclose the entire list in parentheses—for example, *(1 2 3).* You can also specify a set of values with a file name that includes wildcard characters—for example, *(a:*.doc).*

<command> is any MS-DOS command other than another For command. You can use both batch-command parameters (such as *%1*) and the For command replaceable parameter *(%%p)* in <command>.

It's all less complicated than it sounds. When MS-DOS carries out a For command, it assigns to *%%p,* in turn, each value that you specify in (<set>), then it carries out <command>. Each time it carries out <command>, MS-DOS first substitutes for *%%p* the current value taken from (<set>).

A brief example shows how the For command works. The following batch command carries out an Echo command three times, displaying each of the three words enclosed in parentheses. Create FOR1.BAT by typing this:

```
C>copy con for1.bat
for %%p in (able baker charlie) do echo %%p
^Z
      1 file(s) copied
```

In this particular For command, *(able baker charlie)* is (<set>) and *echo %%p* is <command>. The For command tells MS-DOS to carry out the Echo command once for each word in parentheses, substituting, in turn, *able, baker,* and *charlie* for *%%p.* Test it by typing this:

```
C>for1
```

For a change, this batch file doesn't begin by turning echo off, so MS-DOS displays each command it carries out, starting with the For command:

```
C>for %p in (able baker charlie) do echo %p
```

MS-DOS then displays and carries out the Echo command once for each value in the set, each time substituting the next value from the set for the replaceable parameter in the Echo command:

```
C>echo able
able

C>echo baker
baker

C>echo charlie
charlie
```

Instead of specifying actual values in the set, you can use replaceable parameters, such as *%1*. This technique works as it does in any other batch command but can be confusing in this case because now you can have *%1* and *%%p* in the same command. Here's how it works: *%1* refers to the first parameter typed with the batch command, *%2* refers to the second parameter typed with the batch command, and so forth; *%%p* refers to the value MS-DOS selects from the set enclosed in parentheses within the For command.

The next example shows you the difference. Here, the words to be displayed are typed as parameters of the batch command, instead of being included as part of the For command. Create FOR2.BAT by typing this:

```
C>copy con for2.bat
for %%p in (%1 %2 %3) do echo %%p
^Z
      1 file(s) copied
```

This For command tells MS-DOS to carry out the Echo command (*echo %%p*) once for each value in parentheses, substituting for *%%p* the first, then the second, and finally the third parameter you type with the For2 command. Test it by typing this:

```
C>for2 dog easy fox
```

MS-DOS displays each command it carries out. First it displays the For command:

```
C>for %p in (dog easy fox) do echo %p
```

Note that MS-DOS has substituted values for all replaceable parameters. The set in parentheses is now *(dog easy fox)* because those are the three parameters you typed; they have replaced (*%1 %2 %3*). As you may have noticed in the preceding example, MS-DOS has also dropped one of the percent signs from *%%p.* MS-DOS removes the first percent sign when it substitutes the parameters you type with the command for *%1*, *%2,* and so forth. It does, however, leave one percent sign to show that a value must still be substituted

for *%p.*

MS-DOS then displays and carries out three Echo commands, each time substituting one value from the set in parentheses for *%p:*

```
C>echo dog
dog

C>echo easy
easy

C>echo fox
fox
```

If you wish, you can specify the set in a For command with a file name and wildcard characters. MS-DOS then assigns to *%%p,* in turn, each file name that matches the wildcard characters. You can use this technique to create a simple Archive batch file using only a For command. The following command doesn't display instructions and warnings, but it does show you how much can be done with a single batch command.

Create a batch file named ARCH2.BAT by typing the following:

```
C>copy con arch2.bat
for %%p in (%1) do if not exist a: %%p copy %%p a:
^Z
      1 file(s) copied
```

Q.DOC and Q.OLD are on the floppy disk in drive A (you copied them in the example of the Shift command). Use them to test ARCH2.BAT by archiving all Q files:

```
C>arch2 q. *
```

MS-DOS displays each command it carries out, starting with the For command (remember, you have not turned echo off):

```
C>for %p in (q.*) do if not exist a:%p copy %p a:
```

MS-DOS then displays and carries out three If commands, each time substituting a file name that matches the set enclosed in parentheses for the replaceable parameter in the If command:

```
C>if not exist a: Q.DOC copy Q.DOC a:

C>if not exist a:Q.BAK copy Q.BAK a:
      1 file(s) copied

C>if not exist a: Q.OLD copy Q.OLD a:
```

The messages displayed by the Copy command confirm that MS-DOS did not copy either

Q.DOC or Q.OLD because they were already on the archive floppy disk, but did copy Q.BAK, which hadn't been archived.

The For command gives you a quick way to carry out an MS-DOS command several times. As shown in some of the sample batch files that conclude this chapter, the For command makes it possible to create powerful batch commands.

The Call Command: Using a Batch Command in a Batch File

Earlier in this chapter, in the final version of SHOW.BAT, you used batch commands (Showand and Showxcpt) in a batch file. MS-DOS didn't return to SHOW.BAT after carrying out these batch commands even though neither command was actually the last command in SHOW.BAT; the last command was Find. You can use this capability, called *chaining,* with any version of MS-DOS. Once you chain to another batch command, you can't go back to the first.

If you're using version 3.3 or later, however, you can also use the Call command in a batch file to tell MS-DOS to carry out the commands in another batch file. Unlike chaining, the Call command causes MS-DOS to come back and continue with the next command in the original batch file. This lets you use a batch command of your own making in a batch file, just as you would use an MS-DOS command.

The Call command has two parameters:

call <batchfile> <parameters>

<batchfile> is the name of a batch command that you want MS-DOS to carry out from within the calling batch file.

<parameters> represents any parameters that <batchfile> requires.

Two short batch files demonstrate the usefulness of the Call command. First, create a batch file named ECHOIT.BAT by typing the following:

```
C>copy con echoit.bat
@echo off
echo %1
^Z
      1 file(s) copied
```

ECHOIT.BAT simply echoes the parameter you enter with it. For example, to echo *fred,* type the following:

```
C>echoit fred
fred
C>_
```

Now create another batch file named ECHOALL.BAT that uses a For command to carry out ECHOIT.BAT for each of up to three parameters, pauses until you press a key, and displays

an ending message:

```
C>copy con echoall.bat
@echo off
for %%p in (%1 %2 %3) do call echoit %%p
pause
echo End of ECHOALL.BAT
^Z
        1 file(s) copied
```

The For command carries out the Call command for each of up to three parameters that can be entered with the Echoall batch command. The Call command carries out the batch command ECHOIT.BAT, specifying, in turn, each parameter entered with the Echoall batch command.

Now type the following to echo the words *xray, yankee,* and *zulu*:

```
C>echoall xray yankee zulu
xray
yankee
zuIu
Press any key to continue . . .
_
```

Press a key, and MS-DOS displays the final message in ECHOALL.BAT:

```
End of ECHOALL.BAT
```

If you hadn't included the Call command, MS-DOS would have carried out ECHOIT.BAT once to display the first parameter (*xray*) and then would have returned to the system prompt. Because you did use the Call command, MS-DOS carried out ECHOIT.BAT three times, once for each parameter, and then returned to the final lines of ECHOALL.BAT before displaying the system prompt.

The Choice Command: Writing an Interactive Batch Command

Most of the programs you use are interactive. They occasionally ask you for some information (such as the name of a file or whether you want to exit the program), wait for you to respond, then take some action based on your answer. Starting with version 6.0 of MS-DOS, you can give your batch commands the same capability with the Choice command.

The Choice command lets you list the one-character responses that are acceptable and, if you like, the text of a message to be displayed. When it's carried out, the Choice command displays the message (if you specify one) and a prompt that lists the characters that can be chosen, then waits for a keyboard response.

So that you can determine which response was entered, Choice sets a variable called

errorlevel to 1 if the first response you listed was chosen, 2 if the second response was chosen, and so forth. You use the *errorlevel* option of the If command to check what the response was and take the appropriate action in the batch file.

The Choice command has three commonly used parameters. (See Appendix C, "MS-DOS Command Reference," for an explanation of all the parameters.)

choice /C:<keys> /N <text>

/C:<keys> specifies which characters (letters, numbers, or punctuation marks) are permitted as responses. Don't separate the characters with spaces or commas.

/N tells Choice not to display the prompt.

<text> specifies a message to be displayed before the prompt.

You can use the Choice command to make a simple menu system. Suppose that you use Procomm Plus, Microsoft Money, and Microsoft Works and want to display a menu that lets you start any of these three programs just by pressing a letter. The single batch file MENU.BAT would let you do just that:

```
@echo off
cls
echo CHOOSE A PROGRAM
echo.
echo Procomm Plus
echo Money
echo Works
echo.

choice /c:pmwq /n Press P, M, W, or Q to quit:

if errorlevel 4 goto end
if errorlevel 3 goto works
if errorlevel 2 goto money
if errorlevel 1 goto pcplus

:works
\msworks\msworks
menu

:money
\msmoney\msmoney
menu

:pcplus
\pcplus\pcplus
```

```
menu

:end
```

The first section of the batch file clears the screen and displays the menu of your three programs. The Choice command then waits for a reply and sets *errorlevel* to the proper value. Here's a closer look at the three parameters of this Choice command:

- */c:pmwq* specifies the four permitted responses: P (for Procomm Plus), M (for Money), W (for Works), and Q (for Quit).
- */n* tells the Choice command not to display the prompt (which, in this case, would be *[P,M, W,Q]?*).
- *Press P, M, W, or Q to quit* is the message that the Choice command displays before waiting for a reply.

Next the batch file checks the value of *errorlevel* set by the Choice command and runs the appropriate program. Because the If command actually checks whether *errorlevel* is equal to or greater than the specified value, you check for the largest possible value first, then the next largest, and so forth. The first reply listed in the Choice command's /C parameter is P, so Choice assigns *errorlevel* the value of 1 if someone presses P; the fourth reply listed is Q, so Choice assigns *errorlevel* the value of 4 if Q is pressed.

After you're done running the program you chose, the batch file runs Menu, a batch command that carries out this batch file. How can a batch file carry out itself? When MS-DOS finds a batch command in a batch file, it stops carrying out the current batch file and starts the one named. In this case, MS-DOS stops MENU.BAT and starts it again at the beginning; it doesn't matter that the same batch file is being started again. Because this batch command appears before the end of the batch file, it causes the menu to be displayed again after you exit one of the program choices. The only way to leave the menu is to press Q (or Ctrl-Break, which stops any batch file).

When you type *menu*, MS-DOS clears the screen and displays your menu:

```
CHOOSE A PROGRAM

Procomm Plus
Money
Works

Press P, M, W, or Q to  quit:_
```

If you press any key other than one of the four listed, MS-DOS beeps and waits for you to press another key. If you press W, MS-DOS carries out the command that starts Microsoft Works because the Choice command set *errorlevel* to 3. When you exit Microsoft Works, MS-DOS displays your menu again. To leave the menu, press the Q key.

You can adapt this batch file to your own programs by replacing the program names in the first part of the file (using more lines if necessary) and changing the commands that start the programs (again, add more commands if you add items to the menu).

Some More Useful Batch Files

The following batch files use the advanced batch commands. Again, they aren't step-by-step examples; they're working samples to give you an idea of what can be done with the full set of batch commands. You can enter the batch files by copying from the console or by using a text editor.

Displaying a Series of Small Text Files

If you work with many small text files, such as batch files or boilerplate paragraphs for word processor documents, it's handy to be able to review several files with one command, rather than having to print or display them one at a time. The batch file shown here, REVIEW.BAT, uses the Shift, Type, and Pause commands to display any number of files one at a time. Remember, *echo.* (or *echo* Alt-255) echoes a line space. REVIEW.BAT contains this:

```
@echo off
:start
if "%1"=="" goto done
cls
echo<space><tab><tab>*** FILENAME: %1 ***
echo.
type %1
echo.
echo.
pause
shift
goto start
:done
```

This batch file uses the same technique as the earlier examples of the Shift command. The first Echo command displays the file name (moved toward the center of the screen with two tabs) so that you'll know what file is displayed. The last two Echo commands display two blank lines to separate the file from the Pause command message.

With this Review command, you type the names of the files as parameters. To display the P.* files, for example, you type *review p.doc p.old p.bak.* MS-DOS clears the screen and displays the first file, then displays the Pause command message and waits for you to press any key before clearing the screen and displaying the next file. When the If command finds a null parameter (" "), MS-DOS returns to command level.

Searching Files for a String of Characters

Have you ever searched through your paper files to find a specific letter or reference? To help you find all lines that contain a particular string, SCAN.BAT lets you specify a file to be searched and a string to search for. It displays each line in the file that contains the string. You can also use wildcard characters to search a set of files. Here are the contents of

SCAN.BAT:

```
@echo off
cls
echo<space><tab><tab>***LINES IN %1 THAT CONTAIN %2***
echo.
for %%p in (%1) do find "%2" %%p
```

All the work here is done in the last line of the batch file. The earlier lines clear the screen and display a title. The For command actually carries out the Find command for each file name that matches the first parameter you type with the Scan command; the Find command is the one that searches for the string that you type as the second parameter.

As mentioned earlier in the chapter, remember that if you have version 5 or later of MS-DOS, you can use the /I parameter to tell the Find command to ignore differences between uppercase and lowercase characters (*for %%p in (%1) do find /i "%2" %%p*).

To display each line that contained the word *sales* in all files with an extension of DOC, you would type *scan *.doc sales.* MS-DOS would clear the screen and display

```
***LINES IN *.doc THAT CONTAINS sales***
```

followed by each line that contained the word *sales.*

Displaying a Sorted Directory

Beginning with version 5, you can use the /O parameter of the Directory command to sort a directory by name, extension, size, or date and time. If you don't have version 5, however, you can sort a directory by name, extension, or size by creating the four small batch files described here.

Because the items in a directory entry always begin in the same column (unless you specify a wide directory listing), you can sort the directory entries by extension or size by using the column parameter (/+<number>) of the Sort command. (Sorting by name doesn't require the column parameter because the file name starts in column 1.) These are the columns where the information begins:

- Name—column 1 (doesn't need to be specified)
- Extension—column 10
- Size—column 16

Although directory entries include the date and time, sorting them by date (column 24) or time (column 34) doesn't work properly. Sorting by date, for example, would cause January of one year to appear in the list before December of the preceding year (that is, 1-1-95 would *precede* 12-31-94). If you sorted by time, a file created at 9 in the morning on January 5, 1995, would appear in the list *before* a file created at 11 in the morning a month

earlier. Then, too, if you're using version 4 of MS-DOS, which displays time based on a 12-hour clock rather than a 24-hour clock, a file created at 5 in the evening would be listed before a file created at 9 that morning.

Sorting by date or time, therefore, is not particularly useful. On the other hand, the benefits of sorting a directory alphabetically by file name or by extension are obvious. Furthermore, if you need to delete some files to make space on a disk, or if you're just interested in the relative sizes of files, sorting by size is useful.

To create a command that displays a directory sorted by name, extension, or size, you create three batch files that sort and display the directory, and you create a fourth batch file that chains to the correct one of the other three.

First, copy from the console, as you have done before, to create the following one-line batch files; their names tell how they sort the directory:

DIRNAME.BAT:

```
dir %1 œ sort œ find "-" œ more
```

DIREXT.BAT:

```
dir %1 œ sort /+10 œ find "-" œ more
```

DIRSIZE.BAT:

```
dir %1 œ sort /+16 œ find "-" œ more
```

Note If you have MS-DOS version 6.2 or later, add */-p /-w* after the *%1* in these batch files.

Each of these batch files sorts and displays the contents of the directory specified in the parameter *%1* and pipes the output of the Sort command to a Find command that selects only the lines that contain a hyphen, limiting the output to just those lines that contain an entry. (The hyphens separate the parts of the date.) The output of the Find command is then piped to the More command to handle directories more than one screen long. If no parameter is specified, the batch command sorts and displays the current directory. The only difference among the batch files is the column in which sorting begins.

You now need the batch file that chains to the correct one of the previous three. Name the file DIRSORT.BAT; it contains:

```
@echo off
for %%p in (name ext size) do if "%1"=="%%p" dir%%p %2
echo First parameter must be name, ext, or size
```

Be sure *not* to put a blank between *dir* and *%%p* in the second line, or the result won't be a valid file name for any of the chained batch files. The quotation marks around *%1* and *%%p* prevent MS-DOS from displaying an error message if no parameter is typed with the

command.

In the DIRSORT batch file, the For command sets *%%p,* in turn, to each of the words in the set in parentheses, then carries out the If command, which compares *%%p* to the first parameter typed with the Dirsort command. If there is a match, the If command adds the value of *%%p* to *dir* to produce the name of one of the three sort batch files and chains to that file (for example, *dir* + *name* would become *dirname*). The second parameter (*%2*), if any, typed with the Dirsort command is the *%1* parameter (the directory name) needed by the chained batch file.

If the first parameter you type with the Dirsort command isn't one of the three words in the set in parentheses, the condition in the If command is not true. Then the Echo command in the next line is carried out, displaying a message that lists the correct parameters, and the Dirsort batch command ends without chaining to one of the files that display a sorted directory.

To display a sorted directory, you would type the Dirsort command with one or two parameters: The first, which is required, would specify how to sort and would have to be *name, ext,* or *size;* the second parameter, which is optional, would be the drive letter, path name, or name of the directory to be displayed. If you didn't include the second parameter, the command would display the current directory of the disk in the current drive.

For example, to display the current directory of the disk in the current drive, sorted by size, you would type *dirsort size.* To display the root directory, sorted by name, of the disk in drive A, you would type *dirsort name a:\.*

PREVIOUS NEXT

Chapter Summary

These batch files should give you a good start on a collection of special-purpose commands, and they might give you some ideas for creating more of your own. This is where the flexibility of MS-DOS really becomes apparent: The wide range of MS-DOS commands and capabilities is only the starting point for putting this operating system to work. With redirection, filter commands, batch files, and macros, you can combine all the other MS-DOS commands into a set of custom-tailored commands to make your personal computer truly personal.

PREVIOUS NEXT

Chapter 17: Tailoring Your System

Overview

If you have followed the examples in the book, you have used all the major features of MS-DOS. Although it might sometimes seem that MS-DOS offers more options than you need, those options give you flexibility in tailoring MS-DOS; they let you adapt the computer to yourself rather than adapt yourself to the computer.

This chapter shows several ways you can tailor MS-DOS to your needs or preferences. Not only does this tailoring make MS-DOS fit your work needs better, but some of the techniques described here can also make your system immediately useful without your having to buy an application program. Some of these techniques can also make the system more accessible to people who may need to use the computer but who don't have your experience with MS-DOS. And tailoring can also make it easier to achieve consistency in procedures such as backing up a hard disk when several people use the computer.

This chapter describes the following:

- How to define configuration commands, which you put in the file named CONFIG.SYS to tell MS-DOS how to use the hardware devices that make up your system
- How to use the MemMaker program, which helps you maximize the amount of available conventional memory
- How to run MS-DOS, device drivers, and other programs in upper memory, leaving conventional memory for application programs
- How to manage your portable computer with two programs: Power, which can conserve battery life, and Interlnk, which transfers files quickly over a serial cable between two computers
- How to start your system *without* carrying out the commands in CONFIG.SYS or AUTOEXEC.BAT
- How to step through the commands in CONFIG.SYS and AUTOEXEC.BAT one at a time, choosing whether or not to carry out each one
- How to set up multiple configurations and choose the one you want when you start or restart the system
- Some less frequently used MS-DOS commands

Examples are included, but they're not step-by-step exercises. This chapter gives you some ideas about how to tailor MS-DOS by applying what you have learned in previous chapters.

Defining Your System Configuration

Unlike the other MS-DOS commands, which tell MS-DOS what to do, the configuration commands tell MS-DOS *how* to do something, such as using a device or communicating with a disk drive. These commands help define the computer setup, or *configuration,* so that MS-DOS can work well with your equipment. You don't type the configuration commands at the keyboard as you do other MS-DOS commands. You put them in a special file called CONFIG.SYS, which must be in the root directory of your MS-DOS disk. MS-DOS carries out the commands in CONFIG.SYS only when you start or restart the system, so if you change a command in this file, you must restart MS-DOS for the command to take effect.

If you purchase an application program or an accessory device that requires certain commands in CONFIG.SYS, either the installation program adds or modifies the needed commands or the documentation provides step-by-step instructions. Most other times, however, the system configuration remains stable, so you don't change the configuration commands very often. But as you become more familiar with MS-DOS and your computer, you might want to experiment with commands in CONFIG.SYS as a means of speeding up or refining the way your system works.

In version 5 or later, MS-DOS itself either creates a basic CONFIG.SYS file (and an AUTOEXEC.BAT file) for you or it modifies the existing version of each file as part of its installation procedure. In version 4, MS-DOS creates CONFIG.SYS and AUTOEXEC.BAT if they don't already exist; if they do exist, MS-DOS creates versions named CONFIG.400 and AUTOEXEC.400, which you can later combine with your existing CONFIG.SYS and AUTOEXEC.BAT files. Earlier versions of MS-DOS do not automatically create either CONFIG.SYS or AUTOEXEC.BAT.

You create or edit CONFIG.SYS with a text editor just as you create or edit a batch file. The following topics describe some of the configuration commands you might use on your system.

The Device Command and Device Drivers

The Device configuration command specifies a program (a special type of file whose extension is usually SYS) that tells MS-DOS how to use a particular device. Such a program is called a *device driver*. The form of the Device command is this:

device = <**filename>**

<filename> is the name of the file that contains the device driver. You can include a path name if the file is not in the root directory. If you have a mouse, for example, and the program that tells MS-DOS how to use it is named MOUSE.SYS in the C:\MOUSE directory, the device command you put in your CONFIG.SYS file would be *device=c:\mouse\mouse.sys*.

Although the word *device* brings hardware to mind, a Device command needn't always refer to a piece of equipment you can touch or carry. In certain instances, a "device" is actually a

program that simulates hardware. For example, MS-DOS includes a special program file that enables it to use a portion of memory on your system as if that memory were disk-based storage. The file, RAMDRIVE.SYS, enables MS-DOS to use memory as if it were an extremely fast disk drive instead. RAMDRIVE.SYS is described shortly. First, though, a little information about your computer's memory will help you understand how and when to use these device drivers.

Types of Memory and How They Differ

Your computer uses memory for temporary storage of calculations, data, active programs, and for other work in progress. Computer memory comes in three flavors: *conventional, extended,* and *expanded*.

Conventional Memory

Conventional memory is the type sometimes referred to as RAM (for *random access memory*). Computers that run MS-DOS usually come with 1 MB of conventional memory. This memory can be used to hold MS-DOS, application programs, and data. All computer programs can use conventional memory, but IBM and compatible computers running MS-DOS are limited to a maximum of 1 MB (1024 KB) of this type of memory.

Even if your computer has 1 MB of conventional memory, however, not all of it is freely available for applications, data, or even MS-DOS. MS-DOS normally uses only the first 640 KB of conventional memory for applications and data. The remaining 384 KB of conventional memory is known as the *upper memory area* or *reserved* memory and is set aside for special purposes such as hardware control and video memory. This reserved portion of memory is used by memory-management software in units called *upper memory blocks,* or UMBs. Beginning with version 5, MS-DOS can load device drivers and programs such as Doskey into unused UMBs, thus helping you conserve as much as possible of your computer's first 640 KB of memory for applications and data.

Extended Memory

Extended memory is additional memory that begins at the 1-MB boundary where conventional memory ends, so it literally extends beyond the limits of conventional memory. You can add many megabytes' worth of extended memory to a computer, but that does not necessarily mean you can use it. Extended memory can be used for storage only by programs specifically designed to find and take advantage of it. If your programs cannot use extended memory, you might as well not have any. Extended memory is often managed by a program called an *extended memory manager,* which keeps two applications from trying to use the same portion of extended memory at the same time.

Expanded Memory

If you think of extended memory as "high" memory that a program can reach with the right tools, you can think of expanded memory as a separate reservoir from which a program can

draw through a pipeline. Expanded memory must be handled by a program called an *expanded memory manager*. An expanded memory manager should come with any expanded memory you install in your system.

Once your computer is set up with extended memory, expanded memory, or both, you really needn't worry about what type you have or how it's used. It's the job of the managing software to do all that for you. But you can use MS-DOS, especially version 5 or later, to take advantage of extended, expanded, and reserved memory to make your system run faster and more efficiently.

Note If you don't have version 5 or later of MS-DOS, the following few topics don't apply to your system. Skip to the section called "Creating a Disk Cache with SMARTDRV" if you have the Microsoft release of version 4. Skip to the section called "Simulating a Disk Drive in Memory with RAMDRIVE.SYS" if you have version 3 or IBM's version 4 of MS-DOS.

Managing Memory

Your computer needs nothing more than MS-DOS to manage conventional memory, but when you add extended or expanded memory to a system, MS-DOS needs some extra help. In version 5, help appeared in the form of two device drivers named HIMEM.SYS and EMM386.EXE.

HIMEM.SYS is an extended memory manager. If extended memory is installed in your system when MS-DOS version 5 (or later) is installed, MS-DOS finds the memory and adds to CONFIG.SYS an appropriate Device command that names this device driver. If you install extended memory after installing MS-DOS, you can add your own Device command to CONFIG.SYS. For example, if HIMEM.SYS is in the C:\DOS directory, the command would be as follows:

```
device=c:\dos\himem.sys
```

Once you've identified HIMEM.SYS (or another extended memory manager) in CONFIG.SYS, you can turn some of the extended memory into a fast-access storage area, a simulated disk drive, or a home for MS-DOS itself.

On a computer with an 80386, 80486, or a Pentium microprocessor, you can accomplish additional memory management with EMM386.EXE. This memory manager performs two tasks:

- If your system has extended memory but your applications require expanded memory, EMM386.EXE can help MS-DOS treat some of that extended memory as if it were expanded memory.
- If your system has unused portions of the upper memory area available after all necessary amounts have been parceled out for hardware control, video, and other special purposes, EMM386.EXE can make those unused portions of memory

available to MS-DOS for loading device drivers and programs.

EMM386.EXE includes a number of parameters, most of which are needed only by programmers. Three that you might use, however, are described here. Assuming that EMM386.EXE is in the C:\DOS directory, the Device command and the three parameters look like this:

device = c:\dos\emm386.exe <memory> ram noems

<memory> tells MS-DOS how much extended memory, in kilobytes, to treat as expanded memory. You can specify 16 to 32768. If you don't include <memory>, MS-DOS assumes all available memory, rounding down to the nearest multiple of 16 if necessary. When you use this parameter, MS-DOS sets aside the required amount of extended memory for use as expanded memory. It also sets up your system's upper memory area so that part can be used to make the expanded memory available to programs, and the remainder can, if requested, be used for loading device drivers and programs.

ram tells MS-DOS to provide access to both expanded memory and upper memory (UMBs).

noems tells MS-DOS to set aside all available upper memory for loading device drivers and programs. If you use this parameter, you cannot use any expanded memory.

Include either *ram* or *noems* in the Device command that names EMM386.EXE. For example, a basic Device command in CONFIG.SYS would be this:

```
device=c:\dos\emm386.exe
```

To treat 640 KB of extended memory as expanded memory and make UMBs available, the command would be this:

```
device=c:\dos\emm386.exe 640 ram
```

To set aside all available reserved memory for loading programs and device drivers, the command would be this:

```
device=c:\dos\emm386.exe noems
```

Once you give MS-DOS access to upper memory, you can use this memory with either the Loadhigh command or the Devicehigh configuration command described in the section headed "Using the Upper Memory Area."

Running MS-DOS in High Memory

Through version 4, MS-DOS loaded itself into conventional memory, reducing the amount of memory available for applications and data by the amount of space it needed. Beginning with version 5, MS-DOS includes a simple configuration command named Dos that can help you conserve conventional memory on a system with extended memory, upper memory, or both.

If your system has at least 64 KB of extended memory, you can use the Dos command to tell MS-DOS to position itself in a portion of extended memory called the *high memory area* (HMA). If your system has at least 350 KB of extended memory (typical of 80386 systems with 1 MB of RAM), you can use the Dos command to make part or all of the upper memory area available for use by device drivers and programs. By moving MS-DOS, device drivers, and utility programs such as Doskey and other memory-resident software out of conventional memory, you leave as much of that valuable storage space as possible for your applications and data to use.

The Dos configuration command is this:

dos = high/low,umb/noumb

high tells MS-DOS to load itself into the high memory area if it can. (The high memory area is approximately the first 64 KB of extended memory.) *low* tells MS-DOS to load itself into conventional memory. If you don't specify *high,* or if MS-DOS can't load itself into the high memory area, MS-DOS automatically moves into conventional memory.

umb (short for upper memory blocks) tells MS-DOS to make use of the upper memory area for device drivers and programs. *noumb* tells MS-DOS not to use that area. If you don't specify *umb,* MS-DOS does not use the upper memory area.

You can put this command anywhere in CONFIG.SYS, and you can specify both parameters in one command by separating them with a comma. If you use the *high* parameter, however, also include a Device command that names HIMEM.SYS so that you can be sure MS-DOS can find and use extended memory. If you use the *umb* parameter, also include a Device command that names EMM386.EXE or another manager that makes the upper memory area available for use. The following two examples show different ways to use these commands in CONFIG.SYS.

To load MS-DOS into the high memory area on a system with at least 64 KB of extended memory, you would include these commands in CONFIG.SYS:

```
device=c:\dos\himem.sys
dos=high
```

To load MS-DOS into the high memory area and provide access to all of the available upper memory area on an 80386, 80486, or a Pentium system with no less than 350 KB of extended memory, you would include the following commands in CONFIG.SYS:

```
device=c:\dos\himem.sys
dos=high,umb
device=c:\dos\emm386.exe noems
```

Note The Device command naming HIMEM.SYS *must* precede the Dos command and the Device command naming EMM386.EXE in MS-DOS 5.0.

If you tell MS-DOS to load itself into extended memory, you can check its position with the

Mem command. MS-DOS should respond *MS-DOS is resident in high memory area*. If MS-DOS cannot find the high memory area and load itself in extended memory, you don't have to worry. It will load into conventional memory just as it always has.

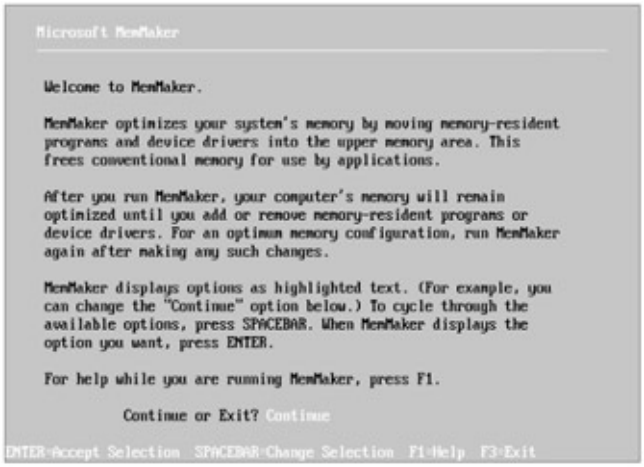
Letting MS-DOS Help You Manage Memory

To take full advantage of the MS-DOS memory-management features in version 5 and later, the proper configuration commands must be placed in CONFIG.SYS, specific device drivers should be loaded into high memory with the Devicehigh command, and memory-resident programs should be loaded into upper memory with Loadhigh commands in AUTOEXEC.BAT. You can add these commands yourself, or an experienced MS-DOS user can do it for you.

Starting with version 6.0, MS-DOS itself offers the same sort of help that the experienced user would. The MemMaker program analyzes your computer's memory and other programs that you use, and then adds the appropriate commands to CONFIG.SYS and AUTOEXEC.BAT to make best use of the memory installed in your system.

You should run MemMaker early in your quest to develop the perfect configuration. As you'll see later in this chapter, versions 6.0 and later of MS-DOS allow you to specify multiple configurations in your CONFIG.SYS file. MemMaker doesn't take the multiple configurations into account when it performs its memory optimization, so if you run it after breaking your CONFIG.SYS file into multiple configurations, your work might be lost.

To start MemMaker, just type *memmaker* at the system prompt. In a moment, MemMaker displays its opening screen, which describes the process it follows:



After you press Enter to continue, MemMaker displays a screen that allows you to choose between letting MemMaker use its default settings (Express Setup) or specifying the settings yourself (Custom Setup). Press Enter to choose Express Setup. The default settings will do a good job of optimizing your memory.

There isn't much for you to do from here on. As the instructions say, you'll be given a few choices, but these are mostly limited to decisions such as choosing between continuing with

MemMaker or returning to MS-DOS. In the course of deciding how best to manage your system's memory and putting the required commands in CONFIG.SYS and AUTOEXEC.BAT, MemMaker restarts your system twice.

After the second restart, MemMaker displays a summary of its actions, letting you know how much memory it freed for you. If your system wasn't set up for memory management before, you should now see significantly more conventional memory available. (It isn't unusual for 610 KB or even more to be free.) You might be able to free a bit more conventional memory by fine-tuning the commands in CONFIG.SYS and AUTOEXEC.BAT, but, by and large, MemMaker should do the job well enough by itself.

Creating a Disk Cache with SMARTDRV

If your computer has a hard disk, either extended or expanded memory, and version 6.0 or later of MS-DOS, you can load SMARTDRV in AUTOEXEC.BAT to turn part of this additional memory into a rapid-access storage area known as a *disk cache*. (If you have version 4 or 5, you can use the Device command and the device driver SMARTDRV.SYS in CONFIG.SYS to do the same thing.) A disk cache speeds up your system because information that your computer reads from disk into memory remains in the cache. It's much faster for your computer to read information that's already in memory, so a disk cache can substantially reduce the amount of time your computer spends going out to disk, finding the information you need, and reading it into memory.

Note If you're using the DriveSpace or DoubleSpace program to compress your hard disk, don't use SmartDrive to create a disk cache on the compressed disk. It will actually slow down access to that disk. You should, however, use SmartDrive on your uncompressed hard disk drive.

To create a disk cache with SMARTDRV, add a command in the following form to AUTOEXEC.BAT:

smartdrv <size> <minsize>

If the program SMARTDRV.EXE is not in the root directory of the current disk, precede the name with a drive and path—for example, *c:\dos\smartdrv*.

<size> is the size you want the cache to be, in kilobytes. It can be any value from 128 to 8192 (8 MB). If you don't specify <size>, SMARTDRV sets the cache size, depending on the amount of extended memory available. When the cache is created, MS-DOS rounds <size> to the nearest multiple of 16. If there is too little memory to create the size cache you specify, SMARTDRV creates a smaller one, using all the memory that is available.

<minsize> is the smallest size you want the cache to be, again in kilobytes. You don't have to specify <minsize>, but if you don't, and you use a program such as Microsoft Windows version 3 or later, the program might be able to reduce the size of your cache to suit its own purposes. In some cases, the program might reduce the cache size to 0.

In MS-DOS versions 6.0 and later, the disk cache is always created in extended memory, unless you specify otherwise. In versions 4 and 5, you can specify the /A parameter to create the cache in expanded memory. Because the disk cache is created in extended or expanded memory, you must include a command in your CONFIG.SYS file that identifies your extended or expanded memory manager. If you have version 4 or 5 of MS-DOS, the Device command specifying SMARTDRV.SYS must appear after the memory manager command in CONFIG.SYS.

Also, when you specify the size, leave enough memory for other programs that also run in extended or expanded memory. If you're not certain how much of this memory you have (or have available), use the Mem command to ask MS-DOS to report on memory usage.

Note SMARTDRV has other parameters that further specify how the disk cache is created and managed. See Appendix C, "MS-DOS Command Reference," for details.

The following example creates a 1-MB (1024-KB) disk cache with a minimum size of 256 KB. The example assumes that SMARTDRV.EXE is in the C:\DOS directory:

```
c:\dos\smartdrv 1024 256
```

Simulating a Disk Drive in Memory with RAM DRIVE.SYS

Disk drives are mechanical and quite slow compared to the computer's memory. Starting with version 3.2, MS-DOS lets you set aside a portion of the computer's memory for use as a simulated disk, making it possible for disk operations to be performed at memory speeds.

This simulated disk is called a *RAM disk* or *RAM drive* because it exists in your computer's memory (RAM) rather than as a solid piece of hardware. It's also known as a *virtual disk* because having a RAM disk is virtually the same as having another disk drive. A RAM disk is particularly useful on a system with extended or expanded memory because MS-DOS can create the RAM disk in either of these types of memory, leaving the computer's conventional memory for other uses.

On the surface, a RAM disk might not seem much different from a disk cache, described in the preceding section. However, it is different. You can think of a disk cache—one created either by SMARTDRV or by a separate disk-caching program—as a warehousing area for information you've already used. In contrast, creating a RAM disk is like adding an extremely fast disk drive to your system.

A RAM disk behaves like any other disk: It has a drive letter and a directory, and you can specify it in any command that refers to a disk. It is much faster than a real disk drive, however, and the difference is especially noticeable when you use commands, such as Copy, that work with disk drives or when you use application programs that access the disk frequently, as many word processors and database programs do. To use a RAM disk, you copy the files you need from a physical disk to the RAM disk after MS-DOS has started, then copy them back to the disk after you have completed the work. Copying them back is

particularly important because any changes you make to the files are recorded only in memory when you are using a RAM disk, and the contents of memory disappear whenever you turn off your computer.

MS-DOS assigns the next available drive letter to the RAM disk. On a system with one or two floppy disk drives and a hard disk, the next available letter is usually D.

Although a real disk drive has a fixed capacity, such as 360 KB or 1.44 MB, a RAM disk can have whatever capacity you want, within the limits described below for the command parameters. If your computer has enough memory, you can define more than one RAM disk by including more than one RAM-disk Device command in CONFIG.SYS.

If your computer is equipped with expanded or extended memory, you should tell MS-DOS to use this additional memory for your RAM disk, leaving your computer's conventional memory space for programs to use. If you have version 5 or later of MS-DOS and an 80386, 80486, or Pentium computer, you can also use the Devicehigh configuration command described later to load the program that creates the RAM disk into the upper memory area. (You can check on available upper memory blocks with the /C parameter of the Mem command.)

As efficient as it is, the remarkable increase in speed offered by RAM disks can have some drawbacks. If your computer doesn't have expanded or extended memory (or if you don't tell MS-DOS to use it), the memory used for a RAM disk reduces the amount of memory available to programs. Furthermore, the contents of a RAM disk are lost each time you turn the computer off.

If the RAM disk is large enough, you can speed operations by copying both programs and data files to it. Make certain that your RAM disk leaves enough memory for the programs you use. In either case, be sure to copy the files you want to keep onto a real disk before you turn the computer off. You can automate this process by using a batch file to copy your working files to the RAM disk, to start the application program, and then to copy the revised working files back to the real disk after you leave the application program.

Depending on your version of MS-DOS, the file that creates a RAM disk is called either RAMDRIVE.SYS or VDISK.SYS. To avoid excessive detail, the following description applies to RAMDRIVE.SYS as it can be used with versions 6.0 and later of MS-DOS. Differences in earlier versions and in VDISK.SYS are not major, but they do exist. If you need exact values, refer to Appendix C, "MS-DOS Command Reference."

To define a RAM disk, the Device command is this:

device = ramdrive.sys <size> <sector> <directory> /E /A

RAMDRIVE.SYS (or VDISK.SYS) is the name of the device-driver program. If it isn't in the root directory of the MS-DOS disk, you must include the drive letter and path name of its directory—for example, *c:\dos\ramdrive.sys*.

<size> is the size, in kilobytes, of the RAM disk. The minimum is 4 KB, and the maximum is

32,767 KB (32 MB). If you omit <size> or specify an incorrect value, RAMDRIVE.SYS sets <size> to 64.

<sector> is the size, in bytes, of each sector on the RAM disk. You can specify 128, 256, or 512. If you omit <sector> or specify an incorrect value, RAMDRIVE.SYS sets <sector> to 512. If you use <sector>, you must also use <size>.

<directory> is the number of directory entries allowed in the root directory of the RAM disk. You can specify any value from 2 to 1024. Each directory entry takes up 32 bytes of the RAM disk. If you omit <directory> or specify an incorrect value, RAMDRIVE.SYS sets <directory> to 64. If you use <directory>, you must also use <size> and <sector>.

/E puts the RAM disk in extended memory. It is valid only if the computer contains extended memory. Using extended memory for a RAM disk leaves the maximum amount of conventional memory for programs. If you use /E, you cannot use /A.

/A puts the RAM disk in expanded memory. Like the /E option, it leaves the maximum amount of conventional memory available for programs. If you use /A, you cannot use /E.

If there isn't enough memory to create the RAM disk as you specify it, and you're creating the disk in conventional memory, RAMDRIVE.SYS doesn't create the disk. If you're creating the disk in extended or expanded memory, RAMDRIVE.SYS creates a disk using all available memory.

The following examples show the commands that would create RAM disks in conventional, extended, and expanded memory. All three examples assume that RAMDRIVE.SYS is in the C:\DOS directory.

To create a small (64-KB) RAM disk in conventional memory, accepting the values MS-DOS assumes for <sector> and <directory>, you would place either of the following commands in CONFIG.SYS:

```
device=c:\dos\ramdrive.sys 64
```

or:

```
device=c:\dos\ramdrive.sys
```

To create a much larger (720-KB) RAM disk in extended memory, again accepting the values MS-DOS assumes for <sector> and <directory>, you would need two commands: first, a Device command to identify the extended memory manager to MS-DOS, and second, a Device command to create the RAM disk in extended memory. The commands would look like this:

```
device=c:\dos\himem.sys
device=c:\dos\ramdrive.sys 720 /e
```

Finally, if you were to create the RAM disk described in the preceding example, but in

expanded rather than extended memory, the commands would be these:

```
device=<manager>
device=c:\dos\ramdrive.sys 720 /a
```

When you create this RAM disk on your own system, you need to replace *manager* with the drive, path, and file name of your expanded memory manager.

Using the Upper Memory Area

If you have an 80386, 80486, or Pentium computer, chances are your system has more than 1 MB of memory. If you have version 5 or later of MS-DOS, you can use the Loadhigh and Devicehigh commands to load programs and certain device drivers into upper memory blocks (UMBs). Both commands can help you conserve regular memory for applications and data.

Loadhigh is an MS-DOS command that you type or use in a batch file. Devicehigh is a configuration command that you put in CONFIG.SYS. Because MS-DOS doesn't normally treat the upper memory area as part of your computer's available memory, you can use the Loadhigh and Devicehigh commands only after you have loaded the following:

- The HIMEM.SYS device driver, which enables MS-DOS to use high memory
- The Dos configuration command, which tells MS-DOS whether to use available upper memory blocks (UMBs)
- The EMM386.EXE device driver or a similar device driver that makes unused blocks of the upper memory area available

Once you've used the Dos command and have identified the appropriate memory managers to MS-DOS with Device configuration commands, you can use the Loadhigh and Devicehigh commands to tell MS-DOS what to load into the upper memory area. To see if you have enough memory for a particular program or device driver, compare the file's size with the free upper memory reported by the Mem /C command.

The following sample CONFIG.SYS file shows the Dos and Device commands you need to use Loadhigh and Devicehigh. Files are assumed to be in the C:\DOS directory. In this example, the Dos command also tells MS-DOS to run in high memory, and the *noems* parameter of the Device command naming EMM386.EXE indicates that the system has no need for expanded memory:

```
device=c:\dos\himem.sys
dos=high,umb
device=c:\dos\emm386.exe noems
```

Loading Programs with Loadhigh

The Loadhigh command can be used either from the system prompt or, for programs you want to load regularly into the upper memory area, from your AUTOEXEC.BAT file. Loadhigh, which can be abbreviated lh, is particularly well suited for use with the type of program known as a TSR (*terminate and stay resident*). Such programs sit in memory but usually reside quietly in the background, as opposed to word processors and other application programs that remain in control of the system as long as they are active. MS-DOS includes a number of TSRs, among them Doskey, Graphics, Mode, and Append.

To load a program into upper memory, the form of the Loadhigh command is this:

loadhigh <filename> <parameters>

<filename> is the name of the program you want to load, including a drive and path if necessary.

<parameters> represents any parameters you type when starting the program.

For example, after starting with the CONFIG.SYS file described earlier, you would load Doskey into upper memory with this command:

```
loadhigh doskey
```

MS-DOS would then attempt to load the program into the upper memory area. If there weren't enough room, MS-DOS would load Doskey into conventional memory instead.

Loading Device Drivers with Devicehigh

The Devicehigh command lets you load device drivers, such as RAMDRIVE.SYS, into the upper memory area and thus conserve as much of your computer's conventional memory as possible. Device drivers that are included with MS-DOS and that can be loaded into the upper memory area are EGA.SYS, DISPLAY.SYS, DRVSPACE.SYS, DBLSPACE.SYS, INTERLNK.SYS, ANSI.SYS, RAMDRIVE.SYS, and DRIVER.SYS. (MS-DOS versions prior to 6.0 also included PRINTER.SYS.)

Note Before experimenting, create a startup floppy disk (use the /S parameter of the Format command), and copy your CONFIG.SYS and AUTOEXEC.BAT files to it. Then if you experience difficulties with a device driver, you have an alternative means of starting your computer so that you can correct the situation. Versions 6.0 and later of MS-DOS makes it possible to skip the loading of CONFIG.SYS and AUTOEXEC.BAT, so there's no need to create the startup disk. See the section "Starting Without CONFIG.SYS or AUTOEXEC.BAT" later in this chapter.

You can also load device drivers that aren't part of MS-DOS into reserved memory. Some, however, need more memory than expected when they're loaded and might cause your system to halt. If this happens, you might be able to fix the problem by using the <minsize> parameter of the Devicehigh command to specify the amount of memory the driver needs.

The form of the Devicehigh command in versions 6.0 and later is this:

devicehigh = /L:<region>,<minsize> /S <filename>

/L:<region>,<minsize> lets you specify where in upper memory you want to place the driver and the minimum amount of memory needed by the device driver. <region> specifies the region in upper memory into which you want to put the device driver. To list the regions available in upper memory, use the Mem command with the /F parameter. <minsize> tells MS-DOS the smallest amount of memory into which the device driver will fit. If a device driver needs two or more regions of memory, separate the region numbers with semicolons; if <minsize> is used with the first <region> specification, then the semicolon follows the <minsize> specification. Use <minsize> if you experience problems loading a device driver into the upper memory area.

/S shrinks the UMB to its smallest size as MS-DOS loads the device driver. This switch is normally used only by the MemMaker program.

<filename> is the name, including extension, of the device driver you want to load. Include a path if the driver is not in the root directory of the disk from which you start MS-DOS. You can also include parameters required by the device driver.

Version 5 of MS-DOS has a slightly different form:

devicehigh size = <memsize> <filename>

size=<memsize> is the amount of memory, given as a hexadecimal value, required by the device driver. Use *size* if you experience problems loading a device driver into the upper memory area. Although the value must be a hexadecimal (base 16) number, you don't have to calculate it yourself; you can use the Mem command with the /C parameter to list the size of the driver. The second column of the Mem command report gives the size of the device driver in decimal; the third column gives the size in hexadecimal. If hexadecimal is new to you, don't be disconcerted if you see a combination of letters and numerals; both 0004A0 and 0038E0, for example, are valid hexadecimal numbers.

<filename> is the name, including extension, of the device driver you want to load. Include a path if the driver is not in the root directory of the disk from which you start MS-DOS. You should also include parameters that are required by the device driver.

When you use Devicehigh, MS-DOS attempts to load the specified device driver in the upper memory area. If there isn't enough room, MS-DOS loads the driver into conventional memory instead.

The following sample CONFIG.SYS file for version 6.0 or later of MS-DOS includes the commands shown earlier as well as a Devicehigh command that loads RAMDRIVE.SYS into the upper memory area and creates a 640-KB RAM disk in extended memory:

```
device=c:\dos\himem.sys
dos=high,umb
device=c:\dos\emm386.exe  noems
devicehigh=c:\dos\ramdrive.sys  640 /E
```

Controlling the Display with ANSI.SYS

MS-DOS includes a device driver called ANSI.SYS that defines a standard set of methods for managing a display, including how to display and erase characters, move the cursor, and select colors. Some programs, including parts of the MS-DOS Mode command, require your system disk to have a CONFIG.SYS file that contains the command *device=c:\dos\ansi.sys* (assuming the file is in your C:\DOS directory). ANSI is an acronym for American National Standards Institute.

Configuration Commands for Your Portable Computer

Portable computers have special needs, some of which are met by two device drivers introduced in version 6.0 of MS-DOS:

- POWER.EXE extends battery life by shutting down portions of the computer's circuitry when no activity is taking place. If the computer conforms to the Advanced Power Management (APM) specification, POWER.EXE can extend battery life as much as 25 percent.
- INTERLINK.EXE lets you connect a portable computer to a desktop computer with a cable and quickly copy files back and forth.

Both programs are fairly simple to use. To take advantage of POWER.EXE, for example, add the configuration command *devicehigh=c:\dos\power.exe* to your CONFIG.SYS file. After you start the system, type the command *power*. If you plan on using these power-saving features permanently, add the Power command to AUTOEXEC.BAT.

To use INTERLNK.EXE, you must have a cable to connect the computers; the most common is a 3-wire serial cable, but you can also use a 7-wire null modem cable or a bidirectional parallel cable. If version 6.0 or later is installed on both computers, add the configuration command *devicehigh=c:\dos\interlnk.exe* to CONFIG.SYS on the portable computer. On the desktop computer, simply type the command *intersvr* (you don't have to add a command to CONFIG.SYS). Disk drives attached to the larger computer appear as drives to the portable; copying from one machine to the other is simply a matter of using the Copy or Xcopy commands to copy files from one drive to another.

For more information about using these two device drivers, use the MS-DOS Help facility (for example, type *help power.exe, help interlnk.exe,* or *help intersvr*) or consult the manual that came with MS-DOS version 6.0 or later.

Other Configuration Commands

Several configuration commands control internal operating characteristics of your system and usually deal with how MS-DOS handles files or reads disks. Some application programs or devices include detailed instructions for adding or changing these configuration commands. The configuration commands discussed in this section are described in more detail in Appendix C, "MS-DOS Command Reference."

Defining Temporary Work Areas

The Buffers configuration command defines the number of work areas in memory (*buffers*) that MS-DOS uses to read from and write to a disk. The effect of this configuration command on system performance depends on the type of disk drive you use and the types of programs you use. The form of the Buffers command is this:

buffers = <number>

Unless otherwise noted, versions of MS-DOS through 3.2 use two or three buffers, depending on your system and the amount of memory it has. If you're using version 3.3, MS-DOS sets the number of buffers primarily according to the amount of memory your system has. You can, of course, override these values by including or changing the Buffers configuration command in CONFIG.SYS. For optimum performance, some programs require you to set buffers to a higher number than MS-DOS assumes. Fastback, for example, needs 40. For a list of values that MS-DOS uses, see the description of the Buffers command in Appendix C.

Specifying the Maximum Number of Open Files

The Files configuration command tells MS-DOS how many files it can use at one time. Unless otherwise instructed, MS-DOS can use a maximum of eight files at a time. The form of the Files command is this:

files = <number>

<number> can be any number from 8 through 255.

Setting the Highest Drive Letter

The Lastdrive configuration command specifies the highest drive letter that MS-DOS recognizes as valid. If CONFIG.SYS doesn't contain a Lastdrive command, the highest drive letter MS-DOS recognizes is E. This command is usually used to specify a higher letter (up to Z) if MS-DOS needs more than five drive letters. MS-DOS might need more drive letters because the computer is part of a network, because it uses many RAM disks, or because a large hard disk is divided into sections, each of which is referred to by a different drive letter. The form of the Lastdrive command is this:

lastdrive = <letter>

<letter> is any letter from A to Z.

Troubleshooting and Tailoring System Startup

When your system behaves erratically—one or more programs won't run or one or more runs incorrectly, or perhaps a device doesn't work properly—one of the most valuable troubleshooting techniques is to remove commands from CONFIG.SYS and AUTOEXEC.BAT until the problem goes away. Starting with version 6.0, MS-DOS takes much of the tedium out of this procedure by letting you start the system without carrying out any of the commands in CONFIG.SYS or AUTOEXEC.BAT or by letting you choose, command by command, whether to carry out the commands in CONFIG.SYS and AUTOEXEC.BAT.

In MS-DOS version 6.0 or later, you can also structure the commands in CONFIG.SYS so that MS-DOS displays a menu of alternate configurations and lets you choose which combination of configuration commands to carry out.

Starting Without CONFIG.SYS or AUTOEXEC.BAT

If you install a new program or device and your system stops working properly—or if someone else uses your system and now it won't do what it's supposed to—there's a chance that either a configuration command in CONFIG.SYS or a command in AUTOEXEC.BAT is causing the problem.

If you're using version 6.0 or later, confirm the diagnosis by pressing Ctrl-Alt-Del to restart the system and when MS-DOS displays *Starting MS-DOS,* press F5; MS-DOS skips both CONFIG.SYS and AUTOEXEC.BAT. If the system starts properly (although it may behave differently from the way it usually does because you aren't carrying out your normal configuration or startup commands), edit your CONFIG.SYS and AUTOEXEC.BAT files to double-check any commands you might have added. If you didn't add any commands recently, you'll need to do further investigating. The next topic tells you how to use another feature to pinpoint the problem.

Choosing Individual Startup Commands

If there's a problem in CONFIG.SYS (MS-DOS version 6.0 or later) or AUTOEXEC.BAT (MS-DOS version 6.2 or later), you can test their commands one at a time by pressing F8 when MS-DOS displays *Starting MS-DOS* at startup. In response, MS-DOS displays each configuration command in CONFIG.SYS, starting with the first, and prompts you *[Y,N]?* Step through the file this way until you locate the offending configuration command; then either correct it or remove it.

When MS-DOS has completed processing CONFIG.SYS, it prompts you with *Process AUTOEXEC.BAT [Y,N]?* In MS-DOS versions 6.2 and later, to step through each command in AUTOEXEC.BAT just as you did in CONFIG.SYS, press *Y*. (In version 6.0, the AUTOEXEC.BAT file will be processed, but you will not be able to step through it one

command at a time).

Note If you call or chain to other batch files in AUTOEXEC.BAT, MS-DOS versions 6.2 and later step through those files one command at a time, too, giving you complete control of your diagnostic process.

Defining a Menu of Configurations

Suppose you have a device that you use rarely—for example, a color scanner—whose device driver is very large and cannot run in high memory. You have also found that it runs much better if you create a RAM drive to hold some temporary files created during the scan, which uses more memory. But because the driver and the created RAM drive take up memory that otherwise would be available for other programs, you'd like to load the device driver and set up the RAM drive only when you're going to use the scanner. Versions 6.0 and later let you do just that by adding some commands to your CONFIG.SYS file.

Let's look at an example. When you start your machine using MS-DOS version 6.2 and the following CONFIG.SYS file, MS-DOS displays a screen titled "MS-DOS 6.2 Startup Menu" that lists two menu items—*NoScanner* and *Scanner*. MS-DOS reads different parts of your CONFIG.SYS file depending on the item you choose.

```
device = c:\dos\himem.sys
device = c:\dos\emm386.exe noems
dos = high,umb
lastdrive = f

[Menu]
menuitem = NoScanner
menuitem = Scanner

[NoScanner]

[Scanner]
device = c:\scan\colrscan.sys
devicehigh = c:\dos\ramdrive.sys 1024 512 64 /e

[Common]
devicehigh = c:\dos\drvspace.sys /move
devicehigh = c:\dos\ansi.sys
buffers = 20
files = 40
```

The first four commands set up upper memory management and specify the highest drive letter that MS-DOS will recognize; these configuration commands are always carried out.

The line that contains *[Menu]* marks the beginning of the list of menu choices. There are two

choices in this configuration menu—*NoScanner* and *Scanner*. You'll need a separate section in CONFIG.SYS for each of these choices to specify what configuration commands, if any, are to be carried out when each choice is made.

The first menu item section starts out with the line *[NoScanner].* This section contains no configuration commands because you don't want to take any special action if you're not going to use the scanner.

The second menu item section also starts with the line *[Scanner].* This section contains two configuration commands that are carried out if you choose *Scanner* from the startup menu. They load the device driver named COLRSCAN.SYS and create a 1-megabyte RAM drive.

The four configuration commands following *[Common]* are carried out regardless of what you choose from the startup menu. These commands complete the job of specifying to MS-DOS how your system is set up and how to manage its resources.

By setting up your CONFIG.SYS file this way, you don't have to give up the use of the memory required to run your scanner when you don't use it. Each time you start your system, you'll have the chance to tell MS-DOS how to configure the system. You can use this technique to handle any special configurations, such as running a network or installing drivers for seldom-used hardware.

PREVIOUS NEXT

Commands for Occasional Use

So far this book has described all the commands you routinely use to operate MS-DOS. There are a few remaining commands you might occasionally need, and there are several commands that you won't need unless you plan to program or use some of the advanced capabilities of MS-DOS. The less commonly used commands are described briefly here and in more detail in Appendix C, "MS-DOS Command Reference."

Displaying the MS-DOS Version Number

The Version command displays the number of the version of MS-DOS you're using. If you use more than one version, or if you are using someone else's machine, this gives you a quick way to check the version.

The Version command has no parameters:

ver

If you're using version 6.0, for example, MS-DOS replies *MS-DOS Version 6.00* in response to the command.

Changing the System Prompt

As shown in examples in earlier chapters, you can change the system prompt with the Prompt command to display much more than just the current drive letter. The change takes effect as soon as you enter the command.

The Prompt command has one parameter:

prompt <string>

<string> is a string of characters that defines the new system prompt. You can use any characters you want. You can also instruct the new prompt to include one or more items of useful information by including a dollar sign and one of the following characters to specify what you want the prompt to contain:

Character	Produces
d	The current date
p	The current drive and directory
n	The current drive
t	The current time
v	The MS-DOS version number
g	A greater-than sign (>)
l	A less-than sign (<)

b	A vertical bar (|)
q	An equal sign (=)
e	An escape character
h	A backspace
$	A dollar sign ($)
_	A signal to end the current line and start a new one (The character is an underscore, not a hyphen.)

You can include as many combinations of $, followed by a character, as you wish. MS-DOS ignores any combination of $ followed by a character that is not in the preceding list.

If you enter the Prompt command with no parameter (you just type *prompt*), MS-DOS restores the prompt to the letter of the current drive followed by a greater-than sign (for example, C>).

The Prompt command takes effect immediately, so it's easy to experiment. You saw how to change the system prompt to a courteous request (*May I help you?*) in Chapter 3, "Getting Your Bearings," and you've changed it to suit your needs in other chapters. To restore the system prompt to its most common form, type *prompt pg*.

Several examples follow. Notice how the system prompt changes each time to show the effect of the previous Prompt command. Press the Spacebar before pressing the Enter key to end each command in order to leave a space between the end of the system prompt and the beginning of the command that you type next.

To define the system prompt as the current drive and directory, type this:

```
C:\MKT\WP>prompt $p

C:\MKT\WP _
```

The example assumes that the current drive is C and that the directory is \MKT\WP. To define the system prompt as two lines that show the date and time, type this:

```
C:\MKT\WP prompt $d$_$t

Thu 01-05-1995
14:57:10.11 _
```

The time and date will vary, depending on how you have set them in your system. Press the Enter key several times to see that MS-DOS keeps the time current.

Finally, combining several of the options shows just how much you can include in a prompt:

```
Thu 01-05-1995
15:02:10.11 prompt $v$_$d $t$_Current directory $q $p$_Command:
```

MS-DOS Version 6.00
Thu 01-05-1995 15:02:57.68
Current directory = C:\MKT\WP
Command: _

The Prompt command lets you easily tailor the system prompt to the balance of information that you prefer. When you design a system prompt that you like, put it in AUTOEXEC.BAT, and you'll never have to type it again; MS-DOS will carry out the command every time you start the system.

Speeding Up File Access

Each time you (or an application program) need a file, MS-DOS might first need to search for the subdirectory that contains the file and then search the directory entries for the file itself. On a hard disk with hundreds or thousands of files, all this searching can take some time.

The Fastopen command tells MS-DOS to keep track (in memory) of the locations of subdirectories and files as it uses them. The next time it's asked for a file or subdirectory, MS-DOS checks in memory before it searches the disk. If it finds the location of the file or subdirectory in memory, MS-DOS can go directly to it on the disk instead of searching for it.

If you or your application programs tend to use the same files or directories over and over, the Fastopen command can make MS-DOS visibly faster. The Fastopen command works only with hard disks. (Note that you shouldn't use Fastopen if you're running Microsoft Windows. You should also avoid using a defragmentation program such as Defrag while Fastopen is loaded.) Fastopen has three parameters:

fastopen <drive> = <files> /X

<drive> is the drive letter, followed by a colon, of the hard disk whose files and subdirectories you want MS-DOS to track (for example, *c:*).

<files> is the number of files and subdirectories whose location MS-DOS will keep in memory; it must be preceded by an equal sign. You can specify a value for <files> from 10 through 999 (the default value is 48). For example, if you had one hard disk, drive C, and you wanted MS-DOS to keep track of the last 75 files and subdirectories used, you would type *fastopen c:=75*.

/X tells MS-DOS to keep track of the locations in expanded memory. If you use /X, check to be sure your expanded memory conforms to the current standard, which is LIM EMS 4.0.

If you're using MS-DOS from a hard disk, the installation program might have set up a Fastopen command telling MS-DOS to keep track of the last *x* files and subdirectories you have used. This command would have been placed in your CONFIG.SYS file in a form like this: *install=c:\dos\fastopen.exe c:=50*. Although the command looks a bit more complicated than *fastopen c:=50,* its purpose is the same. The *install* part of the command (described in Appendix C, "MS-DOS Command Reference") simply tells MS-DOS to use your computer's

memory as efficiently as it can. If you want to change the number of files, you can change the command in CONFIG.SYS. For example, to keep track of the last 75 files and subdirectories, change the command to *install=c:\dos\fastopen.exe c:=75*.

Treating a Directory as if It Were a Disk Drive

The Substitute (subst) command lets you treat a directory as if it were a separate disk. If your directory structure includes long path names, or if you use application programs that accept a drive letter but not a path name, you can use the Substitute command to tell MS-DOS to treat all future references to a particular drive as references to a directory on the disk in a different drive.

After naming a drive letter in a Substitute command, you cannot assign to that drive letter in any other command, so you will want to use a drive letter that doesn't refer to an existing drive. In order to do this, you must tell MS-DOS to accept more drive letters than there are disk drives. You can put a Lastdrive command in the CONFIG.SYS file in the root directory of your MS-DOS system disk. The Substitute command has three parameters:

subst <drive> <pathname> /D

<drive> is the letter, followed by a colon, to be used to refer to <pathname>.

<pathname> is the path name of the directory to be referred to by <drive>.

/D deletes any substitutions that involve <drive>. If you include /D, you cannot include <pathname>.

If you omit all parameters and type only *subst,* MS-DOS displays a list of any substitutions in effect. For example, suppose you find yourself frequently referring to a directory whose path name is C:\SPREAD\SALES\FORECAST and you would like to use a shorter synonym. To substitute *x:* for the path name, you would make certain that your CONFIG.SYS file contained a Lastdrive command that specified the letter *x, y,* or *z*, and then you would type *subst x: c:\spread\sales\forecast*. The substitution would remain in effect until you restarted MS-DOS or canceled the substitution by typing *subst x: /d*.

Part III: Appendixes

Appendix List

Appendix A: Installing MS-DOS

Appendix B: Glossary

Appendix C: MS-DOS Command Reference

Part Overview

Part 3 consists of three appendixes. Appendixes A and B contain reference material for your occasional use or for background information. Appendix A, "Installing MS-DOS," is a guide to installing and upgrading to versions 3 through 6.22 of MS-DOS. It supplements the step-by-step instructions you receive with each new version of MS-DOS. Appendix B, "Glossary," contributes to your background knowledge by defining commonly used terms and those presented in the main portion of this book. Appendix C, "MS-DOS Command Reference," is a comprehensive reference to MS-DOS commands and their parameters. It provides easy-to-find answers to questions about the form and use of commands.

PREVIOUS NEXT

Appendix A: Installing MS-DOS

This appendix shows you how to install versions 3 through 6.22 of MS-DOS. Although MS-DOS was probably preinstalled on your system, someday you might need to reinstall MS-DOS or upgrade to a newer version. If you need more background information or specific instructions, refer to the documentation that comes with your version of MS-DOS.

Installing or Upgrading to Version 5 or Later

Although MS-DOS has grown in capability over the years, it has also become easier to install. Versions 5 and later are the easiest to set up, whether you're installing MS-DOS on a brand-new system or upgrading from an earlier version. The first of the several floppy disks on which versions 5 and later are supplied contains a setup program (SETUP.EXE) that does almost all the work for you.

Whether you're upgrading or installing MS-DOS on a brand-new machine, *use the Setup program*. Even if you've installed earlier versions of MS-DOS by copying files from the MS-DOS floppy disks, you can't install version 5 or later in this way. Many of the MS-DOS files are shipped in a special condensed format that Setup expands during the installation procedure. In addition, Setup checks the memory and devices on your system (so you don't have to), offers help at each stage, prompts whenever it needs a response from you, displays an indicator showing how much of the installation is complete, and even tells you what it's doing as it prepares MS-DOS to work on your computer.

Installing Version 5 or Later for the First Time

If your system is new, chances are that MS-DOS has already been installed for you, especially if your computer has a hard disk. If MS-DOS is not installed, start by inserting the floppy disk labeled *Disk 1* in drive A. Then start or restart your computer.

Setup guides you through the entire installation process, asking you to provide some responses and to make a few choices (for example, whether you want to install the MS-DOS or Microsoft Windows versions of some features, or both). Setup proposes a response whenever it asks you to make a choice, usually either the one most people choose or the one that Setup has determined suits your system best. If you need further information before deciding to accept or change a proposed response, press the F1 key. Accepting all the choices Setup proposes will set up a basic working system for you. Once you're comfortable with this system, you can make any necessary modifications by tailoring the files AUTOEXEC.BAT and CONFIG.SYS to suit your own preferences. These files are described in Chapter 14, "Creating Your Own Commands"; Chapter 17, "Tailoring Your System"; and Appendix C, "MS-DOS Command Reference." After it's done installing MS-DOS, Setup restarts your machine.

Upgrading to Version 5 or Later

If you have a hard disk and are upgrading from an earlier version of MS-DOS, you can upgrade without a worry. Setup provides the Uninstall program, which is a built-in program that lets you restore your earlier version of MS-DOS if the installation fails or if you find that some of your programs cannot work with the new version.

Label a blank floppy disk *Uninstall 1* (more than one may be necessary if you are using low-density floppy disks). Then place the floppy disk labeled *Disk 1* in drive A, and type the

following:

```
C>a:
A>setup
```

After generating a preinstallation report of your machine's configuration and checking your system, Setup displays a welcoming message and tells you which keys to press to request help, continue with the installation, or quit. From this point on, follow the on-screen instructions. If you are uncertain about any responses, press the F1 key for help and further information. After it's done installing MS-DOS, Setup restarts your machine.

After the new version is installed, place the *Uninstall* floppy disk(s) in a safe place. It's unlikely that you'll need to return to your old version of MS-DOS, but if you do, place the *Uninstall 1* floppy disk in drive A, restart the system, and follow the instructions that appear on the screen.

PREVIOUS NEXT

Installing or Upgrading to Version 4

Version 4 of MS-DOS comes with an installation program that prompts you through the installation process. The installation program includes online help that explains most of the choices.

Essentially, all you do is place the floppy disk labeled *Install* in drive A, start or restart your computer, and then press the Enter key to begin. From that point onward, the installation program, named Select, asks you questions about your computer system and tells you which of the original MS-DOS floppy disks it needs. Before you start, however, there are a few things you should know or prepare for to make the procedure as easy as possible.

Your Computer

Find out (roughly) how much memory your system has. The Select program doesn't ask you for the exact amount, but during installation it asks you to choose how you want to balance the memory used by MS-DOS and by your programs.

If your system contains less than 512 KB of memory, choose the first option (*Minimum MS-DOS function; maximum program workspace*) to give your programs as much working room as possible. If your system has 512 KB of memory, choose the option Select proposes (*Balance MS-DOS function with program workspace*). If your system has more than 512 KB of memory, choose the third option (*Maximum MS-DOS function; minimum program workspace*) so that both MS-DOS and your programs have plenty of room to work in.

Check the make and model of printer you use. Also, find out whether it is a serial or a parallel printer, and find out the number of the port it is connected to (for example, LPT1: for a parallel printer or COM1: for a serial printer). The Select program will ask you to choose your printer from a list. It will also propose a printer port, so it is up to you to know whether Select is correct.

The Installation/Upgrade Process

Note If you're using a hard disk and are upgrading to version 4 from an earlier version of DOS, don't try to install IBM's release of DOS if your computer is currently running a version of DOS released by another manufacturer. There is a small but significant difference between IBM's release and others that prevents successful installation. You can install the IBM release, but to do so you must back up your hard disk, reformat it, and then install DOS. If you're not sure whose release of DOS you have, start your computer to display the information; if DOS doesn't display the manufacturer's name and the version number, type *ver* (the Version command).

Whether you install MS-DOS on a hard disk or on floppy disks, Select needs one or more blank floppy disks for the installation. One of the first things Select does is display a list

showing the number and capacity of floppy disks it needs for installing MS-DOS on various types of disk drives.

AUTOEXEC.BAT and CONFIG.SYS

When Select finishes putting MS-DOS on a hard disk that contains an earlier version of MS-DOS, its final message might tell you that Select has saved information about your computer, printer, display, screen, and the MS-DOS Shell in two files that it named AUTOEXEC.400 and CONFIG.400. You'll want MS-DOS to be able to find this information whenever you start or restart your computer, and for that to happen, you'll have to merge those files with the existing AUTOEXEC.BAT and CONFIG.SYS files. Only then will MS-DOS automatically search for the information saved by the Select program.

PREVIOUS NEXT

Installing Version 3

Versions 3.3 and earlier of MS-DOS do not include an installation program, so it's up to you to determine how you want to set up MS-DOS to work on your system. If your computer doesn't have a hard disk, all you really need to do is make copies of your MS-DOS floppy disks using the Diskcopy command you saw described in Chapter 6, "Managing Your Floppy Disks." Diskcopy makes an exact duplicate of a floppy disk, so once you've copied your original MS-DOS floppy disks, you can store them safely away and use the copies in your everyday work with MS-DOS.

If you have a hard disk, you use a two-step procedure to copy MS-DOS from your MS-DOS floppy disk(s) to the hard disk. The first step uses the System (sys) command to copy two hidden files that MS-DOS needs in order to start from your hard disk. The second step uses the Copy command to copy the remaining MS-DOS files to your hard disk.

Note The following instructions assume that your hard disk has already been formatted (prepared for use). If it has not, or if you want to wipe it clean and start anew, you can boot from the MS-DOS startup floppy disk, and then use the command *format c: /s* to prepare the disk and copy the system files. If you try this command, however, and MS-DOS displays a *WARNING* message followed by a request for you to confirm the format, be very sure you want to continue. Once MS-DOS begins formatting the hard disk, you'll lose whatever files it contains—forever.

Copying the System Files

Place your MS-DOS startup floppy disk in drive A. Depending on the version you're installing, this floppy disk is labeled either *Startup* or *MS-DOS*. Turn on the computer or restart it by simultaneously pressing the keys marked Ctrl, Alt, and Del.

If you're upgrading to a newer release of version 3, or if your hard disk is already formatted, use the System command to move the hidden MS-DOS files to your hard disk. Type the following:

```
A>sys c:
```

MS-DOS responds *System transferred*.

Note If you're installing version 3.3 on a system using version 3.1 or earlier, MS-DOS might display the message *Insufficient memory for system transfer*. If you see this, it means that MS-DOS cannot transfer the files properly. You must back up all the files on your hard disk, format it, and later restore all the files you've backed up.

Whether you use the Format or System command, your next step is to copy the MS-DOS files from the MS-DOS floppy disks to a directory named \DOS on your hard disk. The question now is, Where do they go? If you're working with a new or newly formatted hard disk, follow the instructions in the next section, "Copying MS-DOS to a DOS Directory." If

you're upgrading an existing MS-DOS version from one release to another (for example, from 3.1 to 3.3), skip to the section called "Upgrading with Version 3."

Copying MS-DOS to a DOS Directory

If you've just formatted your hard disk, it's an empty storage area you can use as you like, so make your future work easier by copying MS-DOS to a special section—a directory—of its own. Doing this keeps your MS-DOS files separate from application and data files, and it generally helps keep your hard disk neat and organized.

If you're new to MS-DOS, the following commands probably won't mean much to you, but if you type them exactly as shown, you should be able to install any release of version 3 without problems.

First, check that your startup disk is still in drive A, and copy the MS-DOS file named COMMAND.COM to the base (root) directory or your hard disk, where it normally goes. Type this:

```
A>copy command.com c:\
```

Now create the MS-DOS directory by typing this:

```
A>md c:\dos
```

Copy the files from your MS-DOS disk to the new DOS directory:

```
A>copy *.* c:\dos
```

If you have more than one MS-DOS disk, repeat the Copy command to copy the rest of your MS-DOS files.

When you typed *.* (meaning "all files") in your Copy command, you recopied COMMAND.COM. You don't need two copies of this file, so delete the one in your DOS directory by typing this:

```
A>del c:\dos\command.com
```

Finally, create a special file, again in the root directory of your hard disk, that tells MS-DOS where to find its own command files. Type the following. Press Enter at the end of each line, and press—at the same time—the two keys marked Ctrl and Z where you see ^Z.

```
A>copy con c:\autoexec.bat
path=c:\dos
^Z
```

If all goes well, MS-DOS should respond *1 File(s) copied*. If it does not, retype the preceding lines.

Your new version of MS-DOS should now be ready for use. Remove any disk from drive A, and restart the computer by pressing Ctrl, Alt, and Del. When MS-DOS starts up, the

system prompt should be *C*>. You can verify that MS-DOS can find its own command files by typing this:

```
C>chkdsk
```

If MS-DOS produces a report on your hard disk, you're all set. If it does not start from drive C, repeat the setup procedure. If MS-DOS responds *File not found* to your Check Disk command, retype the preceding commands that start with *copy con c:\autoexec.bat*.

Upgrading with Version 3

If you're upgrading from one version of MS-DOS to another, your earlier version might be in a DOS directory, or it might be in the main (root) directory of your hard disk. It all depends on how your hard disk was organized.

Checking on, and possibly reorganizing, your hard disk assumes some knowledge of disks, directories, and directory commands. If this is familiar territory, the following instructions should help you out. If you and MS-DOS are relative strangers and you have a currently functional system, you might prefer to work through the examples in this book or ask someone for advice before trying to organize your hard disk.

If You Have a DOS Directory

If you already have a DOS directory and you are installing MS-DOS version 3.2 or 3.3, use the Replace command to replace all the old MS-DOS files with those in the new version. Change the current drive to C by typing this:

```
A>c:
```

Now, for each new MS-DOS disk you place in drive A, type the following two commands. If necessary, change *dos* to the name of your DOS directory:

```
C>replace a:\*.* c:\dos /s
C>replace a:\*.* c:\dos /a
```

The first command replaces existing files; the second adds any files from the MS-DOS disk that don't exist in the DOS subdirectory on drive C.

You also need to copy COMMAND.COM to the root directory on drive C by typing the following:

```
copy a:command.com c:\
```

That's it. You're done.

If You Don't Have a DOS Directory

If you don't have a DOS directory and all your MS-DOS files are in the root directory of the hard disk, you have two tasks to perform: first, to upgrade your current version of MS-DOS; second, to clear old MS-DOS files out of the root directory.

You've already moved the two new system files to your hard disk. Now upgrade your current version and place your MS-DOS files in a DOS directory; start by creating the new DOS directory:

```
A>md c:\dos
```

Next, copy the file named COMMAND.COM to the root directory of your hard disk:

```
A>copy command.com c:\
```

Now copy the other MS-DOS files from the MS-DOS disk to the DOS directory:

```
A>copy *.* c:\dos
```

If you have another MS-DOS disk, repeat the preceding Copy command. Next, check for an AUTOEXEC.BAT file by typing this:

```
A>type c:\autoexec.bat
```

If MS-DOS displays a file and one of the lines is a Path command like this,

```
PATH=C:\;C:\WP;C:\SPREAD;C:\DB
```

use Edlin or your word processor (if it can save unformatted files) to add your new DOS directory to the Path command:

```
PATH=C:\;C:\DOS;C:\WP;C:\SPREAD;C:\DB
```

If MS-DOS responds *File not found* to your Type command, you need to create a basic AUTOEXEC.BAT file with a Path command as follows (press Ctrl and Z together where you see ^Z):

```
A>copy con c:\autoexec.bat
path=c:\dos
^Z
```

Use the Type command again, this time to check for a file named CONFIG.SYS. If MS-DOS displays a file, use Edlin or your word processor to edit it, adding the name of the DOS directory to all references to MS-DOS files like this:

```
DEVICE=C:\DOS\ANSI.SYS
DEVICE=C:\DOS\VDISK.SYS
```

(If MS-DOS responds *File not found* to this command, you don't have to create a *CONFIG.SYS* file for it to use. You might want to at some point, but for now your system will work without such a file.)

Finally, for the second stage in reorganizing your hard disk, turn on your printer, and print a listing of the root directory with this command:

```
A>dir c:\ sort > prn
```

Using the files in your MS-DOS directory or the files on the disks for your old version of MS-

DOS as a guide, carefully mark the printout, noting all MS-DOS files in the root directory other than COMMAND.COM, AUTOEXEC.BAT, and CONFIG.SYS. That is, mark files such as FORMAT.COM, PRINT.COM, DISKCOPY.COM, and so on.

Once you've marked these file names (and double-checked them to be sure you didn't include any non-DOS files), type a Delete command for each to remove it from the *root* directory. For example, type this:

```
A>del c:\format.com
A>del c:\print.com
A>del c:\diskcopy.com
```

When you're finished, the root directory should contain only COMMAND.COM, AUTOEXEC.BAT, and CONFIG.SYS, as well as the non-DOS files and directories it held before. The remainder of your MS-DOS files should all be in C:\DOS.

PREVIOUS NEXT

Appendix B: Glossary

A

Adapter

A term sometimes used to refer to printed-circuit cards that plug into a computer and control a device, such as a display or a printer.

Anti-Virus

A program in versions 6.0 and later that checks your system for the presence of virus programs. If it finds a virus, it alerts you and, if possible, eliminates the offending file or files that contain the program.

See also Virus.

Application program

A program, such as a word processor or spreadsheet, that performs a specific task; an application of the computer to a particular type of work.

Archive

To transfer files to a separate disk or a backup tape for safekeeping. In versions 6.0 and later of MS-DOS, the MSbackup command helps archive files, and if necessary, it can be used to return archived files to the disk from which they were backed up. (In earlier versions, you use the Backup and Restore commands.)

Argument

See Parameter.

ASCII

A standardized coding scheme that uses numeric values to represent letters, digits, symbols, and so on. ASCII is an acronym for *American Standard Code for Information Interchange* and is widely used in coding information for computers.

AUTOEXEC.BAT

A name reserved for a batch file that contains commands that are carried out by MS-DOS each time the system is started. An AUTOEXEC.BAT file can be used to perform a desired set of startup procedures without your having to type the commands each time.

AUX

Short for *auxiliary.* The communications port MS-DOS uses unless instructed otherwise. AUX can be either COM1 or COM2 in versions of MS-DOS through 3.2; it can be COM1, COM2, COM3, or COM4 in versions 3.3 and later.

B

Backspace key

The key labeled with a single, left-pointing arrow and, often, the word *Backspace*; erases characters to the left of the cursor, one at a time.

Back up

To copy one or more files to disks or to tapes for safekeeping.

BAK

An extension assigned by Edlin and by many word processors to the next-most-recent (penultimate) version of a text file. If the working copy of a file is damaged, the BAK file can be used to salvage a near-current version of the document.

Basic

A programming language included with MS-DOS. When originally developed, Basic was an acronym for *Beginner's All-purpose Symbolic Instruction Code.* Through version 4, MS-DOS includes either Basic and BASICA (Advanced Basic) or GW-BASIC. Versions 5 and later include a more sophisticated, visually oriented form called QBasic.

Batch file

A text file whose extension is BAT; contains MS-DOS commands. When you type the name of a batch file while MS-DOS is at the command level, MS-DOS carries out the commands in the file.

Baud

Broadly, the rate at which data is transmitted over a communications link; more specifically, *baud* refers to the number of changes per second (representing coded data) carried by the signal. For commonly used modem rates, approximately one character per second is transmitted for each 10 baud.

Binary

The base-2 numbering system whose only digits are 0 and 1. The binary system is particularly well suited to use with computers because the two digits can be represented by the presence or absence of a voltage.

Bit

The smallest unit of information used with computers; corresponds to a binary digit (either 0 or 1). Eight bits make up one byte.

Boot

To start up a computer; derived from the saying "pull yourself up by your own bootstraps."

Byte

The unit of measure used for computer memory and data storage. One byte contains eight bits and can store one character (a letter, number, punctuation mark, or other symbol).

PREVIOUS NEXT

C

Character string

A group of characters that you tell MS-DOS to treat as a set of letters or numbers rather than as a command. The Find filter command searches for character strings enclosed in quotation marks (""). In other commands, such as Search and Replace, the quotation marks are not needed.

Chip

See Integrated circuit.

Color Graphics Adapter (CGA)

A printed-circuit card in the system unit of a computer that controls the display; processes both text and graphics at low resolution in up to 16 colors.

COM1, COM2, COM3, COM4

Short for *communications.* The names of the serial communications ports. All versions of MS-DOS recognize COM1 and COM2; versions 3.3 and later also recognize COM3 and COM4.

Command

An instruction you give to control a computer program, such as MS-DOS or an application program.

Command file

A file that contains the program or instructions required to carry out a command. If the file's extension is COM or EXE, the command file contains machine instructions; if its extension is BAT, the command file is a batch file and contains MS-DOS commands.

Communications

The transmission of data between computers; also called telecommunications.

Communications port

See Port.

Compression

See File compression.

CON

Short for *console.* The name by which MS-DOS refers to the keyboard (input) and the display (output).

Control key

The key labeled Ctrl; use it as you do the Shift key by holding it down while pressing another key. The Ctrl key in combination with another key can prompt an action or create a character with special significance. For example, when the Ctrl key is pressed with Break, any command that was being executed is interrupted. The Ctrl key pressed with Z creates a special character called the end-of-file marker. If displayed on the screen, Ctrl is shown as ^, as in the end-of-file marker ^Z (Ctrl-Z).

Conventional memory

The name used for the first megabyte of random access memory in an IBM or compatible computer. MS-DOS and application programs can freely use the first 640 KB of conventional memory. Portions of the memory between 640 KB and 1 MB can be used by versions 5 and later of MS-DOS.

CPU

An abbreviation for *central processing unit.* The part of a computer that performs calculations and processes information. In microcomputers that use MS-DOS, the CPU is an 8086/8088, 80186, 80286, 80386, 80486, or a Pentium microprocessor.

Ctrl

See Control key.

Ctrl-Break

The key combination that cancels a command; entered by holding down the Ctrl key and pressing the Break key.

Ctrl-C

Same as Ctrl-Break.

Ctrl-Num Lock

The key combination that stops MS-DOS until you press any other key. On keyboards without a Pause key, Ctrl-Num Lock is used to freeze the display so you can view long displays. Entered by holding down the Ctrl key and pressing the Num Lock key.

Ctrl-P

Same as Ctrl-PrtSc.

Ctrl-PrtSc

The key combination that controls simultaneous printing and displaying. Pressing Ctrl-PrtSc once causes MS-DOS to print everything that is displayed; pressing Ctrl-PrtSc again causes MS-DOS to stop printing everything that is displayed. Entered by holding down the Ctrl key and pressing the PrtSc key.

Ctrl-S

Same as Ctrl-Num Lock.

Ctrl-Z

The key combination that creates the special character (displayed as ^Z) that MS-DOS uses to mark the end of a file. Created by holding down the Ctrl key and pressing Z or by pressing the function key labeled F6.

Current directory

The directory in which MS-DOS looks for and writes files unless otherwise instructed.

Current drive

The drive containing the disk on which MS-DOS looks for and saves files unless otherwise instructed.

PREVIOUS NEXT

D

Data

The information available for processing by a computer in doing its work.

Data bit

A signal used in serial communications to represent part of a character. Seven or eight data bits can be used to represent one transmitted character.

Data file

A file that contains the data needed by a program; the data can be numbers, text, graphics images, sound clips, video clips, or a combination.

Defragment

The process of rearranging the files on a disk so that all available space is combined into a single area.

See also Fragment.

Device

A piece of computer equipment, such as a display or a printer, that performs a specific task. The program that controls a device is called a device driver.

Device name

The name by which MS-DOS refers to a device (for example, PRN, LPT1, LPT2, or LPT3 for a printer). Device names are treated like file names by MS-DOS.

Directory

The index of files that MS-DOS maintains on a disk. The directory entry for each file includes the file's name, extension, size, date and time it was created or last changed, attributes, and the location of the beginning of the file. All but the file attributes and the beginning location can be displayed by the Directory command, though file entries can be displayed based on attribute type.

Disk

A magnetically coated disk used to store information. The term is used when no distinction need be made between a floppy disk and a hard disk.

Disk cache

A portion of memory set aside for use as a temporary, rapid-access storage area for information read from disk. MS-DOS versions 6.0 and later also allow this area to be used for caching information to be *written* to a disk. A disk cache speeds operations by cutting down on the number of times the computer must perform relatively slow disk operations.

Disk drive

The device that rotates a disk in order to read (retrieve) and write (store) information.

Diskette

See Floppy disk.

Display

The screen on which the computer shows both what you type at the keyboard and the result of its work; assumed by MS-DOS to be the standard output device unless a different device is specified.

Doskey

A program in versions 5 and later of MS-DOS that records commands and enables you to repeat, edit, or store them as batch files or keyboard macros.

DoubleSpace

A program in versions 6.0 and 6.2 that compresses the files on your hard disk, increasing the effective capacity of your hard disk by up to 100 percent.

Drive letter

The letter that identifies a disk drive; can be any letter from A to Z.

DriveSpace

A program in version 6.22 that compresses the files on your hard disk, increasing the effective capacity of your hard disk by up to 100 percent. DriveSpace is the replacement for the DoubleSpace program offered in version 6.0 and 6.2.

PREVIOUS NEXT

E

Edit

As a verb, to change the contents of a file, usually with a word processor or an editing program. As a noun, Edit is the name of the command that starts the MS-DOS Editor, the menu-based text editor included with versions 5 and later of MS-DOS.

Editor

A program used to create or change text files; also called a *text editor.*

Edlin

One of the two MS-DOS text editors. (The other, new with version 5, is the MS-DOS Editor.) Edlin numbers the lines in a file and uses those numbers as references in finding, changing, adding, or deleting lines.

See also Edit.

Electronic disk

See RAM disk.

Enhanced Graphics Adapter (EGA)

A printed-circuit card in the system unit of a computer that controls the display. Processes both text and graphics at medium resolution in up to 64 colors.

Enter key

The key you press to tell MS-DOS that you have finished typing a line. Labeled *Return* on some keyboards.

Escape key

The key labeled Esc; cancels a line you have typed but have not yet entered by pressing the Enter key.

Expanded memory

Additional memory installable on any early PC-compatible computer. Use of this memory requires a special program called an expanded memory manager, and programs must be written specifically for such memory. Standardized use of expanded memory is governed by the Lotus-Intel-Microsoft (LIM) specification.

Extended memory

Memory above 1 MB that can be installed on an IBM PC/AT, IBM PS/2 Models 50 or above, or any computer compatible with those models that has an 80286, 80386, 80486, or a Pentium microprocessor. Extended memory is typically handled by a

program called an extended memory manager.

Extension

A suffix of up to three characters that can be added to a file name to identify the contents of the file more precisely.

PREVIOUS NEXT

F

File

A named collection of information stored on a disk; usually contains data, graphics, or a program.

File compression

The technique of reducing the amount of space a file requires on a disk or tape by replacing common character combinations with shorter codes. If you compress all files on a disk or tape, the effect is to increase the apparent capacity, often by a factor of two or more.

File name

A name of up to eight characters that you assign and that MS-DOS uses to find a file on a disk; can be followed by a period and three additional characters called the file-name extension.

Filespec

The complete specification of a file; can include a drive letter, path name, file name, and extension.

Filter command

An MS-DOS command that reads standard input, processes it some way (for example, sorts it alphabetically), and writes the result to standard output.

Fixed disk

See Hard disk.

Floppy disk

A removable disk for storing files, made of thin plastic and enclosed in a protective jacket.

Floppy disk drive

A disk drive used for floppy disks.

Format

To prepare a disk for use.

Fragment

The division of the existing files on a disk into discontiguous segments, generally the result of enlarging portions of those files. This results in breaking the available storage space on a disk into smaller and smaller pieces. Such a condition slows down MS-DOS file operations. A fragmented disk can be corrected by moving files

into the unused areas until all available storage space is combined into a single, contiguous area.

See also Defragment.

Function key

One of 10 or more keys—usually labeled F1, F2, and so on—that cause MS-DOS (or an application program) to perform a certain application-specific function, such as copying characters in a line of text.

PREVIOUS NEXT

G

Gigabyte

A value equal to 1024 × 1024 × 1024, or 1,073,741,824 bytes.

PREVIOUS NEXT

H

Hard disk

A disk of large capacity (10 MB or more) that cannot be removed from its drive. Also called a *fixed disk.*

Hardware

The equipment that makes up a computer system, as opposed to the programs, or software.

Hexadecimal

The base-16 numbering system whose digits are 0 through 9 and A through F (the letters A through F correspond to the decimal numbers 10 through 15); often used in computer programming because it is easily converted to and from binary, the base-2 numbering system the computer uses.

Hidden file

A file that is not normally listed when you display the directory. MS-DOS uses two special, hidden files on any startup disk. They are hidden so that they cannot be altered or deleted under normal circumstances. Beginning with version 5, directories and data files can also be hidden for a certain amount of privacy. MS-DOS versions 6.0, 6.2, and 6.22 also add hidden files for supporting file compression.

Hierarchical filing system

See Multilevel filing system.

High Memory Area (HMA)

The name given to the first 64 KB of extended memory (additional memory starting at 1 MB). On a computer with extended memory, versions 5 and later of MS-DOS can run in the HMA, leaving more conventional memory available for applications and data.

PREVIOUS NEXT

I

Initialize

See Format.

Input

The data that a program reads.

Input/output

A term that refers to the devices and processes involved in the computer's reading (input) and writing (output) of data.

Integrated circuit

An electronic device that has thousands of transistors on a wafer of silicon. Such devices are the building blocks of computers. Also called a *chip.*

Interface

The boundary between two systems or entities, such as a disk drive and the computer, or the user and a program.

I/O

Abbreviation for *input/output.*

I/O redirection

See Redirection.

K

Keyboard

The device consisting of alphabetic keys and other keys on which instructions and data are typed into the computer; assumed by MS-DOS to be the standard input device unless a different device is specified.

Kilobyte

A value equal to 1024 bytes.

PREVIOUS NEXT

L

LPT1, LPT2, LPT3

Short for *line printer.* The names that MS-DOS uses to refer to the three ports to which parallel printers can be attached.

PREVIOUS NEXT

M

Macro

A set of keystrokes or commands that you assign a name and store either temporarily in memory or permanently on disk. A macro is a means of saving time by assigning a short name to a long or involved set of commands you use frequently. You can create macros with the Doskey program in versions 5 and later of MS-DOS.

Megabyte

A value equal to 1024 × 1024, or 1,048,576 bytes.

Memory

A type of electronic circuitry that the computer uses to store programs and data. Unlike disk storage, which is permanent, a computer's working memory is temporary—its contents are lost when power is removed. Memory is usually measured in units of 1024 bytes, called kilobytes and abbreviated K or KB; a megabyte (M or MB) is equal to 1024 kilobytes, or 1,048,576 bytes, and a gigabyte (G or GB) is equal to 1024 megabytes, or 1,073,741,824 bytes.

Microcomputer

A small computer system whose central processing unit is a microprocessor; usually used by only one person.

Microprocessor

An integrated circuit, or chip, that contains the circuits needed to carry out program instructions. The microprocessor performs calculations, briefly stores instructions and data, and transfers information to and from a computer's memory. In the world of MS-DOS, this includes the 8086 through Pentium processors, and their clones.

Modem

Contraction of *modulator-demodulator.* A device that enables transmission of computer data over telephone lines.

Monitor

A television-like device that displays computer input and output; often used synonymously with *display.*

Monochrome

A term used to describe a computer display capable of displaying one color (usually white, green, or amber).

Monochrome Display Adapter (MDA)

A printed-circuit card, in the system unit of a computer, that controls a monochrome

display. Processes text only, not graphics, at medium resolution in one color.

Multicolor Graphics Array (MCGA)

A printed-circuit card, in the system unit of a computer, that controls the display. Processes both text and graphics at low to medium resolution in up to 256 colors; used in some early low-end IBM PS/2 model computers.

Multilevel filing system

A computer filing system that lets you define directories within other directories, creating a structure with many levels. Also called a *tree-structured* or *hierarchical filing system.*

PREVIOUS NEXT

N

Network

A group of computers that are linked by printed-circuit cards, cables, and network software and can share resources, such as programs, data, disk drives, and printers.

PREVIOUS NEXT

O

Operating system

A program that coordinates the operation of all parts of a computer system. MS-DOS is an operating system.

Option

See Parameter.

Output

The result of a program's processing of input data.

PREVIOUS NEXT

P

Parallel communications

A communications technique that uses multiple wires to send all eight bits of a byte at once (in parallel).

Parallel port

A port for parallel communications; the port to which the printer is usually attached.

Parameter

A qualifier that you include with a command to define more specifically what you want MS-DOS to do; also called an *argument* or an *option.*

Parity

An error-detection technique that is used to ensure accuracy during data communication. Some microcomputers also use parity to ensure the integrity of data in RAM.

Path

The list of directory names that defines the location of a directory or file.

Path name

The portion of a file specification that defines the path to the file; can include a drive letter followed by a colon.

Pipe

To direct the output of one command for use as the input of another command. The pipe symbol MS-DOS uses is the broken vertical bar (œ).

Port

The electrical connection through which the computer sends and receives data to and from devices or other computers.

Printed-circuit card

A thin, rectangular card or board, usually made of fiberglass or epoxy and coated with copper. Electrical circuits and connections are etched into the copper, and electronic devices, such as integrated circuits, are soldered to the circuits. These cards are at the heart of a computer system, giving the machine its ability to perform calculations, store and transfer data, and use the display, disk drives, mouse, printer, modem, and other devices.

Printer

A device that produces images of text and graphics on paper.

Print queue

The list of files to be printed; you can create, examine, and modify the print queue with the Print command.

PRN

Short for *printer.* The printer MS-DOS uses unless instructed otherwise. Can refer to LPT1, LPT2, or LPT3.

Program

A set of instructions for a computer.

Prompt

A request displayed by the computer for you to provide some information or perform an action.

PREVIOUS NEXT

Q

Queue

See Print queue.

PREVIOUS NEXT

R

RAM

Short for *random access memory.* The memory that MS-DOS uses for programs and data; RAM content changes often while you use the computer and is lost when the computer is turned off.

RAM disk

A portion of the computer's random access memory reserved for use as a simulated disk drive. Also called an *electronic* or *virtual* disk. Unless saved on a physical disk, the contents of a RAM disk are lost when the computer is turned off.

Read-only file

A file whose read-only attribute is set so that its contents can be displayed and read but not changed or deleted.

Redirection

The process of making a command or program take its input from a file or device other than the keyboard (standard input), or of causing output of a command or program to be sent to a file or device other than the display (standard output). The MS-DOS redirection symbols are the greater-than (>) and less-than (<) signs.

Replaceable parameter

A symbolic reference, consisting of a percent sign followed by a one-digit number (such as %1), that can be included with commands in a batch file to refer to the parameters entered with the batch command. In versions 5 and later of MS-DOS, replaceable parameters can also be used with Doskey; they are represented by a dollar sign followed by a one-digit number (such as $1).

Return key

The Enter key.

ROM

Short for *read-only memory.* A type of computer memory that is permanently recorded in hardware. ROM contains instructions that help a computer carry out routine tasks, such as starting itself. The contents of ROM cannot be changed and are not lost when the computer is turned off. On some computers, such as some models of the IBM PS/1 series, MS-DOS is contained in ROM.

Root directory

The main directory that MS-DOS creates on each disk; the top directory in a multilevel filing system.

S

ScanDisk

A program in versions 6.2 and later that checks both the structure of the files on a disk and the surface of the disk itself for errors. In many cases, it can correct the structure errors (recovering lost data) or move files from areas where the disk surface is damaged to undamaged locations.

Serial communications

A communications technique that transfers information one bit at a time (serially) rather than one byte at a time (in parallel); used in modem communications and with some printers and other devices, such as mice.

Serial port

The communications port (COM1, COM2, COM3, or COM4) to which a device, such as a modem or a serial printer, can be attached.

Shell

A program that shows itself to the person using the computer and then passes commands to a different program to be carried out. It's called a shell because it effectively surrounds the other program, hiding it from view. Versions 4 through 6.0 of MS-DOS include the Dosshell program that lets you use many MS-DOS commands without having to type commands or file names at the system prompt. Other MS-DOS shell programs are also available.

Software

The programs, as opposed to the equipment or hardware, that are used with a computer system.

Standard input

The device from which a program reads its input unless the input is redirected; in normal MS-DOS operation, standard input is the keyboard.

Standard output

The device to which a program sends its output unless the output is redirected; in normal MS-DOS operation, standard output is the display.

Stop bit

A signal used in serial communications that marks the end of a character.

Subdirectory

A file that contains directory entries; sometimes also used to refer to the group of files whose directory entries are in the same file.

Super VGA (SVGA)

A printed-circuit board, in the system unit of a computer, that controls a high-resolution display. Processes both text and graphics, and is capable of millions of colors.

Switch

A term used by MS-DOS to identify those parameters which are preceded by a forward slash, such as */w* when used with the Directory command.

System program

A program whose purpose is to control the operation of all or part of the computer system, such as managing the printer or interpreting commands. Generally, system programs are components of an operating system.

System prompt

The characters MS-DOS displays when it is at the command level (ready to accept a command). Unless you specify otherwise, the system prompt usually shows the current drive and directory followed by a greater-than sign (for example, C:\DOS>).

System unit

That part of a microcomputer which contains the microprocessor, power supply, disk drives, RAM, and adapter cards, and provides for the attachment of other input and output devices.

PREVIOUS NEXT

T

Telecommunications

See Communications.

Temporary file

A file that MS-DOS or an application program creates for holding interim data. MS-DOS, for example, creates temporary files when redirecting input or output. Temporary files are deleted by the program when they are no longer needed.

Text

Ordinary, readable characters, including the uppercase and lowercase letters of the alphabet, the numerals 0 through 9, and punctuation marks.

Text editor

A program that you use to create or change text files. Also called simply an *editor.*

Text file

A file that you can read (contains ordinary letters, numerals, and punctuation marks).

Tree-structured filing system

See Multilevel filing system.

PREVIOUS NEXT

U

Update

To change a file, creating a new (or updated) version.

Upper memory area

The area of memory between 640 KB and 1 MB, portions of which can be used by versions 5 and later of MS-DOS.

Upper Memory Block (UMB)

A portion of the upper memory area that can be allocated for use by MS-DOS.

PREVIOUS NEXT

V

Video Graphics Array (VGA)

A printed-circuit card, in the system unit of a computer, that controls the display. Processes both text and graphics at medium to high resolution in up to 256 colors.

Virtual disk

See RAM disk.

Virus

A program that hides itself on a disk or inside a file and can spread itself from system to system. Virus programs eventually do something irritating or even destructive, such as destroying files or formatting a disk.

Volume label

An identifying name of up to 11 characters that you can assign to a disk.

PREVIOUS NEXT

W

Wildcard character

A special character that, like the wild card in a poker game, can be used to represent any other character. MS-DOS recognizes two wildcard characters: the question mark (?), which can represent any single character, and the asterisk (*), which can represent any number of characters.

Write-protect

To cover the small notch on a 5.25-inch floppy disk or to uncover the opening on a 3.5-inch floppy disk so that new or changed information cannot be written onto the floppy disk.

PREVIOUS NEXT

Appendix C: MS-DOS Command Reference

<Command> /?

Command Help

See Help (versions 5 and later).

PREVIOUS NEXT

Index

Special Characters

$ (Doskey replaceable parameter), 331, 474–75
$ (Prompt operator), 395–96, 474, 553–54
$T (Doskey command separator), 332
% (batch replaceable parameter), 320–21
% (Dosshell parameter), 270
%% (batch replaceable parameter), 364–66, 504–5
* (Dosshell directory operator), 261
* (wildcard character), 45, 69–70, 344–45
+ (Copy file-combining operator), 82
+ (Dosshell directory operator), 244, 261
- (Dosshell directory operator), 244, 261
. (current directory marker), 145, 153–58
. (date separator), 20
.. (parent directory marker), 145, 153–58
... (Dosshell symbol), 249
/ (date separator), 20
/? (command help parameter), 73, 512
: (batch file label operator), 342
< (redirection operator), 52, 300, 515
> (redirection operator), 54, 299, 310, 515
>> (redirection operator), 308, 515
? (wildcard character), 45, 70–71, 85–86
@ (batch display-suppression operator), 339, 435
\ (canceled command symbol), 31
\ (path name separator), 149
\ (root directory marker), 145, 149
^ (control key symbol), 33
œ (piping operator), 53, 308–10, 515–16

Index

A

accented characters, 137–38
access, file and directory. *See* attributes, file and directory
access, speeding disk. *See* disk buffers; disk caches; Fastopen command
adapter RAM/ROM memory, 122
Advanced Power Management (APM), 390, 550
allocation units, 47, 104
Alt-255 keys, 347, 357
Alternate key (Alt), 32, 246
Alt-F1 keys, 284
Alt-F7 keys, 332
Alt-Minus keys, 284
Alt-Plus keys, 284
Alt-Tab keys, 264
ampersand (@) as batch display-suppression operator, 339, 435
ANSI.SYS device driver
  command reference, 428
  installing, 390, 463
  Mode and, 125, 332–33, 358
anti-virus. *See* virus protection
APM (Advanced Power Management), 390, 550
Append command, 171–72, 428–29
application programs, 6, 175–78, 222–23. *See also* program files
archive attributes, 178, 179–81, 186–90
archiving. *See* backing up files and directories
arrow keys
  activating, 31–32
  Doskey use of, 20, 37–38
  Dosshell use of, 250–55
  Edit use of, 280
  Up Arrow, 17, 18, 20, 37
ASCII (American Standard Code for Information Interchange), 90, 253
Assign command, 430

associating files with programs, 251

asterisk (*), 45, 69–70, 261, 344–45

Attribute command (attrib), 178–83, 187, 431–32

attributes, file and directory
  archive status, 179–81
  changing, 178–83, 431–32
  hidden status, 181–83
  read-only status, 84, 183
  Xcopy and, 187

AUTOEXEC.400 file, 377, 406

AUTOEXEC.BAT file
  batch file commands and, 435
  CONFIG.SYS and, 376–77, 392
  creating, 323–27
  disk caches, 383–84
  memory management (*see* memory)
  MS-DOS installation, 377, 406
  paths, 165–66, 408, 410
  root directory requirement, 202
  starting MS-DOS without, 392

AUX device name, 120

Auxiliary devices. *See* serial ports

Index

B

backing up files and directories. *See also* restoring files and directories
  with Backup, 232–36, 432–34
  with batch files, 336–50, 362–63, 366
  developing procedures, 224
  duplicating floppies, 100–101
  with MSbackup (*see* MSbackup command)
  MS-DOS versions and, 233, 434

backslash character (\), 31, 145, 147

Backspace key, 18, 20–21

BACKUP.001 file, 234–35

Backup command, 232–36, 432–34. *See also* Restore command

BACKUPID.@@@ file, 235

BACKUP.LOG file, 233

BAS extension, 67

Basic language, 262, 555

batch files, 316. *See also* files
  archiving files with, 336–37, 362–63, 366
  calling other, 366–68, 438–39
  canceling, 322–23
  chaining, 354–57, 366–67
  changing command sequences in, 342–46, 508
  command reference, 435–36
  conditional execution, 340–42, 514–15
  copying files between directories with, 360–61
  creating, with Copy, 318–19
  creating, with Doskey, 329–34 (*see also* Doskey command)
  creating, with Edit, 279 (*see also* Edit command)
  creating, with Edlin, 489–95
  deleting files with, 328–29, 358–59
  displaying directories of subdirectories with, 359–60
  displaying line spaces with Echo, 347–50, 357
  displaying long directories with, 357–58
  displaying messages with Echo, 338–39, 486–87
  displaying messages with Pause, 339, 549
  displaying messages with Remark, 319–20, 558
  displaying multiple files with, 370–71
  displaying sorted directories with, 372–74

interactive, 368–70, 443–44
labels, 342–43
macros vs., 329
MS-DOS command search order and, 316–17
printing files with, 327–28
repeating commands with For command, 363–66, 504–5
repeating commands with replaceable parameters, 320–21
searching through files with, 352–57, 371–72
shifting command parameters, 362–63, 569
starting MS-DOS with (*see* AUTOEXEC.BAT file)
wildcard characters and, 344–45

BAT extension, 67, 317

baud rate, 128–29

blank lines, displaying, with batch files, 347–50, 357

booting MS-DOS. *See* starting MS-DOS

Break command, 436. *See also* Ctrl-Break keys

Break configuration command, 437. *See also* Ctrl-Break keys

Break key, 32

Buffers configuration command, 391, 437–38

built-in commands, 316

bytes, 26–27

PREVIOUS NEXT

Index

C

caches, disk, 383–84, 570–73

Call batch command, 366–68, 438–39

capacities of disks, 7, 27, 103, 105, 106, 174, 506

case of characters, 353, 372

cd. *See* Change Directory command (cd, chdir)

CD-ROM drives, 546

chaining batch files, 354–57, 366–67

chains, 442

Change Code Page command (chcp), 439–40

Change Directory command (cd, chdir), 27, 146–47, 153–58, 440–41

characters. *See also* keys
- accented, 137–38
- ASCII, 90, 253
- case of, 353, 372
- end-of-file, 48, 133
- foreign-language (*see* foreign languages)
- graphics (*see* graphics)
- grayed, 249
- hexadecimal, 253
- replacing, with Edit, 292
- searching text for, 54–55, 304–7, 352–57, 371–72, 503–4
- system prompt definition, 395, 554
- wildcard, 45, 69–71, 344–45

character strings, 54

chcp. *See* Change Code Page command (chcp)

Check Disk command (chkdsk)
- checking disk condition, 111–14
- command reference, 441–43
- displaying directory structure, 166–70, 579
- DoubleSpace and, 442
- ScanDisk and, 112, 113, 441 (*see also* ScanDisk command)
- Windows and, 168, 441

Choice batch command, 368–70, 435, 443–44

Clear Screen command (cls), 23, 123, 348, 445

client-server connections, 517–18

Clipboard, Edit, 290–91, 293–94

clock, system. *See* date; time

CLOCK$ device name, 66

cls. *See* Clear Screen command (cls)

clusters, 442

code pages. *See also* COUNTRY.TXT file; foreign languages
  changing, 439–40
  date, time, currency, and decimal formats, 454–58, 547–48
  displaying status of, 131–32, 543
  enabling graphics characters, 508–9
  keyboard layouts, 135–39, 520–23
  preparing, 541–42
  restoring, 543
  selecting, 542

colon (:), 342

color options
  Dosshell, 248
  Edit, 294–95
  Startup Menu, 529–30

COM1-COM4 device names, 120. *See also* serial ports

combining text files, 82–83, 453–54

COM extension, 67, 316

COM files, converting EXE files to, 497

COMMAND.COM file, 202, 239, 408, 445–46, 497, 568

Command command, 445–46

command files, 42–43. *See also* batch files; commands, MS-DOS; Doskey command; program files

command history. *See* Doskey command

command level, 16

command macros. *See* Doskey command

command processor. *See* COMMAND.COM file

command prompt. *See* system prompt

commands, MS-DOS, 9–10, 42–43. *See also* Dosshell command
  canceling, 31, 32, 34
  configuration (*see* CONFIG.SYS configuration file) entering, 16–18
  external vs. internal, 42–43
  file management, 65 (*see also* files)
  filter (*see* filter commands)

help on, 71–73
parameters, 28, 44
redirecting input and output of (*see* input and output redirection)
search order, 316–17
search paths, 165–66, 548–49 (*see also* paths)
sorting ouput of, 52–53, 573–74
user-defined (*see* batch files; Doskey command)
wildcard characters, 45, 69–71, 344–45

communications parameters, 128–30, 537–38

communications ports. *See* serial ports

Compare command (comp), 88, 446–48

comparing disks, 110–11, 469–70

comparing files
with Compare, 88, 446–48
with File Comparison, 88–91, 110, 499–501

compatibility, 9. *See also* versions, MS-DOS

compressed files, expanding MS-DOS, 498

compression, disk
with DoubleSpace, 476–81
with DriveSpace, 190–97, 481–86
of floppy disks, 193–94
of hard disks, 191–93
memory and, 191, 480–81, 486
mounting and unmounting drives for, 194–96
MS-DOS versions and, 191
removing, 196–97
ScanDisk and, 192

computers
configuring portable, 390–91, 517–18, 549–50
displaying information about, 546–47

condensed print, 125

CON device name, 48, 66, 119–20

CONFIG.400 file, 377, 406

CONFIG.SYS configuration file, 376–91
AUTOEXEC.BAT and, 376–77, 392
checking for Ctrl-C (Ctrl-Break), 437
code-page switching (*see* code pages)
command reference, 448–49
console control, 125, 390, 428
date and currency formats, 454–58, 547–48
defining characteristics of disks or tapes, 481

device drivers (*see* device drivers)
disk buffers, 391, 437–38
disk caches, 383–84, 572–73
disk compression, 191, 480–81, 486
emulating earlier versions of MS-DOS, 567–68
highest drive letter, 391–92, 524
including other configuration blocks, 515
installing fast file access, 396–97, 499
loading device drivers, 377, 388–90, 462–66
loading memory-resident programs, 516
loading MS-DOS in high memory, 380–81, 472–73
memory management (*see* memory)
menus, 393–94, 529–31, 574
MS-DOS installation and, 377, 406
MS-DOS options, 576
Num Lock key state, 548
open files, 391, 501, 503
portable computer settings, 390–91, 518, 550
RAM disks, 384–87, 555–57, 585
reserving memory for temporary program use, 574
root directory requirement, 202
specifying command processor, 568
starting MS-DOS without, 392

confirmation options, Dosshell, 254–55

consoles. *See also* keyboards; screens
changing, 458
control program (*see* ANSLSYS device driver)
creating batch files from, 318–19
creating text files from, 48–49
device name, 48, 66, 119–20
displaying status of, 131–32

CONTROL.001file, 234–35

Control key (Ctrl), 31, 33–35, 251

conventional memory, 122, 378

Copy command
combining text files, 82–83, 453–54
command reference, 450–54
copying files, 51, 78–81, 177–78
copying files between directories, 148–49
copying from devices to devices or text files, 133, 453
copying text files to devices, 50, 81, 452
creating batch files, 318–19
creating text files, 48–49, 67–68

installing MS-DOS files, 408–9, 410–11
copying files. *See also* moving files
with Copy (*see* Copy command)
with Dosshell, 251–52
with Replace, 81, 184–86, 409, 560–61
with Xcopy, 81, 176–77, 186–90, 588–89
copying floppy disks, 108–10
country codes. *See also* foreign languages
code pages, 439–40
date, time, currency, and decimal formats, 454–58, 547–48
keyboard layouts, 135–39, 520–23
system floppy disks, 564–65
Country configuration command, 454–58
COUNTRY.SYS file, 458, 547–48
COUNTRY.TXT file, 137, 439
Ctrl-* keys, 261
Ctrl-Alt-Del keys, 33, 38
Ctrl-Alt-F1 keys, 139
Ctrl-Alt-F2 keys, 139
Ctrl-Break keys, 33, 34, 322–23, 436–37
Ctrl-Esc keys, 263–64
Ctrl-F1 keys, 284
Ctrl-Ins keys, 290
Ctrl key, 31, 33–35, 251
Ctrl-Num Lock keys, 33
Ctrl-P keys, 35
Ctrl-PrtSc keys, 33, 33, 34–35
Ctrl-T keys, 313
Ctrl-Z keys, 48, 133
CTTY command, 458
currency formats, 454–58, 547–48
current directory (.), 149
changing, 146–47, 440–41
marker, 145
as system prompt, 146, 153, 175
current disk drive, 16, 22
cursor, 18, 279, 280

cursor-movement keys. *See* arrow keys
cut and paste, 290–91

Index

D

data bits, 128–29

data files, 42, 251

data file search paths, 171–72, 428–29

date
  changing, 19–21, 459
  formats, 454–58, 547–48
  starting MS-DOS with, 15–16

Date command, 19–21, 72–73, 459

dblspace. *See* DoubleSpace command (dblspace)

DBLSPACE.SYS device driver, 191, 480–81

dead keys, 137–38

decimal formats, 454–58, 547–48

Defrag command, 174, 199–201, 397, 459–61

defragmentation, 199–201, 459–61

Delete command (del, erase), 51–52, 83–86, 461–62

Delete key (Del), 32

Delete Sentry, 208–11, 581

Delete Tracker, 208, 211–15, 581

deleting directories, 160–63, 462, 559

deleting files. *See also* undeleting files
  with batch files, 328–29, 358–59
  with Delete, 51–52, 83–86, 461–62
  with DelTree, 162–63, 462
  with Dosshell, 254–55
  wildcard characters and, 71

DelTree command, 162–63, 462

Device configuration command, 377, 462–65

device drivers. *See also* CONFIG.SYS configuration file
  loading, 377, 462–65
  loading, into reserved memory, 388–90, 465–66

Devicehigh configuration command, 380, 382, 385, 387–90, 465–66

devices, 48, 119. *See also* consoles; disks; display adapters; keyboards; memory; parallel ports; printers; screens; serial ports

code-page switching (*see* code pages)
controlling modes of (*see* Mode command)
copying files to, 50, 81, 452
copying from, to files or devices, 48–49, 67–68, 133, 318–19, 453
displaying status of, 131–32, 540–41
drivers (*see* device drivers)
names, 66, 119–20
on networks, 120, 390–91, 517–18
redirection of (*see* input and output redirection)

diagnostics, 546–47

dialog boxes, 252, 271

differential backups, 227

dir. *See* Directory command (dir)

direction keys. *See* arrow keys

directories, 26–28. *See also* files; tree-structured file systems
backing up (*see* backing up files and directories)
controlling access (*see* attributes, file and directory)
copying files between (*see* copying files)
creating, 142–45, 150–53, 526–27
current (*see* current directory (.))
displaying (*see* Directory command (dir); File List, Dosshell)
displaying information about, 256–57
displaying structure of, 166–70, 578–80
joining disks to, 519
markers, 145, 153–58
moving files between (*see* moving files)
for MS-DOS files, 27–28, 175, 201–2, 408–11
naming, as disks, 397–98, 574–75
path names (*see* paths)
removing, 160–62, 559
removing, with Dosshell, 254
removing, with files, 162–63, 462
renaming, 159–60, 544–45
restoring (*see* restoring files and directories)
root (*see* root directory)
scrolling and selecting, with Dosshell, 250–55
speeding access (*see* disk buffers; disk caches; Fastopen command)
subdirectories, 27–28, 142, 149
using, 147–49

Directory command (dir)
command reference, 466–69
directory-only subdirectories, 359–60
displaying long directories, 53–54, 357–58

displaying specific files or sets of files, 44–45, 54–55, 69–70
file-only displays, 77
multiple-directory displays, 163–65
parameters, 74–75
pausing displays, 28–29, 53–54, 76
printing displays, 34–35, 54
sorting displays, 76–77
sorting displays with batch files, 372–74
wide displays, 75

disk access speedup. *See* disk buffers; disk caches; Fastopen command

disk buffers, 391, 437–38

disk caches, 383–84, 570–73

Disk Compare command (diskcomp), 110–11, 469–70

Diskcopy command, 108–10, 471–72

disk drives. *See* disks

diskettes. *See* floppy disks

disk files. *See* files

disk operating systems. *See* Microsoft MS-DOS; Microsoft Windows

disks. *See also* floppy disks; hard disks
changing current, 22
checking, for errors, 192, 197–98, 563–64
compressing (*see* compression, disk)
defining characteristics of, 481
defining highest letter for, 391–92, 524
directory structure (*see* directories; files; tree-structured file systems)
displaying information about, 256–57
displaying status of, 111–14, 168–70, 441–43, 579
file names and, 43, 67
formatting (*see* Format command; Select command)
installing or upgrading MS-DOS (*see* versions, MS-DOS)
joining, to directories, 519
naming directories as, 397–98, 574–75
RAM or virtual, 384–87, 555–57, 585
recovering files from damaged, 557–58
reformatting, 215–18
routing operations to different, 430
saving recovery information, 212, 216, 531–33, 583–85
speeding access (*see* disk buffers; disk caches; Fastopen command)
system (*see* system disks)
system prompt as current, 16
types and capacities, 7, 27, 105, 106, 174, 506
unformatting, 106, 217–18, 583–85

verifying data written to, 585–86
virus protection (*see* virus protection)
volume labels (*see* labels, disk volume)
volume serial numbers, 47, 104, 115

Disk Utilities Program Group, Dosshell, 262, 272–74

display adapters. *See also* consoles; screens
changing display characteristics, 124–25, 534–35
changing display columns or lines, 535–36
Dosshell and, 248
memory, 122

dollar sign ($), 331, 395–96, 474–75, 553–54

DOS. *See* Microsoft MS-DOS

Dos configuration command, 380–81, 387, 472–73

Doshelp command, 512. *See also* Help command

Doskey command
canceling commands, 31
command parameters, 330
command reference, 473–75
creating batch files, 330–31 (*see also* batch files)
creating macros, 331–33
editing commands, 310–14
macros vs. batch files, 329
repeating commands, 20, 35–38
replaceable parameters, 331–33
saving macros, 333–34
starting, 18
using multiple commands, 313–14

Dosshell command, 9
color options, 248
command reference, 475–76
File List (*see* File List, Dosshell)
File menu, 249, 265–68
Help menu, 246–47
keyboard and mouse use in, 245–46
Options menu, 254–57, 263–64
Program List (*see* Program List, Dosshell)
starting, 242–46
starting MS-DOS with and without, 14–15
text mode vs. graphics mode, 243
Tree menu, 261
View menu, 257–61
window, parts of, 243–45

Windows and, 242
DOSSHELL.INI file, 253
DOS subdirectory, 27–28, 175, 201–2, 408–11
dot-matrix printers. *See* printers
DoubleSpace command (dblspace)
Check Disk and, 442
command reference, 476–81
DriveSpace and, 174, 191 (*see also* DriveSpace command (drvspace))
formatting disks, 505
memory and, 191, 480–81
Smartdrv and, 383
Drive Parameters configuration command (drivparm), 481
drivers. *See* device drivers
drives. *See* CD-ROM drives; disks; tape
drivesDriveSpace command (drvspace)
command reference, 481–85
compressing floppy disks, 193–94
compressing hard disks, 191–93
DoubleSpace and, 174, 191 (*see also* DoubleSpace command (dblspace))
formatting disks, 505
memory and, 191, 486
mounting and unmounting drives, 194–96
removing, 196–97
ScanDisk and, 192
Smartdrv and, 383
drivparm. *See* Drive Parameters configuration command (drivparm)
drvspace. *See* DriveSpace command (drvspace
DRVSPACE.000 file, 196–97
DRVSPACE.SYS device driver, 191–92, 486
duplication. *See* copying files; Diskcopy command

Index

E

Echo batch command, 338–39, 347–50, 357, 435, 486–87

Edit command
- accessing, from Dosshell, 262
- changing screen display, 294–95
- command reference, 487–89
- copying or moving text, 290–91
- copying text from another file, 293–94
- deleting text, 288
- editing files, 290–91
- entering text, 285
- exiting, 290
- inserting text, 285–86
- inserting vs. overstriking, 293
- keyboard and mouse use in, 279–81
- margins, 286–87
- printing files, 287
- replacing text, 292
- saving files, 288–89
- searching files, 291
- selecting text, 281
- starting, 281
- Survival Guide help, 281–85

editors. *See* Edit command; Edlin command

Edlin command, 279, 489–95

EGA (Enhanced Graphics Adapter), 124–25

ellipsis (...), 249

EMM386 command, 495–96

EMM386.EXE device driver, 379–81, 387–90, 496–97

end-of-file character, 48, 133

Enhanced Graphics Adapter (EGA), 124–25

enhanced keyboards, 576

Enter key, 17

environment variables, 566

erase. *See* Delete command (del, erase)

errorlevel values
- Backup, 433–34

batch file testing of, 368–70, 514–15
Replace, 561
Restore, 562–63
Escape key (Esc), 31, 284
Exe2bin command, 497
EXE extension, 67, 316
EXE files, converting, to COM files, 497
Exit command, 245, 263, 497
Expand command, 498
expanded memory (EMS), 122, 378–79
disk caches, 384, 572–73
emulating IBM PS/2, 589
enabling and disabling, 495–96
Fastopen use of, 397
LIM EMS, 589
managers, 378–81, 589
RAM disks, 386–87, 557
simulating, with extended memory, 495–97
extended memory (XMS), 122, 378
disk caches, 383–84, 570–72
disk compression in, 191, 480–81, 486
manager, 378, 513–14
memory-resident programs in, 388, 525–26
MS-DOS in, 380–81, 472–73
RAM disks, 386, 556–57
simulating expanded memory with, 495–97
extensions, filename, 26, 43, 65–67, 202
external commands, 42–43

Index

F

F1 function key, 246, 248, 284
F3 function key, 14, 26
F5 function key, 392, 576
F6 function key, 133, 284
F7 function key, 37
F8 function key, 392, 576
F9 function key, 38, 253
F10 function key, 245
Fasthelp command, 512. *See also* Help command
Fastopen command, 396–97, 498–99
fc. *See* File Comparison command (fc)
FCBS configuration command, 501
Fdisk command, 501–2
File Comparison command (fc), 88–91, 110, 499–501
file control blocks, 501
file handles, 503
File List, Dosshell, 242
  copying files, 251–52
  deleting files, 254–55
  displaying directories, 248
  displaying directory structures, 261
  displaying files, 260–61
  displaying files in sorted order, 255–56
  displaying two directories, 258–59
  expanding and collapsing directories, 261
  file information options, 256–57
  File menu, 249
  finding files, 259–60
  Options menu, 254–57
  scrolling directories, 250–55
  selecting directories, 250
  selecting files, 250–51
  Tree menu, 261
  viewing file contents, 253
  view options, 257–61

file-management example, 300–312, 352–57

File menu
  Dosshell, 249, 265–68
  Edit, 488

FILEnnnn.CHK files, 114

files, 8. *See also* batch files; program files; text files
  associating data files with program, 251
  backing up (*see* backing up files and directories)
  comparing, 88–91, 446–48, 499–501
  compressing (*see* compression, disk)
  controlling access (*see* attributes, file and directory)
  converting EXE files to COM, 497
  copying (*see* copying files)
  defragmenting, 199–201, 459–61
  deleting (*see* deleting files)
  directories of, 26–28 (*see also* directories; tree-structured file systems)
  displaying information about, 256–57
  expanding compressed MS-DOS, 498
  finding, with Dosshell, 259–60
  fragmented, 112, 199
  moving (*see* moving files)
  MS-DOS as, 8
  MS-DOS temporary, 54, 310
  naming (*see* names, file)
  overwriting (*see* overwriting files)
  recovering, from damaged disks, 557–58
  required in root directory, 202, 411
  restoring (*see* restoring files and directories)
  search paths (*see* paths)
  selecting, with Dosshell, 250–51
  sharing, 568
  specifying number of open, 391, 501, 503
  speeding access (*see* disk buffers; disk caches; Fastopen command)
  system, 407–8, 576–77
  types, 42–43
  undeleting (*see* undeleting files)
  virus protection (*see* virus protection)

Files configuration command, 391, 503

filter commands
  combining, 54–55, 309
  pausing displays, 49, 53–54, 308, 543–44
  redirection and, 300–302, 307–8, 310
  searching text for characters, 54–55, 304–7, 352–57, 371–72, 503–4

sorting, 52–53, 302–4, 573–74

Find filter command, 54–55, 304–7, 352–57, 371–72, 503–4

fixed disks. *See* hard disks

floppy disks. *See also* disks
capacities, 7, 27, 103, 105, 106, 506
checking, for viruses, 221 (*see also* virus protection)
comparing, 110–11, 469–70
composition of, 101–2
compressing, 193–94 (*see also* compression, disk)
displaying status of, 111–14
duplicating, 100–101, 108–10, 471–72
formatting, 46–48, 104–8 (*see also* Format command; Select command)
formatting, during backup, 233
handling, 99–100
information storage on, 101–4
system (*see* system disks)
unformatting, 217–18
volume labels, 104, 106, 114–15

For batch command, 363–66, 435, 504–5

foreign languages
character sets (*see* code pages)
country-specific information, 547–48
country-specific system floppy disks, 564–65
date, time, currency, and decimal formats, 454–58, 547–48
keyboard layouts, 135–39, 520–23

Format command. *See also* Unformat command
Backup and, 233
command reference, 505–7
compressed floppy disks and, 194
floppy disks, 46–48, 104–8
hard disks, 106–7, 407
parameters, 105–7
reformatting or quick formatting, 216

formatting disks. *See* Format command; Select command

fragmented files, 112, 199–201

full backups, 226–27

function keys. *See also individual function key names*
Doskey use of, 36
Dosshell use of, 14, 26
Edit Survival Guide use of, 284
end-of-file character, 48, 133

Index

G

Goto batch command, 342–46, 435, 508

graftable. *See* Load Graphics Table command (graftable)

graphics
  enabling display of, 508–9
  enabling printing of, 134–35, 509–11

Graphics command, 134–35, 509–11

graphics mode, 243

grayed characters, 249

greater-than symbol (>), 54, 299, 307–8, 310, 515

PREVIOUS NEXT

Index

H

hard disks. *See also* disks
  backing up (*see* backing up files and directories)
  capacities, 7, 27, 174
  checking, for viruses, 219–21 (*see also* virus protection)
  compressing, 191–93 (*see also* compression, disk)
  configuring, 501–2
  defragmenting files, 199–201, 459–61
  directory structure (*see* directories; files; tree-structured file systems)
  formatting, 106–7, 407 (*see also* Format command)
  installing applications, 175–78
  installing or upgrading MS-DOS (*see* versions, MS-DOS)
  installing selected files, 184–90
  maintenance tips, 202–3
  management tips, 174, 201–3
  MS-DOS files on, 201–2
  partitions (*see* partitions, hard disk)
  rebuilding, 218 (*see also* Unformat command)
  restoring (*see* restoring files and directories)
  saving recovery information for, 216, 531–33, 583–85
  speeding access (*see* disk buffers; disk caches; Fastopen command)

hardware, 6. *See also* devices; disks; display adapters; keyboards; printers

Help command, 71–73, 512–13

help features
  Dosshell, 246–47
  Dosshell program items, 269
  Edit, 281–85
  MSbackup, 226
  MS-DOS commands, 71–73, 512–13

Hewlett-Packard Laserjet Plus printer, 23, 35

hexadecimal format, 253

hidden attributes, 178, 181–83, 187

high memory area (HMA), 122. *See also* extended memory (XMS)
  disk compression in, 191, 480–81, 486
  MS-DOS in, 380–81, 472–73

HIMEM.SYS device driver, 379, 381, 387, 513–14

history, command. *See* Doskey command

HLP extension, 67

Index

I

IBMBIO.COM and IBMDOS.COM files, 239, 576

IBM-compatible computers, 9

IBM-compatible keyboards, 17, 30, 36

IBM printers, 134–35

IBM PS/2 computers, 589

If batch command, 340–42, 435, 514–15

Include configuration command, 515

incremental backups, 227

index, Dosshell help, 247

initializing disks. *See* Format command; Select command

input, standard, 300, 515–16

input and output redirection
- appending redirected output to files, 308
- command input, 300
- command output, 299
- command output to parallel printer, 54, 170, 307–8
- command reference, 515–16
- filter commands and, 300–302, 307–8, 310 (*see also* filter commands)
- parallel printer output to serial printers, 130–31, 540
- piping and, 53–54, 308–10
- redirecting both input and output, 307–8

inserting vs. overstriking text, 293

Install configuration command, 397, 516

installing or upgrading MS-DOS. *See* versions, MS-DOS

InterInk command, 517

INTERLNK.EXE device driver, 390–91, 517–18

internal commands, 42–43

international support. *See* foreign languages

Intersvr command, 518

IO.SYS file, 236, 576

Index

J–K

Join command, 519

Keyboard command (keyb), 135–39, 520–22

Keyboard command (keybxx), 522–23

keyboards. *See also* consoles; keys
- blocking functions of enhanced, 576
- changing layouts of, 135–39, 520–23
- control program (*see* ANSI.SYS device driver)
- country-specific system floppy disks and, 564–65
- device name, 48, 66, 119–20
- Dosshell use of, 245–46, 250, 251
- Edit use of, 279–81
- PC-compatible, 17, 30, 36
- printing from, 133
- repeat rate, 123–24, 536–37

KEYBOARD.SYS file, 138

keybxx. *See* Keyboard command (keybxx)

keys. *See also* characters; keyboards
- Alternate, 32, 246
- arrow (*see* arrow keys)
- Backspace, 18, 20–21
- Break, 32
- Control, 31, 33–35, 251
- dead, 137–38
- Delete, 32
- Doskey use of, 35–38, 311
- Dosshell use of, 250, 263–64, 270
- Edit use of, 290–91
- Enter, 17
- Escape, 31, 284
- function (*see* function keys)
- Num Lock, 31–32, 548
- Pause, 32, 33
- screen-printing, 22–23, 33, 127
- Shift, 31, 251, 281
- Spacebar, 251
- special, 17–18, 20–21, 30–38
- Tab, 245
- Up Arrow, 17, 18, 20, 37

Index

L

Label command, 114–15, 523–24

labels, batch file, 342–43, 508

labels, disk volume
  changing or deleting, 114–15, 523–24
  displaying, 115, 586
  formatting and, 47, 104, 106

language, Basic, 262, 555

languages, foreign. *See* foreign languages

laser printers, 23, 35, 126

Lastdrive configuration command, 391–92, 524

less-than symbol (<), 52, 300, 515

lh. *See* Loadhigh command (lh)

LIM EMS (Lotus-Intel-Microsoft Expanded Memory Specification), 589. *See also* expanded memory (EMS)

line editor, 489–95

line spaces, displaying, with batch files, 347–50, 357

Loadfix command, 524

Load Graphics Table command (graftabl), 508–9

Loadhigh command (lh), 380, 382, 387–88, 525–26

loading MS-DOS. *See* starting MS-DOS

look-ahead buffers, 391, 437–38

LPT1-LPT3 device names, 120. *See also* parallel ports

Index

M

macros. *See* Doskey command

maintenance, hard disk, 202–3

Make Directory command (md, mkdir), 144–45, 150–53, 526–27

margins, Edit, 286–87

markers, directory, 145, 153–58

md. *See* Make Directory command (md, mkdir)

mem. *See* Memory command (mem)

MemMaker command, 382–83, 529

memory
- adapter, 122
- balancing, during MS-DOS installation, 405–6
- conventional, 122, 378
- disk buffers, 391, 437–38
- disk caches, 383–84, 570–73
- disk compression and, 191, 480–81, 486
- displaying usage of, 121–23, 527–28
- expanded (*see* expanded memory (EMS))
- extended (*see* extended memory (XMS))
- loading programs above first 64 KB of, 524
- managers, 378–81, 495–97, 513–14, 589
- optimizing use of, 382–83, 529
- RAM disks, 384–87
- reserved or upper (*see* reserved memory)
- reserving, for temporary program use, 574
- running MS-DOS in high memory area, 380–81, 472–73
- types of, 122, 378–81

Memory command (mem), 121–23, 381, 527–28

memory-resident programs, 263, 388, 516, 525–26

Menucolor configuration command, 529–30

Menudefault configuration command, 530

Menuitem configuration command, 530–31

menu systems, 9
- of configurations, 13, 393–94, 529–31, 574
- Dosshell (*see* Program List, Dosshell)
- interactive batch files as, 368–70

messages
displaying, with Echo, 338–39, 347–50, 357, 435, 486–87
displaying, with Pause, 339, 435, 549
displaying, with Remark, 319–20, 348, 436, 558

Microsoft Anti-Virus command (msav). *See also* Vsafe command
checking floppy disks, 221
checking for unknown viruses, 221–22
checking hard disks, 219–21
command line parameters, 223
command reference, 545–46
updating software using, 222–23

Microsoft Backup. *See* MSbackup command

Microsoft CD Extension. *See* MS cd extension command (mscdex)

Microsoft Diagnostics command (msd), 546–47

Microsoft MS-DOS
commands (*see* commands, MS-DOS)
configuration (*see* CONFIG.SYS configuration file)
devices (*see* devices)
disk management (*see* disks; floppy disks; hard disks)
as disk operating system, 6–10
DOS subdirectory for, 27–28, 175, 201–2, 408–11
editors (*see* Edit command; Edlin command)
file management (*see* directories; files; tree-structured file systems)
installing or upgrading (*see* versions, MS-DOS)
international support (*see* foreign languages)
memory management (*see* memory)
memory-resident programs, 263, 388, 516, 525–26
optimization (*see* optimization)
QBasic, 262, 555
root directory files, 202, 411
running, in high memory, 380–81, 472–73
shell (*see* Dosshell command)
starting (*see* starting MS-DOS)
system prompt (*see* system prompt)
temporary files, 54, 310
virus protection (*see* virus protection)

Microsoft Windows
Check Disk and, 168, 441
Dosshell and, 242
MSbackup for, 232
opening screen, 13
starting MS-DOS without, 15
Undelete for, 206

WINA20.386 file, 576
minus sign (-), 244, 261
Mirror command, 212, 216, 531–33, 584
MIRROR.FIL file, 217, 531–32
mkdir. *See* Make Directory command (md, mkdir)
Mode command, 119
ANSI.SYS and, 125, 332–33, 358
changing display characteristics, 124–25, 534–35
changing display columns or lines, 535–36
changing keyboard repeat rate, 123–24, 536–37
changing parallel printer width and spacing, 125–28, 328, 539–40
command reference, 534–43
connecting serial printers, 130–31, 540
displaying and changing code-page status, 543
displaying device status, 131–32, 540–41
preparing code pages, 541–42
restoring code pages, 543
selecting code pages, 542
setting communications parameters, 128–30, 537–38
More filter command, 49, 53–54, 308, 543–44
mounting drives, 194–96
mouse
Dosshell use of, 245–46, 250, 251, 252, 254
Edit use of, 279–81
MSbackup use of, 228, 229
Move command, 83, 158–60, 201, 346, 544–45
moving files. *See also* copying files
with batch files, 360–61
with Dosshell, 252
with Move, 83, 158–59, 201, 544–45
msav. *See* Microsoft Anti-Virus command (msav)
MSbackup command
backing up files, 230–31
backup types, 226–27
command reference, 546
configuration, 224–26
online help, 226
restoring files, 231–32
screen elements, 227–28
selecting files, 228–30
MS cd extension command (mscdex), 546

msd. *See* Microsoft Diagnostics command (msd)
MS-DOS. *See* Microsoft MS-DOS
MSDOS.SYS file, 239, 576

PREVIOUS NEXT

Index

N

names, device, 66, 119–20

names, directory, 159–60

names, disk volume. *See* labels, disk volume

names, file. *See also* paths
  changing, 86–87, 559–60
  of data files, 202
  special, 66–67, 202
  valid, 26, 43, 65–67

national language support, 547–48. *See also* foreign languages

National Language Support Function command (nlsfunc), 547–48

networks, 120, 390–91, 517–18, 568

nlsfunc. *See* National Language Support Function command (nlsfunc)

Numlock configuration command, 548

Num Lock key, 31–32, 548

Index

O

online help, 71, 226, 282, 512. *See also* help features

operating systems. *See* Microsoft MS-DOS; Microsoft Windows

optimization
  disk buffers, 391, 437–38
  disk caches, 383–84, 570–73
  disk compression (*see* compression, disk)
  file and directory access, 396–97, 498–99
  file defragmentation, 199–201, 459–61
  memory, 382–83, 529 (*see also* memory)
  RAM disks, 384–87, 555–57, 585

Options menu
  Dosshell, 254–57, 263–64
  Edit, 489

output. *See also* input and output redirection; printing; screens
  command, 52
  standard, 299, 515–16

overstriking vs. inserting text, 293

overwriting files
  with Copy, 79, 82, 451, 454
  with Move, 158–59, 545
  with Replace, 186

OVL extension, 67

Index

P

parallel ports, 120. *See also* printers
  redirecting output to serial ports, 130–31, 540

parameters
  command, 28, 44
  command, Dosshell program item, 269–70
  communications, 128–30, 537–38
  replaceable (*see* replaceable parameters)
  shifting, in batch files, 362–63, 569

parent directory (..), 145, 153–58

parity, 128–29

partitions, hard disk
  configuring, 501–2
  rebuilding, 218, 583–85
  saving recovery information, 212, 216, 531–33

PARTNSAV.FIL file, 584

passwords, Dosshell, 267

paste, cut and, 290–91

Path command, 165–66, 171, 324–26, 548–49

paths
  AUTOEXEC.BAT and, 324–26, 408, 410
  command search, 165–66, 548–49
  data search, 171–72, 428–29
  directory names and, 145–47, 149
  Dosshell program items, 270

Pause batch command, 339, 435, 549

Pause key, 32, 33

pause messages, Dosshell program item, 270

PC-compatible keyboards, 17, 30, 36

PCTRACKR.DEL file, 208, 212, 532, 582

percent sign (%) as batch replaceable parameter, 320–21

percent signs (%%) as batch replaceable parameter, 364–66

period (.) as date separator, 20

permanent commands, 42–43

phone-list example, 300–312, 352–57

piping, 53–54, 308–10, 515–16

plus sign (+), 82, 244, 261

pop-up programs, 263, 388, 516, 525–26

portable computers, configuring, 390–91, 517–18, 549–50

ports. *See* parallel ports; serial ports

Power command, 549–50

POWER.EXE device driver, 390–91, 550

Print command, 50, 91–95, 551–53

printers. *See also* printing
- changing width and spacing for parallel, 125–28, 328, 539–40
- connecting serial, 130–31, 540
- copying files to, 81
- copying from keyboard to, 133
- device names, 50, 54, 92, 120
- graphics support for, 134–35, 510
- laser, 23, 35, 126
- redirecting command output to, 170

printing. *See also* printers
- command output, 54, 170, 307–8
- graphics, 134–35, 509–11
- from keyboards, 133
- screens, 22–23, 33, 34–35, 127
- text files, in queues, 50, 91–95, 551–53
- text files, with batch files, 327–28
- text files, with Copy, 50, 81
- text files, with Edit, 287

print queue, 91. *See also* Print command

Print Screen key (PrtSc, Print Scrn), 23, 33, 127

PRN device name, 50, 54, 92, 120. *See also* parallel ports

program files, 6, 42. *See also* commands, MS-DOS; files; Program List, Dosshell
- associating data files with, 251
- converting EXE programs to COM, 497
- displaying memory use by, 121–23, 381
- expanding compressed MS-DOS, 498
- installing applications, 175–78
- installing MS-DOS, 564, 566–67
- loading, above first 64 KB of memory, 524
- loading, into reserved memory, 388, 516, 525–26
- MS-DOS as, 6
- reserving memory for temporary use by, 574
- search paths (*see* paths)

starting, from Dosshell, 262–63
switching between, 263–64
updating, with Microsoft Anti-Virus, 222–23 (*see also* Microsoft Anti-Virus command (msav))

program groups, 262
adding, 268–69
deleting, 274–75
Disk Utilities, 272–74
selecting, 264–65

program items, 262
adding, 265–66
changing, 266–67
controlling pause message, 270
deleting, 267–68
designing dialog boxes, 271
specifying parameters, 269–70
specifying paths, 270
specifying shortcut keys, 270
testing, 271–72

Program List, Dosshell, 243
adding program groups, 268–69
adding program items, 265–66
changing program items, 266–67
controlling pause messages, 270
deleting program groups, 274–75
deleting program items, 267–68
designing dialog boxes, 271
Disk Utilities Program Group, 272–74
File menu, 265–68
selecting program group, 264–65
specifying command parameters, 269–70
specifying paths, 270
specifying shortcut keys, 270
starting programs, 262–63
switching programs, 263–64
testing program items, 271–72

prompt, system. *See* system prompt

Prompt command, 39, 153, 324–26, 395–96, 553–54

properties, Dosshell program item, 266–67

PrtSc key, 23, 33, 127

PS/2 computers, 589

Index

Q

QBasic command, 262, 555

question mark character (?), 45, 70-71, 85-86

queue, print, 91. *See also* Print commmand

quick formatting disks, 106, 216, 507

Index

R

RAM (random access memory), 378

RAM disks, 384–87, 555–57, 585

RAMDRIVE.SYS device driver, 384–87, 555–57, 585

rd. *See* Remove Directory command (rd, rmdir)

read-only attributes, 84, 178, 183

rebooting MS-DOS, 32, 33, 38

Recover command, 557–58

recovering deleted files. *See* undeleting files

redirection. *See* input and output redirection

reformatting disks, 106, 216, 507

Remark batch command (rem), 319–20, 348, 436, 558

Remove Directory command (rd, rmdir), 160–62, 559

Rename command (ren), 86–87, 559–60

renaming directories, 159–60

replaceable parameters
  batch files, 320–21, 364–66
  Doskey macros, 331–33

Replace command, 81, 184–86, 409, 560–61

rerouting. *See* input and output redirection

reserved memory, 122, 378
  enabling, 387, 495–97
  loading device drivers into, 388–90, 465–66
  loading programs into, 388, 525–26
  managing, 380–81, 387–90

resizing Help windows, 284–85

restarting MS-DOS, 32, 33, 38

Restore command, 237–39, 561–63. *See also* Backup command

restoring files and directories. *See also* backing up files and directories
  with MSbackup, 231–32, 546
  with Restore, 237–39, 561–63

reverse slash character (\), 31

rmdir. *See* Remove Directory command (rd, rmdir)

root directory, 142, 149
marker, 145, 149
required MS-DOS files, 202, 411

PREVIOUS NEXT

Index

S

ScanDisk command, 174, 197–98, 563–64
  Check Disk and, 112, 113, 441 (*see also* Check Disk command (chkdsk))
  DriveSpace and, 192

screens. *See also* consoles; display adapters
  changing, in Dosshell, 248
  changing, in Edit, 294–95
  clearing, 23, 123, 348, 445
  control program (*see* ANSI.SYS device driver)
  device name, 48, 66, 119–20
  displaying graphics characters, 508–9
  pausing displays, 49, 53–54, 308, 543–44
  printing, 22–23, 33, 34–35, 127
  printing graphics images, 134–35, 509–11

scroll bar, 250

searching text for characters, 54–55, 304–7, 352–57, 371–72, 503–4

Search menu, Edit, 488–89

search paths. *See* paths

sectors, 102–3, 112

Select command, 405–6, 564–65

serial numbers, volume, 47, 104, 115

serial ports, 81, 119
  device names, 120
  displaying status of, 131–32
  redirecting parallel port output to, 130–31, 540
  setting communications parameters, 128–30, 537–38

serial printers, 130–31, 540

server-client connections, 517–18

Set command, 468, 566

Setup command, 404–5, 566–67, 578

SETVER.EXE device driver, 567

Set Version command (setver), 567–68

Share command, 568

Shell configuration command, 568

shell programs, 14–16, 242. *See also* Dosshell command

Shift batch command, 362–63, 436, 569
Shift-Del keys, 291
Shift-Enter keys, 264
Shift-F8 keys, 251
Shift-F9 keys, 245
Shift-Ins keys, 290–91
Shift keys, 31, 251, 281
Shift-Print Screen keys, 23, 33, 35, 127, 134–35
Shift-Tab keys, 245
shortcut keys
  Dosshell, 250, 270
  Edit, 290–91
slash (/) as date separator, 20
Smartdrv command, 383–84, 570–72
SMARTDRV.EXE device driver, 572
SMARTDRV.SYS device driver, 383–84, 572–73
software, 6, 175–78, 222–23. *See also* program files
Sort filter command, 52–53, 302–4, 573–74
sorting
  directory displays, 76–77
  directory displays in Dosshell, 255–56
  directory displays with batch files, 372–74
  text lines, 52–53, 302–4, 573–74
Spacebar key, 251
special keys, 17–18, 20–21, 30–38. *See also* keys
spooling. *See* Print command
Stacks configuration command, 574
standard input, 300, 515–16
standard output, 299, 515–16
starting MS-DOS. *See also* system prompt
  with AUTOEXEC.BAT, 323–27
  with date and time requests, 15–16
  with (or without) Dosshell, 14–15 (*see also* Dosshell command)
  from floppy disk, 12
  in high memory, 380–81, 472–73
  with menu of configurations, 13, 393–94, 529–31, 574
  restarting, 32, 33, 38

with shell programs, 16
without Windows, 15
Startup Menu, 13, 393–94, 529–31, 574
stop bits, 128–29
strings, 54
subdirectories, 27–28, 142, 149. *See also* directories
Submenu configuration command, 574
Substitute command (subst), 397–98, 574–75
Supplemental Disk, 242, 475, 489, 497, 519, 531
Survival Guide, Edit, 281–85
Switches configuration command, 576
syntax, command, 73
sys. *See* System command (sys)
SYS extension, 67
system attributes, 178, 187
system clock. *See* date; time
System command (sys), 407–8, 576–77
system configuration. *See* CONFIG.SYS configuration file
system date. *See* date
system disks
copying system files, 407–8, 576–77
country-specific floppy disks, 564–65
formatting (*see* Format command; Select command)
starting MS-DOS from, 12, 388
system files, 407–8, 576–77
system memory. *See* memory
system messages. *See* messages
system programs, 6
system prompt
accessing, from Dosshell, 245, 262–63
AUTOEXEC.BAT and, 324–26
changing, 39, 153, 395–96, 553–54
current directory as, 146, 153, 175
current drive as, 16
system setup. *See* starting MS-DOS; versions, MS-DOS
system shutdown, 23

system time. *See* time

Index

T

Tab key, 245

tape drives, 481

Task Swapper, Dosshell, 263–64

telephone file example, 300–312, 352–57

temporary commands, 42–43

temporary files, MS-DOS, 54, 310

terminate-and-stay-resident (TSR) programs, 263, 388, 516, 525–26

text files, 42. *See also* batch files; files
- appending redirected output to, 308
- combining, 82–83, 453–54
- copying (*see* copying files)
- creating, with Copy, 48–49, 67–68
- creating, with Edit (*see* Edit command)
- creating, with Edlin, 279, 489–95
- deleting (*see* deleting files)
- printing, in queues, 50, 91–95, 551–53
- printing, with batch files, 327–28
- printing, with Copy, 50, 81
- searching for characters in, 54–55, 304–7, 352–57, 371–72, 503–4
- sorting lines, 52–53, 573–74
- viewing contents of, with batch files, 370–71
- viewing contents of, with Dosshell, 253
- viewing contents of, with Type, 49–50, 77–78, 580

text mode, 14, 243

time
- changing, 21–22, 577–78
- formats, 454–58, 547–48
- starting MS-DOS with, 15–16

Time command, 21–22, 577–78

tracks, 102–3

Tree command, 166–70, 578–80

Tree menu, Dosshell, 261

tree-structured file systems. *See also* directories
- adding levels to, 150–60
- creating, 143–49
- defining directories, 142–43

displaying structure of, 166–70, 578–80
managing, 160–72
TSR (terminate and stay resident) programs, 263, 388, 516, 525–26
Type command, 49–50, 77–78, 580

Index

U

UMBs (upper memory blocks). *See* reserved memory

uncompressing disks, 196-97. *See also* compression, disk

unconditional format, 106, 507

Undelete command, 71, 206-15, 580-83

undeleting files, 206-15. *See also* deleting files
  command reference, 580-83
  delete tracking, 211-15, 531-33
  levels of protection, 207-11

Unformat command, 106, 217-18, 583-85. *See also* Format command

Uninstall command, 405

unmounting disk drives, 194-96

Up Arrow key, 17, 18, 20, 37. *See also* arrow keys

updating software using Microsoft Anti-Virus, 222-23

upgrading MS-DOS. *See* versions, MS-DOS

upper memory blocks (UMBs). *See* reserved memory

user-defined commands. *See* batch files; Doskey command

Index

V

variables, environment, 566

VDISK.SYS device driver, 384–87, 555–57, 585

ver. *See* Version command (ver)

Verify command, 585

Version command (ver), 18–19, 394, 585–86

versions, MS-DOS, 8
- backup files, 233, 434
- checking disk condition, 112, 113
- choosing startup commands, 392–93
- directory listings, 49
- disk compression, 191
- displaying version numbers, 18–19, 394, 585
- Doskey, 329
- Dosshell, 242
- DOS subdirectory, 28
- editors, 279
- emulating earlier, 567, 567–68
- floppy disk duplication, 109
- hard disk management, 174
- help features, 71
- installation commands, 404, 404–6, 564, 566–67
- installing or upgrading to version 3, 407–11
- installing or upgrading to version 4, 405–6
- installing or upgrading to version 5 or later, 404–5
- memory management features, 379
- overwriting files (*see* overwriting files)
- time formats, 21
- Xcopy and file attributes, 187

Video Graphics Array (VGA), 124–25

View menu, Dosshell, 257–61

virtual disks. *See* RAM disks

virus protection, 218–24
- checking floppy disks, 221
- checking for unknown viruses, 221–22
- checking hard disks, 219–21
- command line parameters, 223
- constant protection with Vsafe, 223–24, 586–87

Microsoft Anti-Virus command reference, 545–46
updating software, 222–23
Volume command (vol), 115, 586
volume file, compressed, 193
volume labels. *See* labels, disk volume
volume serial numbers, 47, 104, 115
Vsafe command, 223–24, 586–87. *See also* Microsoft Anti-Virus command (msav)

Index

W

wildcard characters, 45, 69–71, 344–45

WINA20.386 file, 576

windows
  parts of Dosshell, 243–45
  sizing Edit help, 284–85
  switching Edit, 284

Windows (operating system). *See* Microsoft Windows

word processing. *See* Edit command

Index

X

Xcopy command, 81, 176–77, 186–90, 588–89

XMA2 EMS.SYS device driver, 589

XMAEM.SYS device driver, 589

PREVIOUS NEXT

List of Figures

Chapter 2: Starting MS-DOS

Figure 2-1: The opening (Program Manager) screen of Microsoft Windows.

Figure 2-2: The Backspace, Enter, and Up arrow keys on the early PC-compatible keyboard.

Figure 2-3: The Backspace, Enter, and Up arrow keys on the enhanced PC-compatible keyboard.

Chapter 3: Getting Your Bearings

Figure 3-1: A sample directory display of version 6.0 MS-DOS files.

Figure 3-2: A sample directory entry.

Figure 3-3: Special keys on the early IBM PC-compatible keyboard.

Figure 3-4: Special keys on the enhanced IBM PC-compatible keyboard.

Figure 3-5: Control key combinations.

Figure 3-6: Keys on the early PC-compatible keyboards that have special meaning to Doskey.

Figure 3-7: Keys on the enhanced PC-compatible keyboard that have special meaning to Doskey.

Chapter 5: Managing Your Files

Figure 5-1: Some valid and invalid file names.

Figure 5-2: Some special MS-DOS file name extensions.

Chapter 6: Managing Your Floppy Disks

Figure 6-1: A 5.25-inch floppy disk.

Figure 6-2: A 3.5-inch floppy disk.

Figure 6-3: Tracks and sectors on a floppy disk.

Figure 6-4: Storage capacity of different floppy disks.

Figure 6-5: Ways to type <size> as part of the /F:<size> parameter of the Format command in versions 4 and later.

Chapter 7: Managing Your Devices

Figure 7-1: MS-DOS device names.

Figure 7-2: Common displays and adapters.

Figure 7-3: Serial communications parameters.

Figure 7-4: Printers supported by the Graphics command.

Chapter 8: A Tree of Files

Figure 8-1: Two-level file system.

Figure 8-2: Three-level file system.

Figure 8-3: Two-department file structure.

Chapter 9: Managing Your Hard Disk

Figure 9-1: Subdirectories and files for hard disk examples.

Figure 9-2: DriveSpace's main screen.

Chapter 11: The MS-DOS Shell

Figure 11-1: Parts of the MS-DOS Shell window in versions 5 and later.

Figure 11-2: Basic keyboard and mouse techniques.

Chapter 12: Creating and Editing Files of Text

Figure 12-1: Edit's cursor-movement keys.

Chapter 13: Taking Control of Your System

Figure 13-1: Telephone and business-card list.

Figure 13-2: Column numbers of items in the telephone list.

Figure 13-3: The Doskey editing keys. A number of function keys are also available for editing; see the online Help Doskey.

Chapter 14: Creating Your Own Commands

Figure 14-1: Replaceable parameters in a batch file.

Chapter 16: Creating More Smart Commands

Figure 16-1: Chaining batch files.

Appendix C: MS-DOS Command Reference

Figure C-1: Valid country codes and code pages for different versions of MS-DOS.

Figure C-2: Device drivers shipped with MS-DOS

Figure C-3: Doskey special character combinations.

PREVIOUS

Running MS-DOS Version 6.22, 20th Anniversary Edition

by Van Wolverton

ISBN:0735618127

Microsoft Press © 2003 (612 pages)

The classic reference to the classic operating system!

Table of Contents

Running MS-DOS Version 6.22, 20th Anniversary Edition

Preface to the 20th Anniversary Edition

Introduction

Part I - Getting to Know MS-DOS

Chapter 1 - What is MS-DOS?
Chapter 2 - Starting MS-DOS
Chapter 3 - Getting Your Bearings
Chapter 4 - A Look at Files and Floppy Disks

Part II - Learning to Use MS-DOS

Chapter 5 - Managing Your Files
Chapter 6 - Managing Your Floppy Disks
Chapter 7 - Managing Your Devices
Chapter 8 - A Tree of Files
Chapter 9 - Managing Your Hard Disk
Chapter 10 - Protecting Your Disks and Files
Chapter 11 - The MS-DOS Shell
Chapter 12 - Creating and Editing Files of Text
Chapter 13 - Taking Control of Your System
Chapter 14 - Creating Your Own Commands
Chapter 15 - Creating Smart Commands

Chapter 16 - Creating More Smart Commands
Chapter 17 - Tailoring Your System
Part III - Appendixes
Appendix A - Installing MS-DOS
Appendix B - Glossary
Appendix C - MS-DOS Command Reference
Index

List of Figures