Osborne McGraw-Hill

PC-DOS

TIPS & TRAPS

Dick Andersen, Janice M. Gessin, Fred Warren, and Jack Rodgers

PC-DOS
Tips & Traps

Includes MS-DOS®

PC-DOS
Tips & Traps

Includes MS-DOS®

Dick Andersen
Janice M. Gessin
Fred Warren
and
Jack Rodgers

Osborne **McGraw-Hill**
Berkeley, California

Osborne **McGraw-Hill**
2600 Tenth Street
Berkeley, California 94710
U.S.A.

For information on translations and book distributors outside of the U.S.A., please write to Osborne **McGraw-Hill** at the above address.

A complete list of trademarks appears on page 213.

PC-DOS Tips & Traps Includes MS-DOS®

1234567890 DODO 898765

ISBN 0-07-881194-5

Cynthia Hudson, Acquisitions Editor
Kay Nelson, Technical Editor
Jack Trainor, Technical Reviewer
Jean Stein, Senior Editor
Elizabeth Fisher, Editorial Assistant
David Sweet, Copy Editor
Judy Wohlfrom, Text Design
Yashi Okita, Cover Design
Gayle Zanca, Composition

Dedicated to Dr. James S.B. Mackie

D.A.

To Grandma, who always believed in me.

J.M.G.

Contents

Acknowledgments

Thanks to Cindy Hudson, Jean Stein, Kay Nelson, and Jack Trainor for their help in the planning, editing, and review of the book.

Introduction

Most introductions are not essential to the books they introduce. This one is important because it tells how to use this unique book. Please read it!

Why This Book Is Unique

IBM PC-DOS is a version of MS-DOS (Microsoft Disk Operating System) designed especially for the IBM PC. Many books have been written on PC-DOS, which is sometimes referred to simply as DOS. Unfortunately, few of these books are easy to use. Most PC users want to solve immediate problems with practical solutions and to perform specific business tasks without reading through a lot of text. The unique and well-proven "Tips and Traps" format in this book gives the user concentrated, "bite-sized" nuggets of information that are immediately relevant to the task at hand. For other examples of the Tips and Traps format, see Dick Andersen's other books: *Jazz Tips and Traps* (Osborne, 1985), *dBASE III Tips and Traps* and *AppleWorks Tips and Traps* (Osborne, 1986), and *1-2-3 Tips, Tricks, and Traps* (Que, 1984).

This is a hands-on book. We have deliberately used simple examples so that you can re-create them on your own PC in a few minutes. We assume that you

will keep *PC DOS Tips and Traps* next to your PC at all times and refer to it every time you encounter a problem or say to yourself, "I wonder if there's a way to...."

This book is unique because of its modular format; it is also unique because of its emphasis on Traps. One of the most frustrating problems for a microcomputer user is encountering roadblocks in finishing a task. Most books give too little attention to what can go wrong. The Trap entries in this book, however, focus on preventing problems and finding cures. They have one purpose in common: to help you, the user, stay out of trouble.

Who Can Use This Book

Each separate Tip or Trap is independent of the others, which means that it was possible for us to mix beginning, intermediate, and advanced material in one book. That's why we can say with confidence that this book is for everyone who uses a microcomputer that runs any version of MS-DOS, including PC-DOS on the IBM Personal Computer and compatibles.

There is very little difference among the various species of MS-DOS that have the same integer version number. For instance, PC-DOS version 2.1 is virtually identical to MS-DOS version 2.11. We focus primarily on IBM PC DOS because this species is the most widespread. We also look at version 2.1 of PC-DOS because it has become the current standard.

What's Missing From DOS

One of the reasons a book like this is necessary is that PC-DOS is not very "friendly" to the nontechnical user. It was developed on the pattern of the older CP/M operating system that Digital Research used for 8-bit microcomputers. The CP/M user interface was designed for professional programmers and other technical users.

The recent revolution in end-user computing with personal computers has made it necessary for millions of nontechnical users to become familiar with the mysteries of an operating system that is not friendly or safe enough for them. In *PC DOS Tips and Traps* we try to show you how you can make PC-DOS safer and friendlier.

One way to do this is by using the batch file facility to create your own "custom commands." The DOS batch file feature is a collection of special commands that give DOS its own programming language. They are similar to, but

not quite as powerful as, mainframe command languages like the IBM VM/CMS EXEC language. You will find batch file tips in every chapter; one is devoted exclusively to batch file programming techniques.

A custom command is a batch file program that provides a "front end" to one or more DOS commands. This gives you a more natural interface to DOS and also protects you from making some critical mistakes. We illustrate several of these commands throughout the book.

You can also use batch files to automate certain tasks, such as backing up and recovering files and setting initial conditions when you first boot your PC. Many Tips throughout the book show you how to use batch files in this way. Automating standard tasks is extremely important because it saves time and reduces the chance of error.

How the Book Is Organized

Chapters 1 and 2: "Getting Started" parts I and II. These chapters constitute a Beginner's Tutorial designed both for self-study and for training sessions. These chapters contain a collection of the Tips and Traps entries that are most helpful to beginners. Unlike most of the other chapters, the Tips and Traps here are designed to be worked through from beginning to end. They discuss booting your system, formatting diskettes, and copying diskettes.

Chapter 3: "Managing Files." Files on disks are like those in a filing cabinet. You need a good system for organizing files or you will have a mess on your hands. DOS provides you with several commands to help you organize effectively.

Chapter 4: "Managing Disks and Directories." Versions of DOS beginning with 2.0 allow you to set up what is called a "hierarchical directory" structure on a disk. This is usually most valuable on a hard disk system (IBM PC XT or equivalent). Here you will learn how to set up certain common problems. Some DOS commands permit you to deal with an entire disk as a unit. These commands, together with key manual procedures, become especially important when you use a PC XT or other hard disk system. This chapter covers the critical and troublesome topic of disk backup and recovery.

Chapter 5: "Batch Files." Many of the chapters discuss batch files related to the subject matter of those chapters. This chapter, however, concentrates on Tips and Traps that relate to the "programming" process involved in creating sophisticated batch files. As you will see, you can make DOS do some unexpected handstands knowing a few of these tricks.

Chapter 6: "Redirection, Piping, and Filters." This chapter covers the use of redirection, piping, and filters, functions brought over to DOS from UNIX and present in all versions from 2.0 on.

Chapter 7: "Controlling Peripherals." The term *peripherals* includes the display screen, printers, plotters, the keyboard, modems, and so forth. We exclude disks from this chapter since the chapter before it covers them. Here you will learn how to make peripheral devices do some unusual tricks that can greatly increase your productivity.

Chapter 8: "Advanced Topics." DOS contains several features that allow you to make significant changes in the overall environment: reconfiguring the keyboard, installing custom device drivers to control special peripherals, installing your own version of the DOS command processor, creating a virtual disk (RAM disk) in memory, and so forth. Here we introduce you to some of these more advanced techniques. Your DOS diskettes contain several utility programs, including DEBUG, which facilitates program debugging and allows you to alter any part of a program or data file. In this last chapter, we look at some helpful Tips for using this program. We finish up with a special section on the new version 3.

Other Tips and Traps Publications

PC DOS Tips and Traps is the product of a dedicated group of people at Net 1, a Walnut Creek, California, company. We specialize in gathering useful information from the PC-user community and then redistributing it in many different forms.

In addition to the other Osborne Tips and Traps books on dBASE III, AppleWorks, and Jazz, we are also preparing for the publication of regular "PC Tips and Traps" and "Apple Tips and Traps" newsletters, which will cover both Apple II and Macintosh. We solicit Tips and Traps from the user community on every topic, for our books and newsletters. We will be glad to pay for material that we publish and credit the author. For details, write to:

Net 1
1280 Boulevard Way
Suite #207
Walnut Creek, CA 94595

Net 1 also offers training sessions at various levels using our Tips and Traps books, diskettes, and newsletters. We can also train others to conduct training sessions with these materials.

1

Getting Started, Part I

The first two chapters in this book are special. They constitute a Beginner's Tutorial, and are designed for you to work through from beginning to end. If you're not new to DOS, you may want to skip to Chapter 3.

We have designed the Beginner's Tutorial so that it can be used either by a single person learning to use DOS for the first time, or in a group-training session. (The sample files we use here, along with the others in the rest of the book, are available on a diskette, which you can order by filling out the form inside the back cover.) We assume that you will turn on your PC, put *PC-DOS Tips and Traps* beside it, and follow the steps below from beginning to end.

Once you have worked through this chapter and the next one in hands-on mode, we suggest that you read Appendix A, Behind the Scenes, immediately. This will give you a good grasp of how DOS allows you to run programs and get your work done. Once you have read the Appendix, review Chapters 1 and 2 quickly; you will probably see certain things in a new light.

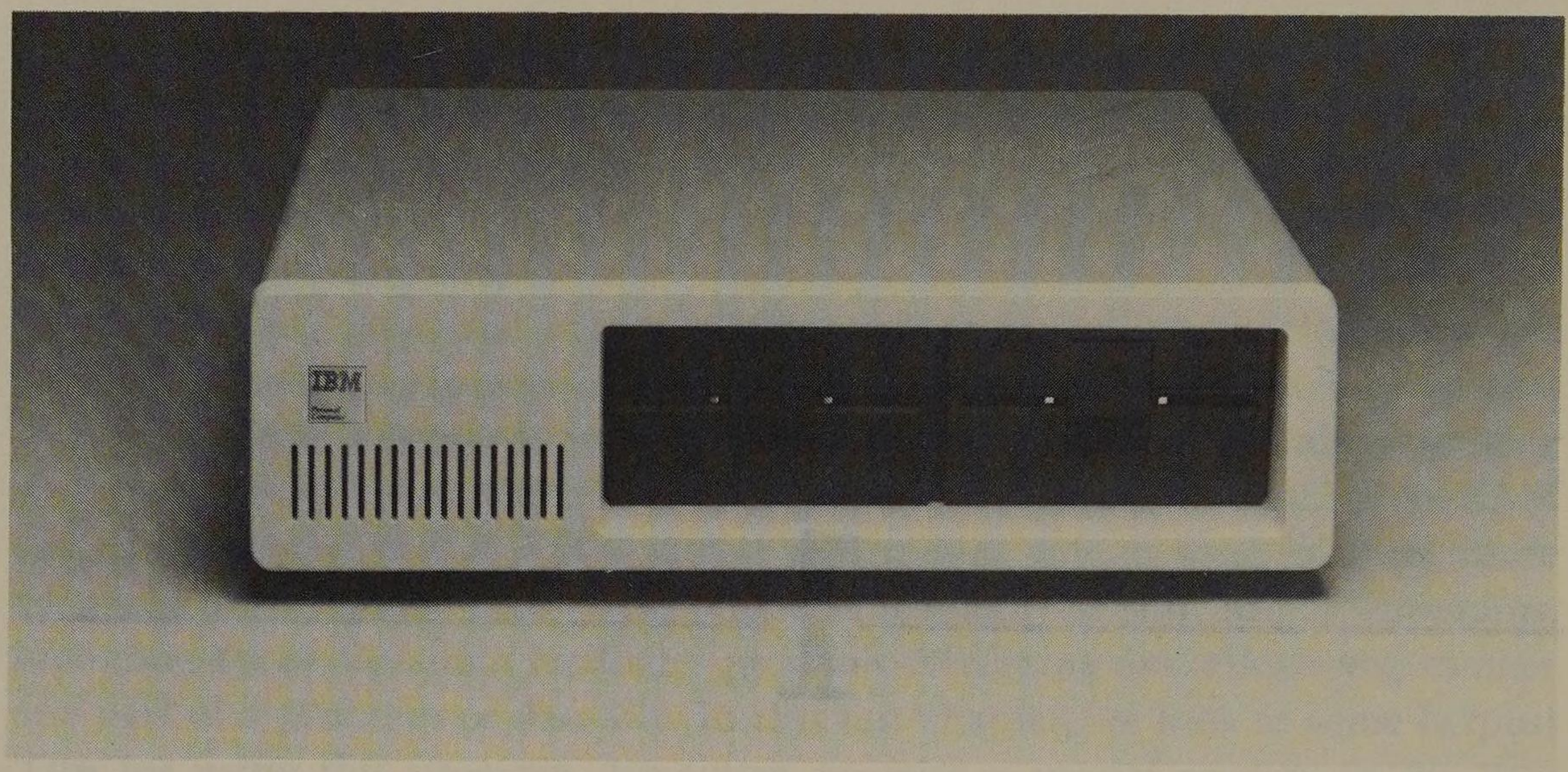


Figure 1-1. The standard IBM PC

Knowing Your PC Hardware

A personal computer is made up of various hardware components, and many different configurations are possible. Throughout this book we will refer to the two most typical IBM system configurations:

- The dual floppy system (containing two floppy disk drives), shown in Figure 1-1. This is the standard IBM PC.
- The hard (fixed) disk system (containing one floppy disk drive), shown in Figure 1-2. This is the standard IBM PC XT.

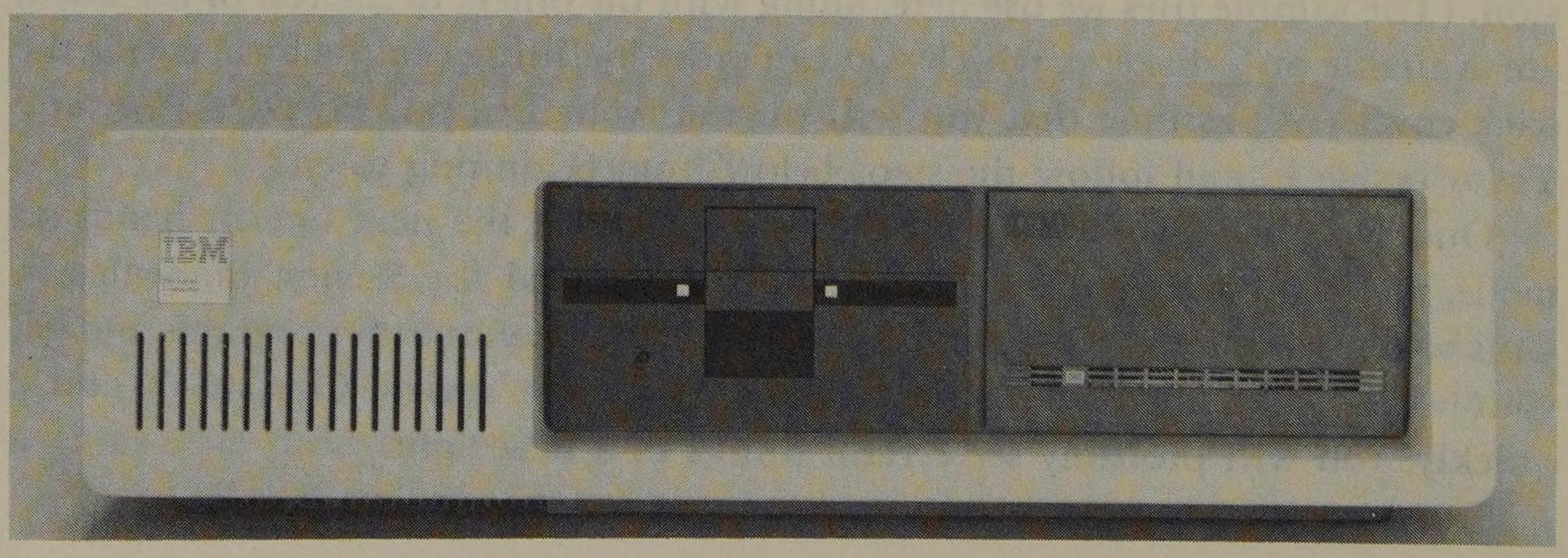


Figure 1-2. The IBM PC XT

The easiest way to identify which system you are using is to refer to these two figures. Keep in mind that there are various alternatives to the "typical configurations," so your system may look slightly different, particularly if you are using an IBM-compatible system. As we will see shortly, the disk drives allow you to read from and write to the disk media, which provide permanent storage both for programs and data.

Floppy diskettes are flexible plastic squares coated with metallic oxide that allows the write heads of the disk drives to record data magnetically on the surface of the diskette. You must handle and care for floppy diskettes carefully to avoid damaging the programs and data they contain. The box contains a list of guidelines to follow.

Do's and Don'ts
For Diskettes

DO'S:

Insert diskettes into the disk drive carefully. Diskettes are inserted with label facing up, write-protect notch on the right, and the oblong "access hole" inserted first. If inserted properly, the diskette will slide into the disk drive with little or no resistance, and the disk drive door can be closed without force. If you have difficulty, take out the diskette and try it again. You can damage the diskette or the disk drive, or both, if you use force.

Store diskettes in their envelopes. When you are not using it, store the diskette in the protective envelope that came with it. This helps to protect the sensitive access hole from dust and other contaminants that could cause permanent physical damage and loss of data.

Store diskettes in an appropriate temperature range. Excessive cold (below 50 degrees Fahrenheit) or heat (over 100 degrees Fahrenheit) could damage your diskettes. Be careful not to leave your diskettes in your car on warm summer days; they can warp from the heat. If your diskettes are subjected to extremely cold temperatures, let them warm to room temperature before using them. Failure to do so could result in damage to your disk drive.

Label your diskettes. In the beginning you will have just a few diskettes to keep track of. However, it won't be long before you have a large library of diskettes, perhaps well over a hundred. Imagine trying to locate a particular

continued

diskette in a group of 100 identical diskettes without labels! Take the extra minute to apply a label, and save yourself the irritation of having to load each diskette to determine its contents. Write out the label *before* affixing it to the diskette, or use a *felt-tip* pen if you need to modify a label that is already on a diskette. The pressure of writing on the label could damage your diskette.

DON'TS:

Don't touch the sensitive access hole. A diskette consists of a Mylar disk coated with a magnetic medium that is very sensitive to touch; the moisture from your fingers can interfere with the disk drive's ability to read and write information on the diskette. The disk is permanently encased in a protective "jacket," with an oblong opening, or access hole, for the recording head to gain access.

Keep your diskettes away from magnetic fields. Take inventory of the desktop items that are in proximity to your computer system. Some of them may be hazardous to the health of your diskettes: magnetic paper clips, magnetic clocks, and magnetic bulletin boards, to name just a few. If you travel with your diskettes, request that airport security hand-check your box of disks. The metal-scanning devices could possibly erase some data from your diskettes.

Don't feed your diskettes. Food and liquids can kill your diskettes. If you spill coffee on a diskette, a residue will remain that not only renders your diskette useless but could also permanently damage your disk drive. If food particles find their way into the access hole, the diskette surface will be scratched.

Don't bend your diskettes. A crease in your diskette could result in the recording head "jumping" over the crease as it reads your data, and you will lose information.

A hard disk is a recording medium that is usually enclosed permanently in a cabinet with the hard disk drive itself. A typical single hard disk can hold at least 30 times more data than a typical floppy diskette can hold.

Figure 1-3 is a schematic diagram of the main components of a personal computer system. The "brain" of the system is the central processing unit (CPU), which executes the programs that help you to do your work. You communicate

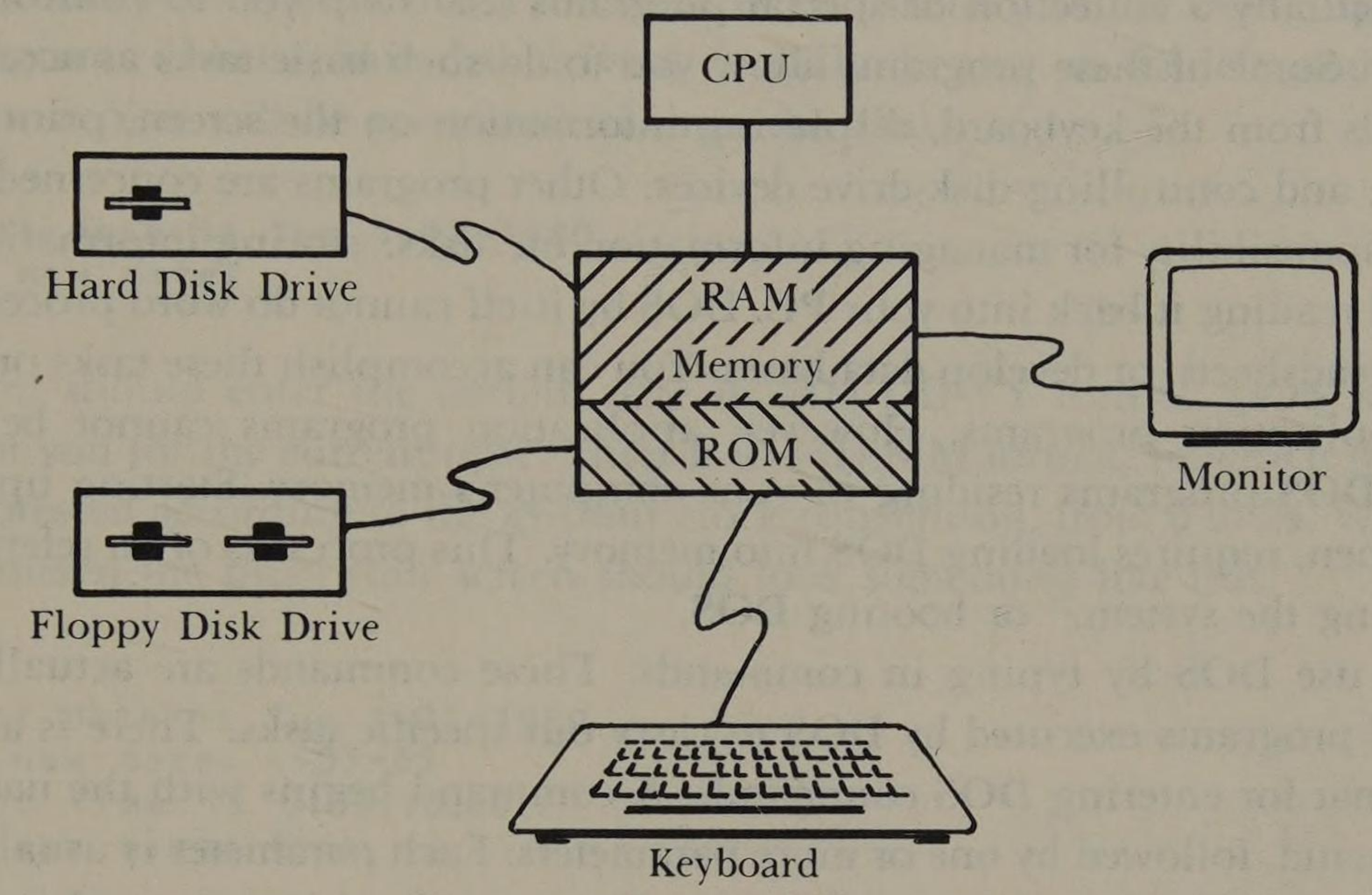


Figure 1-3. Components of a PC system

with the CPU primarily through the keyboard. The CPU communicates with you primarily by displaying information on the screen.

Just like a human being, a computer has memory. There are two types of memory. Read-only memory (ROM) is "nonvolatile"; that is, it retains its contents even after you turn off the machine. Random access memory (RAM) is normally volatile; its contents (both program and data) are lost whenever you turn off the machine. The IBM PC and some compatible systems come with a subset of the BASIC language program already in ROM. In general, however, if you want to run a program, you have to load it into RAM, together with the data on which you want it to operate.

You load programs and data into memory from the disks on which they are stored. As Figure 1-3 shows, they may be stored on both floppy diskettes and hard disks.

In the following sections, you will use DOS to help you direct all the hardware components we have mentioned to perform your business tasks.

How DOS Works

DOS is actually a collection of special programs that help you to control your computer. Some of these programs allow you to do such basic tasks as accepting commands from the keyboard, displaying information on the screen, printing it on paper, and controlling disk-drive devices. Other programs are concerned with DOS's responsibility for managing information on disks: storing information on disks and reading it back into your PC. DOS by itself cannot do word processing, create spreadsheets, or develop data bases. You can accomplish these tasks only by using application programs. However, application programs cannot be used without DOS programs residing in your computer's memory. Starting up your system, then, requires loading DOS into memory. This process is often referred to as "booting the system," or booting DOS.

You use DOS by typing in commands. These commands are actually the names of programs executed by DOS to carry out specific tasks. There is a standard format for entering DOS commands. A command begins with the name of the command, followed by one or more parameters. Each parameter is usually the name of a disk file or a group of disk files. For example, this command creates a backup copy of the file MYDATA:

```
COPY MYDATA MYBACKUP
```

After you have typed a command, you press the ENTER key to indicate that you want DOS to accept the command and carry it out. In what follows, we will introduce you to several DOS commands. As you will see, all follow the same general format.

Booting Your System

As we mentioned earlier, starting your computer and getting DOS going is called booting the system. You will need to do this every time you turn on the computer. You may also need to boot your system occasionally if a "bug" in a program causes your keyboard to freeze up.

1.01 Tip: *If you are using a hard disk system, you must make sure that the hard disk is formatted before you proceed.*

This has probably been done already, since dealers generally do it as part of the

initial setup of your new system. If they haven't done so, follow the procedure outlined in Chapter 4, "Managing Disks and Directories."

To boot your system, insert the main DOS diskette (not the DOS Supplemental Programs diskette) into disk drive A (usually the one on the left) and turn on the power. You will be prompted to enter the date.

```
Current date is Tue 1-01-1980
Enter new date:
```

You should enter the current date in MM-DD-YY format. DOS will then prompt you for the current time. Enter it in HH:MM format, in which the hours are expressed according to the 24-hour clock convention, from 0 to 24. When you have entered the time, your screen should look something like this:

```
Current date is Tue 1-01-1980
Enter new date: 4-03-85
Current time is 0:00:10.60
Enter new time: 4:44

The IBM Personal Computer DOS
Version 2.10  (C) Copyright IBM Corp 1981, 1982, 1983

A>
```

1.02 Trap: *If you do not respond to the date and time prompts, your files will be stamped with the date and time your version of DOS was created.*

Since DOS always notes the date and time when a file is saved, it will apply what is called a default entry if you do not respond. The date and time DOS was created will be entered, making it impossible for you to use this information to decipher when a file was created or last modified. Shown below is what you will see on your screen if you bypass the date and time prompts by pressing the ENTER key:

```
Current date is Tue 1-01-1980
Enter new date:
Current time is 0:00:10.60
Enter new time:

The IBM Personal Computer DOS
Version 2.10 (C)Copyright IBM Corp 1981, 1982, 1983
```

Here is a list of files that were created on a day when date and time entries were not made during the booting process:

```
A>DIR

Volume in drive A has no label Directory of A:\

FILE1      WKS       1280    1-01-80   12:05a
FILE2      WKS       4210    1-01-80   12:05a
FILE3      WKS      11264    1-01-80   12:05a
FILE4      WKS       3652    1-01-80   12:05a
FILE5      WKS      14658    1-01-80   12:05a
     5 File(s)     326196 bytes free
```

Here is another list. These files were created with the correct date and time stamp.

```
A>DIR

Volume in drive A has no label
Directory of A:\

FILE1      WKS       2115    4-07-85    2:36p
FILE2      WKS      18442    7-23-84    4:53p
FILE3      WKS      41283    7-12-84    2:03a
FILE4      WKS      66270    4-10-85    2:37p
FILE5      WKS      10280    4-15-85    5:48p
MYFILE     WKS      22860    4-24-85   14:22p
     6 File(s)     200010 bytes free
```

The difference in the usability of the two lists should be obvious. To avoid having to enter the date and time, you can install an electronic clock-calendar in your system so that the current date and time is automatically passed to DOS every time you boot your system. The importance of the date and time stamp is such that we highly recommend adding this hardware feature.

After you make the date and time entries, DOS displays a prompt telling you that the operating system is waiting for you to issue a command.

When you are using a dual floppy system, you will see this prompt:

```
A>
```

For a hard disk system, the prompt will be

```
C>
```

The prompt also tells you which disk drive DOS considers to be the current default drive; that is, the one DOS will automatically go to when looking for files or commands. A default drive is in effect unless you specify otherwise. You can change the default drive by entering the drive letter of the new default drive, followed by a colon, followed by ENTER.

On a dual floppy system:

```
A>B:

B>
```

On a hard disk system:

```
C>A:

A>
```

1.03 Trap: *If you get BASIC rather than DOS on the screen when you boot a floppy system, you have either the wrong diskette or no diskette in drive A, or something is wrong with your DOS diskette.*

You are in BASIC if your screen looks like Figure 1-4.

Check the diskette in drive A. If it is not a DOS diskette, switch it for one, turn off your system, and then start it up again. Should you discover that the diskette is in fact a DOS diskette but will not start up the system, you may have a

```
The IBM Personal Computer Basic
Version D2.10 Copyright IBM Corp. 1981, 1982, 1983
61327 Bytes free

Ok


F1-help F2-save F3-new file F4-import data F9-expand  F10-contract Esc-exit
```

Figure 1-4. BASIC startup screen

bad copy of DOS. If this is the case, you will have to check with your dealer about getting a new copy.

1.04 Trap: *If your floppy system gives you a message that a disk error has occurred, the wrong diskette may be in drive A.*

If you get a message like this:

```
Non-system disk or disk error
Replace and strike any key when ready
```

your system is telling you that the wrong diskette is being used to start up the system. Follow the procedure in the previous Trap, checking first the diskette in drive A.

1.05 Trap: *If you do not close the drive A door securely after inserting the DOS diskette, your system will not boot properly.*

In this case, the drive A red light will probably go on and stay on, indicating that the machine is attempting unsuccessfully to read DOS into memory. Do not remove the DOS diskette! If you do so, you may damage it. Turn off the machine, reinsert the DOS diskette, and close the door of drive A securely. Then turn on the power again. The system should now boot correctly.

1.06 Trap: *Never remove or insert a diskette in a drive when the red light for that drive is on.*

Failure to heed this warning could result in permanent damage to your diskette. You wouldn't try to remove a tape cassette while your unit is playing, would you? Need we say more?

1.07 Trap: *If you get a blank screen when you boot, you have one of several problems.*

Check your monitor first:

- Is it plugged in?
- Is it turned on?
- Are the brightness and contrast knobs adjusted correctly?
- Check the cable between your monitor and the computer. Is it secure? Is it connected to the proper output jack?

If these functions are working, then try to start your system again by turning it off, waiting approximately 15 seconds, and turning on the power again.

If you still get a blank screen, call your dealer and arrange to have your system checked. You probably have a problem with the system unit, and only a reputable service agent should attempt to fix it.

Formatting and Copying Disks

1.08 Tip: *Before you start using your DOS diskette (or any copyable program diskette), it is a good idea to make a backup copy for daily use, and to store the original.*

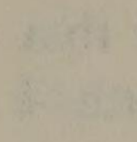
Before we can go on, it is important to discuss briefly the concept of a file. Information is stored on disks as files. Files can consist of either data or programs. Address lists, correspondence, accounting records, and spreadsheets are just a few examples of typical data files. A program file might contain a word-processing program, a data base program, a spreadsheet program, or even a game.

Before you can use any blank diskette for the first time, it must be formatted with the FORMAT command. This command checks for physical damage on your diskette and creates a directory so that DOS knows where your files are stored. Once the diskette is formatted, you can store information on it. (A diskette that comes to you with something already on it—for example, a program diskette—has already been formatted. You should never format such a diskette; if you do, you will erase whatever it contains.)

If you have a dual floppy drive system, be sure that you begin the procedure below with your DOS diskette in drive A and a blank diskette in drive B. If you are using a system with a hard drive, you will follow the same procedure from your C> prompt, except that you will start with a blank diskette still in its jacket

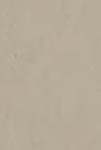
and insert it only when DOS prompts you to do so. If you have only one floppy drive on your hard disk system, DOS considers that the one physical floppy drive is both A and B. Consequently, executing the COPY command as shown below will result in DOS prompting you to change diskettes as each individual file is copied. This is extremely tedious. You should do it once just to see how it works, but in the future you may want to use a temporary area of the hard disk as an intermediate transfer area for copying from one floppy diskette to another. We will cover this technique later in this chapter. Now, enter the FORMAT command:

```
A>FORMAT B:/S

Insert new diskette for drive B: and strike
any key when ready

Formatting...Format complete System transferred

     362496 bytes total disk space
     40960 bytes used by system
     321536 bytes available on disk

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



1.09 Tip: *When you use the FORMAT command, you need to decide whether you will be using that disk to start up your system.*

To create a backup copy of DOS with which you can boot your system, you must first format the blank disk as a system disk, as shown above, by using the /S option. This process not only formats the disk but also places two special DOS files at the beginning of the disk. These files are "hidden" and are generally invisible to you, but they are an essential part of DOS (see Appendix A, Behind the Scenes).



1.10 Trap: *The FORMAT command is a potentially dangerous command. One mistake could result in wiping out all the information stored on your floppy diskette or hard disk!*

When using the FORMAT command, be very careful to specify the correct drive. The FORMAT procedure begins by erasing everything from the drive you specify. This process is irrevocable—all of your information is lost forever.

Now that your diskette is formatted, you can use the COPY command to duplicate the DOS files.

Making a backup copy of your DOS diskette involves duplicating all the DOS files. By grouping files together when using COPY, you can complete this process with a single command. This approach involves wildcard characters, which are two special characters you can use in the file name and/or extension to specify more than one file. The question mark (?) is used to match a single character at a specific position, while the asterisk (*) refers to any number of characters at a specified location. The following command will duplicate all DOS files, regardless of name or extension:

```
A>COPY A:*.* B:
COMMAND.COM
ANSI.SYS
FORMAT.COM
CHKDSK.COM
SYS.COM
DISKCOPY.COM
DISKCOMP.COM
COMP.COM
EDLIN.COM
MODE.COM
FDISK.COM
BACKUP.COM
RESTORE.COM
PRINT.COM
RECOVER.COM
ASSIGN.COM
TREE.COM
GRAPHICS.COM
SORT.EXE
FIND.EXE
MORE.COM
BASIC.COM
BASICA.COM
       23 File(s) copied
```

When you issue the COPY command on a dual floppy system, you will be prompted by DOS to insert the *Original* diskette (DOS diskette) in drive A first. After DOS reads the diskette, you will be prompted to insert the *Target* diskette (the one to be copied) in drive B. If you have a hard disk system with only one floppy drive, the entire process will take place in the one physical drive, even though DOS will be referencing both drives.

```
A>COPY *.* B:

COMMAND.COM

Insert diskette for drive B: and strike
any key when ready

Insert diskette for drive A: and strike
any key when ready
```

```
ANSI.SYS

Insert diskette for drive B: and strike
any key when ready

Insert diskette for drive A: and strike
any key when ready

      ......

  23 File(s) copied
```

1.11 Tip: *If you have a hard disk system, copy DOS to the hard disk (drive C), so you can boot the system and have access to DOS commands without having to use your floppy diskette.*

When you have copied DOS onto your hard disk, you can start up your system by simply turning on the power (first make sure that no diskette is in drive A). The only difference in command format in copying DOS files to a hard disk, instead of a floppy, is the drive designation. To start this procedure, switch to drive A with the following command:

```
C>A:
```

You then issue the COPY command:

```
A>COPY *.* C:
```

1.12 Trap: *The DOS manual instructs you to make a backup copy of DOS, using the DISKCOPY command. This can be a quick and easy way to make an exact duplicate, but you should be aware of potential problems.*

DISKCOPY is a command that reads all the formatting and information from one diskette and creates an exact duplicate on another diskette. If either the diskette you are copying from or the one you are copying to have any bad sectors, then the DISKCOPY command will create an ineffective copy (see Appendix A, Behind the Scenes). There are times when DISKCOPY cannot complete the process successfully, and none of the files are copied.

2

Getting Started, Part II

In the last chapter, we stepped our way through the first part of our Beginner's Tutorial, covering some basic information about your personal computer and PC DOS, as well as what is involved in booting the system.

In this chapter, we will look first at the DOS commands that help you to load and use programs. We'll also learn how files are organized and how to get a listing of the files in a particular subdirectory. We will examine the extremely important question of backing up files, and begin to look at how you can use DOS to automate your procedures.

Using Applications Programs

Since you did not buy your computer just to make copies of DOS, it is time to get you started using an application program. Lotus 1-2-3 is an electronic spreadsheet program that has been the top-selling IBM software program ever since its release in 1982. Because of its popularity and tremendous capability, we will use it here to show you what you need to do to install and run programs under DOS.

Before you can use a program, you must usually complete a series of tasks that will prepare the program disk(s) you will use. This is often referred to as installing a program, and we will follow the steps for installing Lotus 1-2-3 to demonstrate this procedure. Even if you do not intend to use 1-2-3, you will find it helpful to read through this section, because the installation of 1-2-3 is in many ways similar to the installation of other programs.

Installing a Program

Before you can install a program, you may need to know the answers to any or all of the following questions:

- What kind of display monitor are you using, color or monochrome?
- Do you have a graphics display card installed?
- What are the name and model number of your printer?
- Which version of DOS are you using?
- Is your computer a dual floppy or hard disk system?

Here is an outline of the tasks you need to perform during the installation procedure:

- Transfer DOS files and 1-2-3 files:
- Dual floppy system: DOS files are transferred to the 1-2-3 program diskette so that it can be used to boot the system.
- Hard disk system: 1-2-3 program files are transferred to the hard disk so that you can run the program from the hard disk.
- You may also need to make 1-2-3 compatible with the various components of your system.

A personal computer system is not a fixed, standardized set of equipment. It is similar to a component stereo system in that you can select hardware components to suit your needs. To establish compatibility, you may need to install what are called "drivers" to fit certain components of your particular system such as your monitor and your printer. Appendix A, Behind the Scenes, will help you to understand the significance of this step.

Follow the step-by-step procedure outlined below to install the Lotus 1-2-3 program for a dual floppy system. Screen prompts help to guide you through the

process. Before you type in the first command, be sure that DOS is in drive A, and the 1-2-3 System Disk is in drive B.

```
A>B:INSTALL

A>rem   *-----------------------------------------------------
A>rem  *    DOS INSTALLATION for 1-2-3 SYSTEM DISK     *
A>rem   *-----------------------------------------------------
A>rem  *    Make sure that:          *
A>rem  *    (1) A DOS 1.10 or DOS 2.00 disk is in Drive A.
A>rem  *    (2) The 1-2-3 SYSTEM DISK is in Drive B.          *
A>rem  *                *
A>rem  *    If this is NOT so, then terminate this batch job now:
A>rem  * - Hold [Ctrl] and press "C". Then press "Y".          *
A>rem  *    Otherwise . . .          *
A>rem   *-----------------------------------------------------
A>pause
Strike a key when ready . . .
```

After you strike a key, the install program will automatically issue the next two commands.

```
A>sys  b:
System transferred
```

The "system" refers to two DOS files we mentioned in Chapter 1 that represent the all-important link between you and your system. These files include the instructions needed to perform such tasks as translating keyboard input, creating screen displays on the monitor, and interacting with the disk drive. Because of their importance, these files are hidden from the user and cannot be accessed. You can learn more about these files in Appendix A, Behind the Scenes.

This next command copies the command processor file to the 1-2-3 System Disk. It is this file that starts up your system and gives you the date and time prompts.

```
A>copy    COMMAND.COM    B: 1 File(s) copied
```

Before going on to the next step, you need to know what kind of monitor

configuration you are using. The following is a list of possibilities recognized by the 1-2-3 program:

Monochrome display (MONO)

Color display (COLOR)

Monochrome with graphics capability (HERCULES)

Both monochrome and color monitors (BOTH)

Select the appropriate reference from the list shown above to issue the last command. The drive designation (B:) is that of the target drive, where the 1-2-3 System Disk should be, and the 1-2-3 Utility Disk needs to be in drive A. Drivers will be installed on all of the 1-2-3 program diskettes, so you will be prompted to switch disks during the process. For example, if you have a color monitor, you would issue the command COLOR B: as follows:

```
A>COLOR B:
A>rem    *-----------------------------------------------------

A>rem          "COLOR"  DRIVER INSTALLATION

A>rem    *-----------------------------------------------------

A>rem    *      Be sure that you have read GETTING STARTED.    *

A>rem    * Make sure that the source drive is the "default  drive"

A>rem    *            and that "B:" is the target drive.

A>rem    *

A>rem    *   If this is NOT so, then terminate this batch job now:

A>rem    *            - Hold [Ctrl] and press "C"          *

A>rem    *            - Press "Y"

A>rem    *   Otherwise . . .

A>pause  *-----------------------------------------------------
Strike a key when ready . . .

A>copy     IBM0COLO.DRV    TD.DRV
      1 File(s) copied

A>copy     IBM1G2.DRV      GD.DRV
      1 File(s) copied

A>copy     IBM2KB.DRV      KB.DRV
      1 File(s) copied

A>copy     IBM3PR.DRV      PR.DRV
      1 File(s) copied
```

The appropriate driver files have been selected from the 1-2-3 Utility Disk. The next step involves copying these files to your 1-2-3 System Disk. The process continues automatically.

```
A>continue  COLOR B:

A>rem    *-----------------------------------------------
A>rem    *  TWO-DISKETTE INSTALLATION  *  HARD-DISK INSTALLATION
A>rem    *-----------------------------------------------
A>rem    *                            *                  *
A>rem    *          Place your        * The next step  will  *
A>rem    *      1-2-3 SYSTEM DISK     * copy the drivers to  *
A>rem    *        in drive "B:".      * drive "B:".          *
A>rem    *                            * Then, you will be    *
A>rem    *                            * instructed to  terminate
A>rem    *                            * this batch job.      *
A>rem    *                            *                  *
A>pause  *-----------------------------------------------
Strike a key when ready . . .

A>copy     ??.DRV    B:
TD.DRV
GD.DRV
KB.DRV
PR.DRV
      4 File(s) copied
          ...
```

The 1-2-3 System Disk is now installed and ready to use. (The driver files need to be copied to the 1-2-3 Printgraph Disk as well. The screen will prompt you to replace the System Disk with the Printgraph Disk, and then the files will be transferred.)

Directories and Subdirectories

Before we outline the installation procedure for a hard disk system, we need to talk about the hard disk environment. Even high-density floppy diskettes hold less than 1,000,000 bytes of data. Rarely can you get more than 15 or 20 files on a floppy diskette before running out of space. A hard disk can usually store 10 or 20

million bytes, making it possible to store hundreds of files. The problem with storing this many files is that finding a particular file becomes very difficult. It would be like throwing all your papers into one big file folder without organizing them into separate categories. To solve this problem, DOS 2.0 and later versions incorporate a tree-structured directory scheme that allows you to organize files into various subdirectories (see Figure 2-1). In the following Tips and Traps, we will illustrate the basics of how to create and use these subdirectories. (See Chapter 4, Managing Disks and Directories, for more information.)

2.01 Tip: *If you have a hard disk system, create a subdirectory for each program, and a second subdirectory for data files for that program.*

Using the DOS command MKDIR (abbreviation for MAKE DIRECTORY), you can create a subdirectory. Suppose we want to create a subdirectory for Lotus

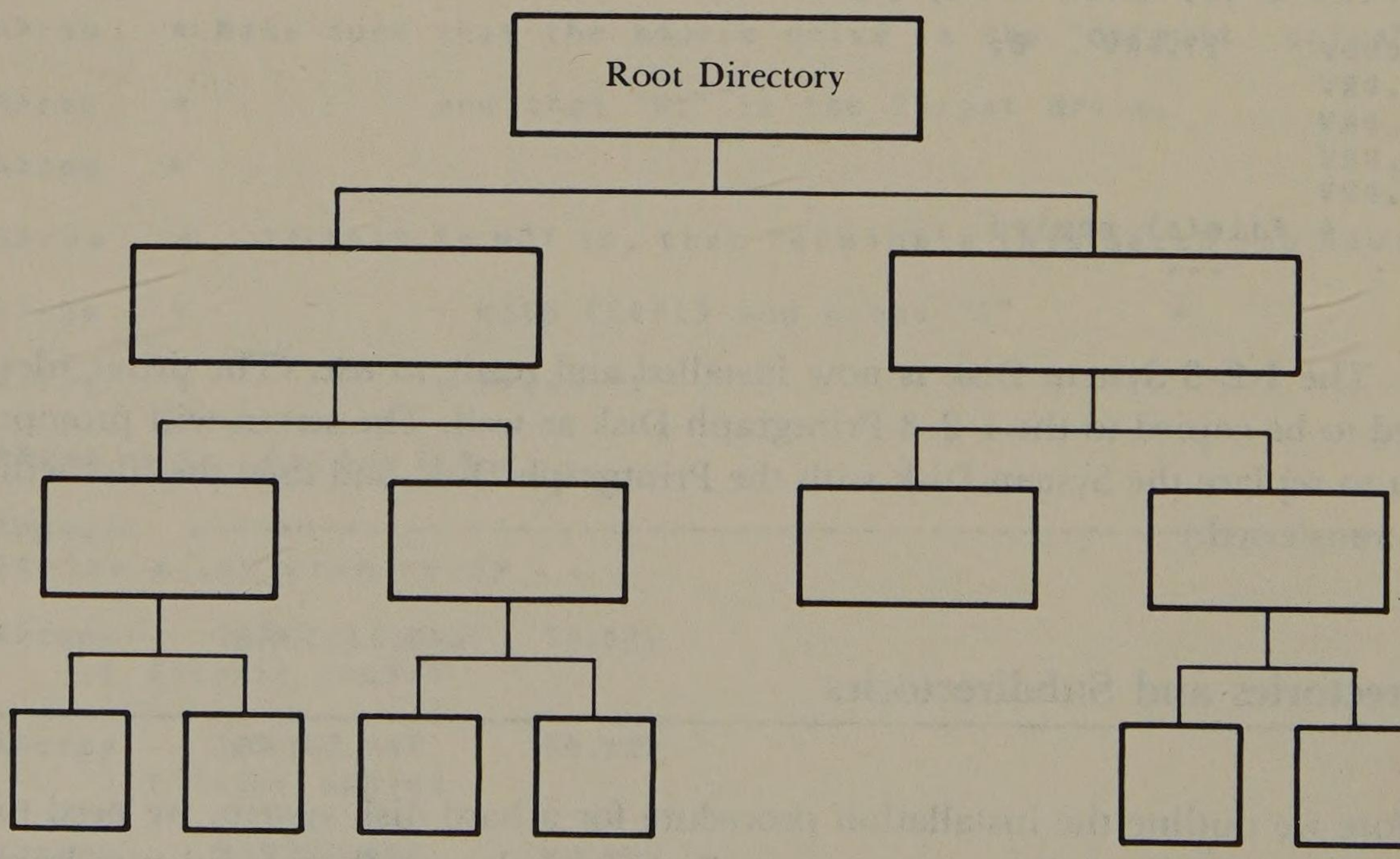


Figure 2-1. Directories and subdirectories

1-2-3 program files. We might give it the name LOTUS by executing this command:

```
MKDIR\LOTUS
```

The MKDIR is the command name. The " \" (backslash) refers to the root directory. The root directory has already been created for you; it is the area of the disk onto which you copied DOS in Chapter 1. (It's called a "root directory" because a directory structure like the one shown in Figure 2-1 can grow from it.)

2.02 Tip: *If you are using a hard disk system, you should copy each application program to the hard disk before using it.*

Follow the instructions below to install 1-2-3 on your hard disk. If you have not already done so, create a LOTUS subdirectory. (See the previous Tip.)

```
C>MKDIR\LOTUS
```

The initial phase of the installation procedure involves copying all 1-2-3 files to the subdirectory on your hard disk.Three program diskettes are involved: Lotus System, Printgraph, and Utility.

This first step uses the COPY *.* command to copy all of the files from the 1-2-3 System Disk to the LOTUS subdirectory. Before you type in this command, be sure that the 1-2-3 System Disk is in drive A.

```
C>COPY A:*.*
A:123.EXE
A:123.HLP
A:123.CNF
A:LOTUS.COM
A:LTSLOAD.COM
A:AUTOEXEC.BAT
A:INSTALL.BAT
A:TD.DRV
A:GD.DRV
A:KB.DRV
A:PR.DRV
      11 File(s) copied
```

Use the COPY *.* command again, with the Printgraph Disk in drive A.

```
C>COPY A:*.*
A:GRAPH.EXE
A:GRAPH.HLP
A:GRAPH.CNF
```

```
A:LOTUS.COM
A:LTSLOAD.COM
A:LOTUS.DLB
A:BLOCK1.FON
A:BLOCK2.FON
A:ITALIC1.FON
A:ITALIC2.FON
A:ROMAN1.FON
A:ROMAN2.FON
A:SCRIPT1.FON
A:SCRIPT2.FON
A:AUTOEXEC.BAT
A:INSTALL.BAT
A:TD.DRV
A:GD.DRV
A:KB.DRV
A:PR.DRV
      20 File(s) copied
```

Use the COPY *.* command for a third time. The Utility Disk should be in drive A this time.

```
C>COPY A:*.*
A:LOTUS.COM
A:LTSLOAD.COM
A:FILEMGR.COM
A:TRANSLAT.COM
A:VCWKS.EXE
A:DIFWKS.EXE
A:WKSDIF.EXE
A:DBFWKS.EXE
A:WKSDBF.EXE
A:IBMOHERC.DRV
A:IBMOMONO.DRV
A:IBMOB&W.DRV
A:IBMOCOLO.DRV
A:IBM1HERC.DRV
A:IBM1G1.DRV
A:IBM1G2.DRV
A:IBM2KB.DRV
A:IBM3PR.DRV
A:CPQOTD.DRV
A:BOTH.BAT
A:MONO.BAT
A:B&W.BAT
A:COLOR.BAT
A:HERCULES.BAT
A:COMPAQ.BAT
A:CONTINUE.BAT
A:AUTOEXEC.BAT
A:INSTALL.BAT
A:FIXDOS.COM
A:TD.DRV
A:GD.DRV
A:KB.DRV
A:PR.DRV
      33 File(s) copied
```

This next phase of the installation process incorporates some of the DOS utilities with the 1-2-3 functions to form a single, easy-to-use umbrella, referred to as the Lotus Access System. (Figure 2-2 shows the Lotus Access System startup screen.) While running the 1-2-3 program, you will be able to employ several DOS functions, such as formatting new disks and copying files from one disk to another. The 1-2-3 program will need to access the appropriate DOS file when you issue one of these commands and will look for it in the LOTUS subdirectory.

Insert your DOS diskette in drive A, and issue the following commands one at a time to copy the necessary DOS files to the LOTUS subdirectory.

```
C>COPY A:COMMAND.COM
      1 File(s) copied

C>COPY A:FORMAT.COM
      1 File(s) copied

C>COPY A:CHKDSK.COM
      1 File(s) copied

C>COPY A:DISKCOPY.COM
      1 File(s) copied

C>COPY A:DISKCOMP.COM
      1 File(s) copied
```

The next and last step installs the 1-2-3 drivers. Before issuing this last command, refer to the dual floppy version of the installation procedure for an

```
Lotus Access System  V.1A  (C)1983 Lotus Development Corp.              MENU
------------------------------------------------------------------------------
1-2-3  File-Manager  Disk-Manager  PrintGraph  Translate  Exit
Enter 1-2-3 -- Lotus Spreadsheet/Graphics/Database program
==============================================================================

                              Tue  03-Apr-86
                                4:44:06am

Use the arrow keys to highlight command choice and press [Enter]
Press [Esc] to cancel a choice; Press [F1] for information on command choices
```

Figure 2-2. Lotus Access System startup screen

explanation of the available drivers (based on the type of monitor you are using). You will need first to change the drive designation, from C: to A:.

```
C>A:

A>COLOR C:

A>rem    *-----------------------------------------------
A>rem      "COLOR" DRIVER INSTALLATION
A>rem    *-----------------------------------------------
A>rem    *       Be sure that you have read GETTING STARTED.
A>rem    * Make sure that the source drive is the "default  drive"
A>rem    *       and that "C:" is the target drive.
A>rem    *
A>rem    *   If this is NOT so, then terminate this batch job now:
A>rem    *    - Hold [Ctrl] and press "C"
A>rem    *    - Press "Y"
A>rem    *   Otherwise . . .

A>pause  *-----------------------------------------------
* Strike a key when ready . . .
```

Follow the screen prompts to complete the installation process.

2.03 Trap: *If you do not put a write-protect tab on the original program diskette before copying it, you are courting disaster.*

There is a way to protect important diskettes whose contents you do not want written over or erased. Most diskettes have a small rectangular notch cut out of one side. You can write to a diskette only when this notch is uncovered. Some diskettes, like the DOS diskette, do not have a notch and therefore can never have new information written on them. Notched diskettes can be protected by covering the notch with one of the adhesive-backed tabs that are supplied with each box of new diskettes. If you attempt to write on a diskette that is so protected, DOS will display the following error message:

```
Write Protect Error Writing Drive
Abort, Retry, Ignore?
```

If you are going to copy DOS onto your program diskette, you must make sure it is not write-protected while you execute this copy (see Chapter 1). Once you have completed the copy, it is a good idea to put on a write-protect tab. This protects you from inadvertently formatting or copying over a permanent program diskette.

2.04 Tip: *On a floppy system, be sure that you always have a formatted diskette available when you are using a software program.*

Whenever you get a new box of blank diskettes, it is a good idea to format all of them to ensure that you will have a "usable" diskette when you need to save data. When you want to create files with an application program, like a spreadsheet program, you normally must have a formatted diskette on hand because most application programs do not let you execute FORMAT and other DOS commands without leaving the program. Not being prepared in this way could result in having to abort a procedure without saving valuable data.

This is not a problem on a hard disk system unless you are close to running out of space on your hard disk.

2.05 Trap: *On a floppy system, your DOS disk must normally be in the default drive (typically, drive A) when you issue the FORMAT command.*

DOS can only carry out those commands for which it can access the program files. Some DOS commands are subprograms within the main DOS command processing program (normally resident in memory), while others remain on the DOS diskette. The FORMAT command is an external file, which means that your DOS diskette must be in the default drive (usually drive A), so that DOS can retrieve the necessary information for implementing the formatting procedure. If the FORMAT program file cannot be found, then DOS will respond with the following message:

```
Bad command or filename
```

DOS is telling you that it was unable to locate the program file required to execute the command you issued. Of course, if you misspelled the name of the command, then you will also see this message, because DOS will be looking for a file with the name as you typed it.

Running a Program

2.06 Tip: *To start a program, simply type the program's name in response to the DOS prompt.*

To implement 1-2-3, for example, you type LOTUS at the DOS prompt.

```
A>LOTUS
 or
C>LOTUS
```

The red light on your disk drive will go on, and you will hear your disk drive working. The Lotus Access program will be read into memory, and in just a few seconds, you will see the main Lotus menu on the screen.

2.07 Trap: *If you get the "Bad command or filename" message after entering a program name, either you have misspelled the program name or it is not on the disk pointed to by the current default drive/directory combination.*

Suppose that you have your 1-2-3 program on your hard disk (drive C), but have set the current default drive to A. If you issue the command LOTUS in hopes of loading 1-2-3, you will get a "Bad command or filename" message. What you probably want to do is to enter a C: in response to the DOS prompt; this will change the default drive to C. It would also work to issue the command C:LOTUS, thus telling DOS to look for the program on drive C.

If you have a hard disk system or are using version DOS 2.0 or greater on a floppy system, you may have created one or more subdirectories on your disk. In this case, the current default drive could be correct but the current subdirectory is not. The cure is to change the subdirectory using the Change Directory (CD) command. In our example, the command would look like this:

```
CD C:\LOTUS
```

2.08 Trap: *On a hard disk system, if you get the DOS prompt or a strange message after entering a program name, the program may be copy protected.*

```
                    WEEKLY EXPENSE REPORT

                  MON      TUE      WED      THU      FRI      TOTAL
                -------  -------  -------  -------  -------  -------
Airfare
Auto Rental
Meals:
        Breakfast
        Lunch
        Dinner
Hotel
Miscellaneous
                =======  =======  =======  =======  =======  =======
            TOTAL
```

Figure 2-3. Sample spreadsheet

Generally this means that you can only copy the program to the hard disk (not to another floppy diskette), and that the original program diskette must be in drive A when you start the program from the hard disk.

A special note for IBM AT users: For some programs on the IBM AT (such as Symphony), you must insert the program diskette in drive A and then issue a command like DIR A: so that DOS can determine what kind of diskette it is dealing with—high-density 720K bytes or low-density 360K bytes. You can then enter the name of the program, and DOS should load it successfully.

After you load 1-2-3, create a simple spreadsheet like the one shown in Figure 2-3, or develop your own application. Save this spreadsheet as a file, using the 1-2-3 /File Save command. If you are not already familiar with 1-2-3 commands, refer to the 1-2-3 manual, or some other available resource.

Once you have saved your file, return to DOS by issuing the 1-2-3 /Quit command.

Listing the Files in a Subdirectory

The DIR command displays a listing of the files that have been saved in a particular disk subdirectory. You also get descriptive information about your files, including the size of each one, and the date and time the file was created or last changed.

2.09 Tip: *Use the DIR command to get a list of the files in a subdirectory.*

If you are using a dual floppy system, insert DOS in drive A, and issue the DIR command. Your screen should look something like this:

```
A>DIR

 Volume in drive A has no label
 Directory of  A:\
 COMMAND   COM    17792  10-20-83   12:00p
 ANSI      SYS     1664  10-20-83   12:00p
 FORMAT    COM     6912  10-20-83   12:00p
 CHKDSK    COM     6400  10-20-83   12:00p
 SYS       COM     1680  10-20-83   12:00p
 DISKCOPY  COM     2576  10-20-83   12:00p
 DISKCOMP  COM     2188  10-20-83   12:00p
 COMP      COM     2534  10-20-83   12:00p
 EDLIN     COM     4608  10-20-83   12:00p
 MODE      COM     3139  10-20-83   12:00p
 FDISK     COM     6369  10-20-83   12:00p
 BACKUP    COM     3687  10-20-83   12:00p
 RESTORE   COM     4003  10-20-83   12:00p
 PRINT     COM     4608  10-20-83   12:00p
 RECOVER   COM     2304  10-20-83   12:00p
 ASSIGN    COM      896  10-20-83   12:00p
 TREE      COM     1513  10-20-83   12:00p
 GRAPHICS  COM      789  10-20-83   12:00p
 SORT      EXE     1408  10-20-83   12:00p
 FIND      EXE     5888  10-20-83   12:00p
 MORE      COM      384  10-20-83   12:00p
 BASIC     COM    16256  10-20-83   12:00p
 BASICA    COM    26112  10-20-83   12:00p
      23 File(s)      28672 bytes free
```

If you are using a hard disk system, invoke the DIR command from drive C:

```
C>DIR
```

2.10 Tip: *If you find that your list of files is too long to fit on the screen, you can use the DIR command in PAUSE or WIDE mode.*

If your directory listing is too long to fit on a single screen display, the DIR command can be modified with the /P and /W switches. /P will cause the DIR command to pause after it fills the screen. /W creates a directory with only the names of the files. Date and time stamps and the size of the file parameter will be suppressed. The /W switch will produce a listing like the following.

```
A>DIR/W

Volume in drive A has no label
Directory of    A:\

COMMAND  COM     ANSI     SYS     FORMAT   COM     CHKDSK   COM
SYS      COM     DISKCOPY COM     DISKCOMP COM     COMP     COM
EDLIN    COM     MODE     COM     FDISK    COM     BACKUP   COM
RESTORE  COM     PRINT    COM     RECOVER  COM     ASSIGN   COM
TREE     COM     GRAPHICS COM     SORT     EXE     FIND     EXE
MORE     COM     BASIC    COM     BASICA   COM
      23 File(s)   28672 bytes free
```

2.11 Tip: *You can use a wildcard specification in the DIR command to list only a subset of the files in a subdirectory.*

Sometimes you want to list a group of files that are related by name. DOS makes this possible through the use of wildcards.

Suppose we want to find all the files in the current subdirectory that have a file extension of "*.EXE". We can do this by using the "*" wildcard character. In the example below it means "accept any combination of characters for the main part of the filename":

```
A>DIR  *.EXE

  Volume in drive A has no label
  Directory of A:\

  SORT       EXE      1408   10-20-83   12:00p
  FIND       EXE      5888   10-20-83   12:00p
         2 File(s)     28672 bytes free
```

Or suppose we want to list all the four-character filenames in the current subdirectory that had an "O" in the second position. We could accomplish this by using the "?" wildcard character, which means "accept any character in this position." Here is what the command and the resulting DOS response would look like:

```
A>DIR  ?O??.*

 Volume in drive A has no label
 Directory of A:\

 COMP       COM      2534   10-20-83   12:00p
 MODE       COM      3139   10-20-83   12:00p
 SORT       EXE      1408   10-20-83   12:00p
 MORE       COM       384   10-20-83   12:00p
      4 File(s) 28672 bytes free
```

Saving and Backing Up Data Files

It is important that you save your data files frequently while you are working on them. Do not wait until you are finished. After you save a file, you can still add, edit, or revise it as much as necessary. It takes only seconds to save your file, while it could take hours to re-create a lost file.

Even if you save your data frequently, various things can go wrong that result in a loss of data. Good file backup procedures can minimize the consequences of such disasters. Creating backup copies is a simple process of making copies of files on floppy diskettes and hard disks.

The following Tips and Traps introduce you to the importance of saving and backing up your data. The time to develop the right file-saving and backup habits is now! If you do so, you will spare yourself much irritation and gnashing of teeth in the future.

2.12 Trap: *If you do not save your work every 15 to 30 minutes, then unexpected problems can result in the loss of a significant amount of work.*

The more you use computers, the more you depend on their reliability. Unfortunately, a multitude of possible errors could cause you to lose all or some of the data you create with application programs. The following are some of the errors you might encounter:

- *Power failure.* A loss of power for just a second is enough to cause the loss of work you have just completed. This could mean simply losing the current application. However, if you are saving information at the time of the power failure, major damage to your entire disk could occur. This means that more than just the current file could be destroyed.
- *Hardware failure.* Computer hardware is usually very reliable. However, one of the components may not function properly. Your disk drive is the most likely component to fail, since it is the only component with moving parts. Disk drive problems can also result in data being destroyed.
- *Frozen program.* Your computer stops in the middle of a program, the screen freezes, the disk drive head either doesn't move or keeps spinning, and pressing the keys has no effect. This kind of problem is usually due to a bug in the program. You will lose the work you have created during the work session.

- *Bad disk.* Even with meticulous disk-handling practices, it is possible to encounter physical damage on a disk that can make the whole disk unreadable.
- *Pulling the plug.* Unlikely, you say? True, but it has been known to happen. Be careful to keep power cords away from likely pedestrian paths.

The lesson to be learned from this list is that you cannot depend on your system 100% of the time. So take preventive measures, and BACK UP YOUR WORK. Murphy was an optimist—so don't take chances!

2.13 Tip: *You can create a backup copy of your current file directly from the application program you are working with.*

With 1-2-3, for example, your work is saved with the /File Save command. Every time you create a file, save it on more than one diskette. This is a quick and easy way to create backups of key files. You may say to yourself, "I'll back up this file later; I don't want to take the time to save it twice now." But will you think of it later? Get in the habit of saving a backup copy on a second diskette immediately after you save the latest copy of the original file. Then you won't even have to think about it later.

2.14 Trap: *If you do not keep two generations of backups of important data files, you can sometimes lose data.*

If a file is corrupted in some way, the process of backing it up writes over the last good copy of it. The solution is to use TWO backup diskettes, so you have two generations of backups at all times. Figure 2-4 explains this important procedure.

2.15 Tip: *On a hard disk system, you can create a temporary subdirectory that will serve as a transfer area for copying from one floppy diskette to another.*

It is an extremely cumbersome process to copy files from one floppy diskette to another when using a hard disk system. You usually have only one floppy disk

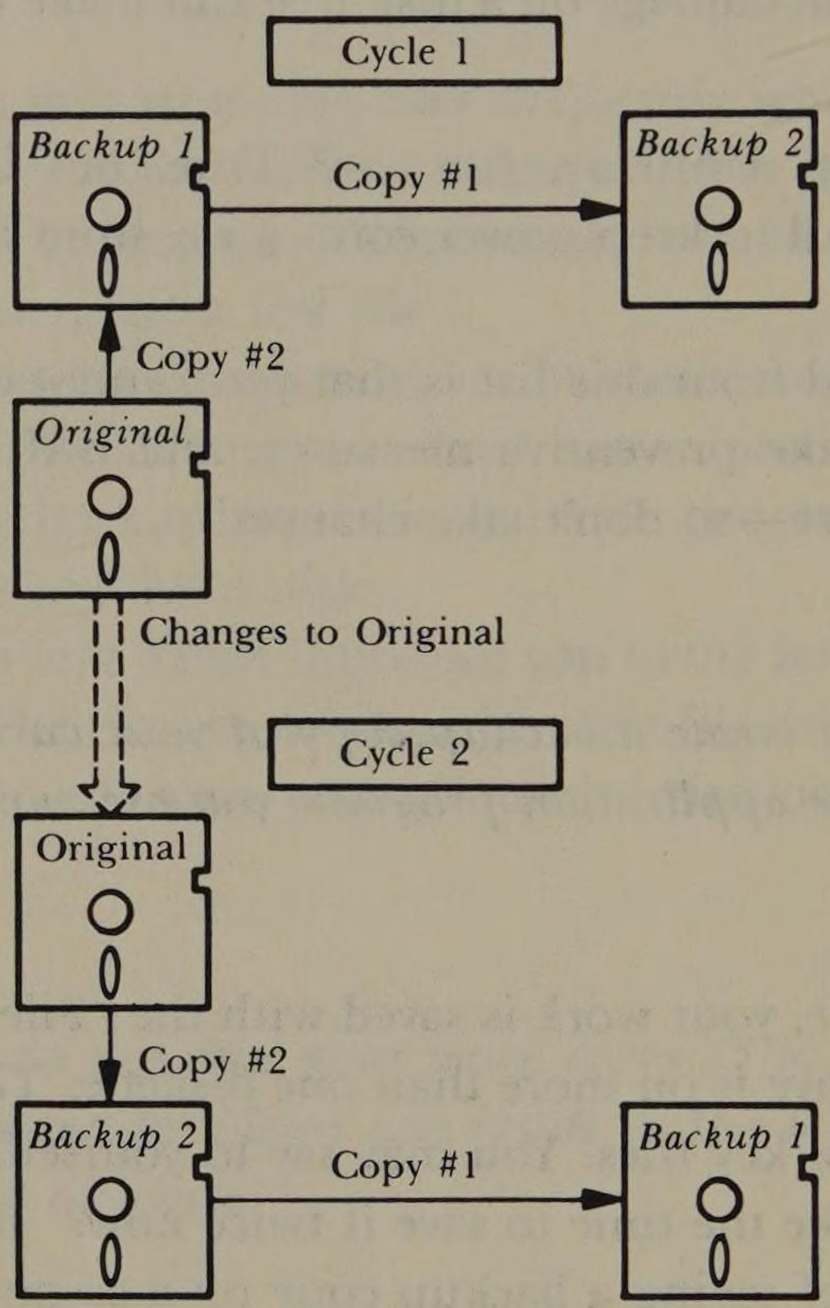


Figure 2-4. Generational backup procedure

drive, and DOS will prompt you to switch disks in the A drive countless times during the copying process.

Instead, you can create a subdirectory to serve as a temporary transfer area. Next, use the COPY command to transfer the files from your floppy diskette to your hard disk. Finally, use the COPY command again to transfer the files from the hard disk to a second floppy diskette. When you are done, erase all the files in the temporary subdirectory. The commands and DOS responses look something like this:

```
C>MKDIR\TEMP

C>CD\TEMP

C>COPY A:*.*
A:BUDGET83.WKS
A:BUDGET84.WKS
A:BUDGET85.WKS
A:YTDACCTS.WKS
      4 File(s) copied
```

```
C>COPY *.* B:

Insert diskette for drive B: and strike
any key when ready

 BUDGET83.WKS
 BUDGET84.WKS
 BUDGET85.WKS
 YTDACCTS.WKS
    4 File(s) copied
C>ERASE *.*
```

This approach saves time and could also prevent disaster: It is easy to get confused when you have to switch disks so often. You can simplify this procedure even more by creating a batch file that contains all the commands we showed above. (Refer to Chapter 5, "Batch Files.")

2.16 Tip: *You can back up hard disk files onto floppy diskettes using the COPY command.*

Losing files stored on a floppy diskette can be quite disastrous. If a hard disk fails, the severity of the disaster is magnified to a devastating level. Consider the following scenario: All accounting records for a company are stored on a hard disk. Backing up is done infrequently, at best. A momentary power failure occurs, and from that time on, the hard disk is no longer accessible. It may or may not be possible to recover the stored files; the company's records could be lost forever. Even if the worst case isn't realized, the time spent recovering from this near disaster could be very costly. Don't let this happen to you. If you are using a hard disk system, it is imperative that you back up religiously!

The simplest way to back up a small number of files from the hard disk is to do it by subdirectory. Here is an example:

```
C>CD\BUDGDATA
C>COPY *.* A:
BUDGET83.WKS
BUDGET84.WKS
BUDGET85.WKS
YTDACCTS.WKS
        4 File(s) copied
```

This will copy all the files in the BUDGDATA subdirectory on the hard disk to the floppy diskette in drive A, as long as the files in this subdirectory do not completely fill up the floppy. If this happens before all the files have been copied,

complications arise: you have to use one or more additional floppy diskettes, record which files are on which disk, make sure you haven't missed any, and so forth.

Versions of DOS from 2.0 on include special BACKUP and RESTORE commands to facilitate the process of backing up hard disk files onto floppy diskettes. However, some versions of these commands have been unreliable, and unfortunately, even with these special commands, backing up a hard disk to floppies is a tremendous bother. Backing up to removable tape cartridges or removable Winchester disk cartridges is usually a much more convenient and practical solution. (See the following Tip.)

2.17 Tip: *It is usually worth getting a removable Winchester cartridge unit or removable tape cartridge unit for hard disk backup purposes.*

As we saw in the previous Tip, backing up hard disk files to floppy diskettes can be a tremendous bother. You really want to be able to issue one command and have the entire contents of your hard disk backed up onto some removable medium. Currently, the two major options are removable tape cartridges and removable Winchester disk cartridges. The advantage of the Winchester cartridge system (like the Bernoulli Box made by IOMEGA Corporation in Ogden, Utah) is that it is useful not only for backup but also for additional on-line storage.

Automating DOS Procedures

As you become more familiar with the DOS commands introduced in these first two chapters, you will discover that many tasks will become routine procedures, usually involving more than one DOS command. You can save time and effort by combining several commands into a single file known as a *batch file,* and then execute it by simply issuing the name of the batch file. Throughout this book you will be introduced to batch files that will not only save time but could also prevent you from making a terrible error, like erasing valuable data.

Batch processing files are one of the most useful and powerful features of DOS, so it is appropriate to introduce them at this time. We will explore the more advanced techniques in succeeding chapters, especially Chapter 5.

2.18 Tip: *If you use one program most of the time, use an AUTOEXEC.BAT file (the extension stands for BATCH) to load that program automatically when you boot the system.*

Now that you have installed your 1-2-3 program, let's create a batch file that will automatically load 1-2-3 when you first start up your system. This is a special kind of batch file that must be named AUTOEXEC.BAT. Writing a batch file is simply a matter of listing the DOS commands to be performed.

The first line accomplishes two tasks: naming the file (AUTOEXEC.BAT), and instructing DOS to copy it to the current default directory. That directory should be the root, where DOS resides. The second line is the command you issue to load the 1-2-3 program. The last line involves pressing two keys, the F6 function key and then the ENTER key, and instructs DOS to save your batch file to the diskette in drive A. Be sure to have your 1-2-3 System Disk in drive A before you start this procedure.

```
COPY CON: AUTOEXEC.BAT
LOTUS
<F6> <ENTER>
```

To test your batch file, be sure that the 1-2-3 System Disk is in drive A and then restart your system by pressing the CTRL, ALT, and DEL keys simultaneously. When DOS starts the booting process, it checks to see if an AUTOEXEC.BAT file exists, and if so, reads the commands listed in the file. Now, wasn't that easy?

3

Managing Files

As with a paper-filing system, there are different approaches to organizing disk files, and some are more effective than others. This chapter focuses on the DOS commands that can assist you in maintaining your files. You will learn how to use these commands effectively and how to avoid potential hazards. The Tips and Traps in this chapter also include many time-saving suggestions and batch files.

In this chapter we will cover file-naming conventions and the DOS commands COPY, COMP, RENAME, ERASE, DELETE, and VERIFY. These commands are concerned primarily with managing single files or pairs of files. What you need to do with larger groups of files will be covered in Chapter 4, "Managing Disks and Directories." This chapter is divided into three sections:

- File-naming conventions. Tips and Traps that will be especially helpful when you first start using DOS file-handling commands, with emphasis on the COPY command.
- More uses of the COPY command. These intermediate and advanced applications will show you how to use the COPY command to update files, use wildcards safely, create files, and copy files between devices, to name but a few.

- Other file-management commands. The commands COMP, RENAME, ERASE, and VERIFY represent effective file management techniques when properly implemented. However, several potential traps are associated with these commands, and this section focuses on them.

File-Naming Conventions And the Basic Uses of COPY

Before you can effectively use DOS commands that refer to files, you must understand the DOS file-naming conventions. DOS needs to know the following to identify a file or a group of files:

1. The drive designation (A:, B:, C:, and so on) for the disk drive containing the file(s).

2. The path from the current subdirectory to the subdirectory containing the file(s). (See Chapter 4, "Managing Disks and Directories," for a discussion of DOS subdirectories.) When you have no subdirectories, as is usually the case on floppy disks, there are, of course, no subdirectory paths to be concerned about.

3. The filename proper, made up of from 1 to 8 characters. (Wildcard characters can be used in this part of the specification. See point 4.)

4. The optional file extension, made up of a period and from 1 to 3 characters. (Wildcards can also be used here.)

Recall from Chapter 1 that the wildcard characters * and ?, also called *global filename characters,* allow you to refer to groups of files. See the index of your DOS manual for additional descriptions of how they work and how you can use them.

In the simplest case, when you are referring to a file on the current default drive in the current subdirectory, with no global filename characters specified, you supply a specification something like this:

```
A>ERASE BUDGET85.WKS
```

When DOS interprets this command, it may "say" to itself: "Hmm, erase a file on the current default disk drive (that's drive A because I'm giving an A prompt) in the current subdirectory on that drive, and I know it's just a single file because there are no wildcard characters. Okay, I'll go ahead and erase that file since I know just where to find it." But consider this example:

```
B>ERASE A:\LOTUS\*.WKS
```

When DOS interprets this command, it says, "Aha, this one's got real meat to it. I need to erase a file that's not on the current default drive (B), but on the A drive. Furthermore, it's in a subdirectory called Lotus within the root directory on this disk. (The initial \ after the A: denotes the root directory. Each subsequent subdirectory name is followed by a \ to separate it from what follows.) But whoops, it's not just one file: the global * character tells me I need to erase all files in this subdirectory with the WKS file extension on this disk. Okay, done!"

Understanding how and why DOS interprets filename specifications is important because it is very easy to erase the wrong file, copy a file to the wrong subdirectory, and so forth.

When you create files and give them names (either from DOS or your applications programs like Lotus 1-2-3), you need to take great care in choosing the names to assign. The following Tips and Traps give you some hints.

3.01 Tip: *Meaningful filenames make it easier for you to identify your files.*

If you were handed a file folder labeled BDT5, you would probably have to open the file and examine its contents to know what the file is about. However, you would not have to examine the contents of a file labeled 85BUDGET because the name alone is sufficient identification.

The same principle applies to naming files on a disk. If you had to display the contents of a file on the screen in order to identify it, you would spend more time identifying files than using them. These names clearly identify file contents: JAN85P&L, Q1ACCTS, and CH3BOOK.

3.02 Tip: *Assign mnemonic names to your files to help you identify what's in them.*

The previous Tip stressed the importance of assigning meaningful names to your files. One helpful convention is to use mnemonic references. Because a filename can have only eight characters, abbreviations are generally necessary. Mnemonic filenames can help you get the most out of a limited number of characters.

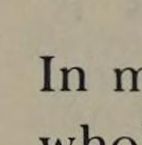
You must take care when creating mnemonic labels so that they are meaningful both to you and other users who will access your files. It is easy to get carried

away and end up with cryptic filenames that only you can understand. Consider these examples: 85FIST (1985 Financial Statement), QTFIREP (Quarterly Financial Report), and BK3 (Chapter 3 of a book manuscript). These names might be more understandable to other users if they were expanded to 85FINSTM, QTFINRPT, and BOOKCH3.

3.03 Tip: *You can code a great deal of information into your filename.*

In many cases, you may be working with an application for which you have a whole collection of files. For instance, you may be doing budget worksheets for several different departments. Unless you code key descriptive bits of information into the filename itself, you will soon lose track of which file is which.

Suppose you are working on a budget for department A for the first quarter of 1985. You may want to assign a name like BUDGA851.WKS. The first four characters identify the file as a budget file, the fifth as belonging to department A, and the last three as representing the first quarter of 1985. (The process of encoding such filenames can be automated by creating a Lotus macro that will prompt you for the department, year, and quarter when you save the worksheet.)



3.04 Tip: *By coding your filenames, you can sort them later.*

Since this method codes information into fixed positions in the eight-character filename, it also allows you to use the DOS SORT command to sort a directory listing in a meaningful way (see Chapter 6, "Redirection, Piping, and Filters"). The order in which you place the various pieces of information in the filename can make a difference. For instance, we would not want BUDGA185.WKS as a name. Why? Because a SORT would then give you year within quarter rather than quarter within year, which is probably what you want. If you want department within year instead of year within department, as shown, just change the name to BUDG851A.WKS.

In the example we have given here, the program automatically assigns a file extension. Most applications software does this. For instance, Lotus 1-2-3 automatically assigns a ".WKS" file extension to all worksheet files. In other cases, however, you are free to assign the file extension yourself, which gives you another three characters that you can use to encode important descriptive information. Usually, you want to use the file extension to indicate the general type of file, which comes in handy when you want to handle those files as a group. For instance, you may use a file extension of .DAT to denote "data."

3.05 Trap: *When you assign file extensions, do not use ones already used by DOS or applications programs.*

There are technically no reserved extensions. However, DOS and applications programs do follow certain conventions, and if you assign an inappropriate extension to a file, you may get unexpected and undesirable results. Therefore, it is important to be aware of standard filename extensions, which we will discuss in the following paragraphs.

Executable programs have .EXE or .COM extensions. Most DOS program files are .COM files. Batch files must have the .BAT extension so that DOS can recognize and execute them as batch-processing routines.

Each programming language has its own file extension convention for its program source files. If you look at a directory listing of the files on the DOS Supplemental Programs diskette, you will see some files with the .BAS extension. These files were created using the BASIC programming language. Other languages use different extensions for identification.

A program often refers to files containing data needed to run it. These are called program data files and may have a .DAT extension.

Sometimes a program needs a temporary work file, like a scratch sheet. The .$$$ extension may be used to identify these files, which are deleted before a program finishes. If you see one of these files on a directory listing, something may be wrong with your program, or something may have gone wrong when you last used your program—a power failure, for example.

Most applications programs assign a unique extension to standard data files, and will access only files with that extension. For example, Lotus 1-2-3 assigns the .WKS extension to your spreadsheet files.

Some applications programs can create printer-type output, but store it as a file, rather than send it to the printer. For example, Lotus 1-2-3 assigns such a file a .PRN extension. Print file data is stored in standard ASCII text format, making it accessible to word-processing programs.

Some word-processing programs, like WordStar, automatically create backup copies of your files, assigning a .BAK file extension to them. The first time you create and save a document, it is saved on the disk using the filename you have specified, with a .TXT extension. If you edit the file and save it, WordStar first renames the old version of the .TXT file, giving it a .BAK extension. It then saves the newly edited version, using the .TXT extension.

3.06 Trap: *If you type a filename longer than eight characters, or an extension longer than three, DOS truncates the extra characters.*

If DOS has to shorten your filename, you will not be informed of this action, which can cause problems. When you search for your file later, you may not recognize the truncated filename. Also, and even more troublesome, you could inadvertently overwrite an existing file because the names would be identical after DOS truncated the name you entered. Let's say you created a file with budget data for the first quarter of 1985 and assigned the filename BUDGET1Q85 when you saved it. Only eight characters are allowed, so DOS truncates the last two characters, and your file is actually saved with the name BUDGET1Q. A first-quarter budget file for another year thus could be easily overwritten.

3.07 Trap: *Be careful not to use reserved device names as filenames.*

Device names are names for specific hardware devices, like a printer. These reserved device names show the devices they reference:

AUX *same as COM1; first serial port,*

COM# *asynchronous channel (#1 or #2), serial port,*

CON *console (display screen),*

LPT# *parallel printer port (#1 or #2),*

NUL *the null device; a temporary holding place in memory,*

PRN *first parallel printer port.*

DOS will not allow you to create a file with any of these names.

3.08 Tip: *Use the COPY command whenever you want to copy a file.*

The COPY command is one of the most frequently used of the various DOS commands. It is usually issued to copy one or more files from one disk to another. (Other Tips in this chapter will give more uses of this versatile command.) The command notation for creating duplicate files is COPY *source destination*, in which *source* is the name of the file you want to copy from, and *destination* is where you want to copy the file to. For instance, assume that you have a file named BUDGET85.WKS on a disk in drive A, and you need an exact duplicate of it placed on a disk in drive B. The following command would accomplish this task for you:

```
A>COPY BUDGET85.WKS B:
```

This notation does not include a filename specification in the destination because you want the duplicate file on drive B to have the same name as the original file on drive A.

Similarly, if you want to copy a file from another drive to the current default drive, and the filename is to remain unchanged, the drive destination is not required. The COPY command will then look something like this:

```
A>COPY B:INVEST.WKS
```

In this example, the file named INVEST.WKS will be copied from drive B to the current subdirectory in drive A, and the name will stay the same.

3.09 Trap: *When you issue the COPY command, be sure to supply the source parameter first and the destination parameter second.*

A common error in using the COPY command is to reverse the order of the source and destination filenames. This is most likely to happen if you are familiar with other operating systems and mistakenly use their conventions to specify filenames. (The PIP command in the CP/M operating system, for example, follows the reverse convention, with the destination appearing before the source in the command line.)

PC-DOS, however, expects the source first and the destination second. To prevent disaster, you should drill this order into your head and fingers. Or think about it this way: Copying a file is like picking up a book and moving it from one table to another. You have to pick it up first before you can put it anywhere, so deciding on the source logically comes first.

3.10 Trap: *Only formatted disks can be used as destination disks for the COPY command.*

DOS cannot place files on a disk that has not been previously formatted. (Chapter 1 explains why disks need to be formatted and describes the procedure in detail.) To check whether a disk has been formatted, issue the DIR command. It is good practice to format several disks at one time so that you can always count on a usable disk.

3.11 Tip: *Using a write-protect tab will prevent you from destroying your original file when you use the COPY command.*

Although the COPY command is versatile and very powerful, it does have potential hazards. You can minimize the risk by making it impossible to overwrite your original files. A small square notch, called the write-enable notch, is on the right-hand side of all but a few commercial program diskettes. When the notch is not covered, information can be read from and written to the diskette. If the notch is covered, the diskette can be read, but nothing can be written on it. When you buy a box of blank diskettes, you get a sheet of write-protect tabs that can be used on a diskette to make it read-only.

You can prevent accidental erasing of important files if you habitually write-protect the disk you are copying. This is especially important when you copy program files to make backup copies. If you inadvertently copy over your only program disk before you have made a backup, you are in big trouble. Unfortunately, some manufacturers employ copy protection schemes that require you to use their own copy programs to copy the main program disk to a hard disk. These special copy programs typically write a key onto the original program disk that prevents it from being copied a second time. Therefore, you must be sure that the write-enable notch on the program disk is not covered; otherwise, the copy won't work. If you carefully follow the installation instructions for the software in question, you will have no problems.

3.12 Trap: *When you use the COPY command, include the filename extension so that DOS can locate the files on your disks.*

Even though filename extensions are optional, once they have been assigned DOS requires that you include them when specifying filenames. In the case of the COPY command, you must enter the extension for the source file. It is wise to enter it for the destination file, too. For instance, if you wanted to copy the BUDGET.WKS file but forgot the extension, this is what would happen:

```
A>COPY BUDGET B:

        0 File(s) copied
```

No files were copied because DOS couldn't find a file named BUDGET. A file named BUDGET.WKS exists on the disk, but DOS wasn't told to look for it.

If you omit the extension when identifying the destination file, the COPY

will be successful, but you will have a file without a filename extension. This could cause confusion if you had intended to create a copy of the file with the same filename and extension as the original.

3.13 Tip: *If you have only one floppy drive, you can still copy from one diskette to another.*

Suppose you have only one floppy drive and a hard disk (or just one floppy drive). With this configuration, there are two possible approaches to copying files from one diskette to another.

The first is to issue the COPY command exactly as you would if you had two floppy drives.

```
C>COPY A:*.WKS B:
```

DOS knows that only one floppy drive exists and considers it to be both A: and B:. DOS will copy your files, one by one, from one diskette to another. However, the entire operation will take place in drive A, and you will be required to switch diskettes for each file that is copied.

The second and more efficient approach is to utilize a temporary directory on your hard disk. You copy the desired files from the source diskette to the temporary directory. You then switch diskettes in drive A, placing the destination diskette in the drive, and copy the contents of the temporary directory to this diskette.

You can create a batch file to handle this procedure and ensure a safe operation. Before you execute the batch file, be certain to have a formatted diskette ready to insert in drive A when you're prompted to do so. This example assumes A: is the default drive and that a subdirectory called TEMPDIR exists as a child of the root directory on the hard disk. This is how we would create the batch file by using the COPY CON: command, which accepts input from the keyboard (console):

```
A>COPY CON: COPYFILE.BAT
ECHO OFF
ECHO COPIES FILES FROM ONE
ECHO DISKETTE TO ANOTHER.
ECHO OK TO ERASE TEMPDIR ON C: ?
ECHO PRESS ANY KEY - ELSE CTRL+BREAK
PAUSE
ERASE C:\TEMPDIR\*.*
ECHO INSERT SOURCE DISKETTE IN DRIVE A:
ECHO PRESS ANY KEY WHEN READY
PAUSE
```

```
COPY A:*.* C:\TEMPDIR
ECHO REMOVE SOURCE DISKETTE FROM A:
ECHO INSERT DESTINATION DISKETTE IN DRIVE A:
ECHO PRESS ANY KEY WHEN READY
PAUSE
COPY C:\TEMPDIR\*.* A:
^Z
```

The last line, ^Z, is created by pressing the F6 function key and pressing ENTER. It indicates that you have finished creating the file and want to save it on your disk. (You can of course create batch files using any word-processing program or text editor if you find creating them with COPY CON: too cumbersome.)

More Uses of the COPY Command

3.14 Trap: *Regardless of the type of copying you are doing, the COPY command will proceed whether or not there is already a file with the destination name.*

If there is no file with the destination name, then a new file will be created when the COPY command is executed. If a file already exists on the disk, it will be overwritten, and important information might be destroyed. DOS doesn't warn you that an existing file will be erased, so you must be careful when you copy files. If you want to be warned, you can create your own batch file to "front end" the copying process. Here is one, COPYSAFE.BAT, which controls copying from drive A to drive B: IF NOT EXIST B:%1 COPY A:%1 B:.

To copy the file BUDGET85.WKS from drive A to drive B you would issue: COPYSAFE BUDGET85.WKS. DOS would execute the COPY only if the file did not already exist on the target disk.

3.15 Trap: *When you try to replace an old version of a file with a new version using COPY and there is insufficient disk space, the old version on the destination drive will be wiped out in the process.*

When a file is updated, the old version is erased to make room for the new version to be copied onto the disk. When DOS attempts to copy the new version, the process is interrupted if there isn't enough disk space to accommodate the entire file. Consider this example:

```
A>COPY NOTEFILE.BAK B:
Insufficient disk space
        0 File(s) copied
```

If you look at a directory listing of your files on the destination disk, you won't find a copy of NOTEFILE.BAK. You will need another formatted disk if you want to try to save the file again, unless you first erase one or more other files to create more space.

This problem can be prevented if you use the DIR command to get a directory listing of the files on a particular disk. This command reports available disk space by letting you know how many bytes are free. Here is a representative directory listing:

```
A>DIR

 Volume in drive A has no label
 Directory of  A:\

FILE1      WKS      54224   2-08-85   7:37a
FILE2      WKS      81093   2-10-85  11:08p
FILE3      WKS      91886   3-22-85  10:09a
FILE4      WKS      67917   4-03-85  20:19p
FILE5      WKS     101220   5-01-85  12:33p
FILE6      WKS      49105   5-01-85  17:21p
       6 File(s)        8961 bytes free
```

With less than 9,000 bytes free, this disk is almost full. If you were to make substantial changes to one of these files, you probably would not have sufficient space on the disk to save the file again. If you suspect that your disk may be nearing maximum storage capacity, use the DIR command to determine available disk space.

3.16 Trap: *If there is insufficient space on the destination disk for the file(s) to be copied to it, the COPY command will be unable to complete the assigned task.*

This Trap differs from the previous one because you are copying files from the source drive that are not necessarily duplicates of those on the destination drive. This is due to not having sufficient space on the destination disk for the COPY command to copy all specified files. Should you encounter this Trap, DOS will display one of two error messages. The first you will see if you attempt to copy more files than there is room for on the destination disk. DOS will copy as many files as possible and then stop executing.

```
A>COPY *.WKS B:
BUDGET.WKS
INVEST.WKS
Insufficient disk space
        2 File(s) copied
```

If your destination disk doesn't have enough room for the first file that satisfies the COPY specification, DOS halts the COPY procedure. DOS is not programmed to search for a file that will, in fact, fit in the available space on the destination disk. Consequently, none of the files will be copied. You will see the following message on your screen:

```
A>COPY 85BUDGET.WKS B:
Insufficient disk space
        0 File(s) copied
```

3.17 Trap: *When you use directory path specifications in the destination parameter of the COPY command, make sure the directories you specify actually exist.*

Consider the following command:

```
COPY A:\MYPROG C:\PROGRAM1
```

You assume that a directory called PROGRAM1 exists on drive C. If it does, you will end up with a copy of MYPROG (with the same name) in the PROGRAM1 directory of drive C. If the PROGRAM1 directory does not exist, however, you will end up with a copy of MYPROG named PROGRAM1 in the root directory of drive C. This could be confusing. If there were already a useful file in the root directory of drive C named PROGRAM1, the result could be disastrous, because the old PROGRAM1 would be completely replaced.

The DIR command can help you by listing the directories on a disk before you issue the COPY. For instance, DIR C:*. will list all the files in the root directory of drive C with no file extension. Such "files" are usually subdirectories and will be listed with the <DIR> notation in the second column of the directory listing.

3.18 Trap: *If you are not careful in supplying the destination parameter of the COPY command, you can wipe out existing files.*

Files saved in the same subdirectory on the same disk must have unique filenames and extensions. However, files with the same name can exist in different subdirectories. If both the source and destination disk/subdirectory combinations contain files with identical filenames, issuing the COPY command could result in losing an important file on the destination disk. If the contents of both files are identical, then the COPY command will not change anything. However, if the file contents are different, then COPY will replace the file on the destination disk with the one on the source disk of the same name.

3.19 Tip: *Using wildcards makes it easy to copy files selectively.*

The wildcard characters (* and ?) can be used with many DOS commands. However, using them with the COPY command is particularly handy, because you can copy several files by typing a single command. You do this by using the * and ? to reference either the full filename or extension, or just some of the characters. Remember, * substitutes for the remainder of the full name or extension, while ? stands in for a single character anywhere in a filename. The following examples show this shorthand approach to specifying which files are to be copied:

```
A>COPY *.* B:
```

This command copies all files from the current subdirectory of drive A to the current subdirectory of drive B.

Note: If you have a hard disk, this Tip could potentially be a Trap. If you are using subdirectories, you may think you are copying the whole disk, when in fact the command affects only the current source subdirectory. (See Chapter 4, "Managing Disks and Directories," for Tips and Traps on creating and using subdirectories.)

Here is another way to use wildcard specifications:

```
A>COPY B:*.COM
```

This command copies all the .COM files from the current subdirectory in drive B to the current subdirectory in drive A. (The default drive, drive A, need not be specified.)

```
A>COPY ??BUDGET.WKS B:
```

This command copies all files with BUDGET as the last six characters in the filename, with a .WKS extension. Issuing this command would copy the following

files and any others whose filenames ended in BUDGET.WKS, such as:

```
83BUDGET.WKS
84BUDGET.WKS
Q1BUDGET.WKS
Q2BUDGET.WKS
MYBUDGET.WKS
```

The following command copies all files with a .COM extension to drive B. It illustrates how useful assigning filename extensions can be: you can extract all files of a particular type, regardless of name.

```
A>COPY *.COM B:
```

3.20 Tip: *Develop file-naming conventions that organize files by subject.*

This Tip goes hand in hand with the previous one. If you use a common identifier when assigning names to files that relate to the same subject, you will be able to track them more effectively. You can use this type of naming convention, together with wildcard specifications, to perform DOS functions easily on related groups of files.

You can't always count on files being related by a common extension. You might have correspondence files, data base applications, and spreadsheet analyses all related to the same subject. Their file extensions won't identify related files. But if you use a common identifier in the main part of the filename, you can extract all related files, regardless of the type of file. Consider the following files that are related by a common naming convention, though the file extension is different: NET1BUDG.WKS, NET1P&L.WKS, and NET1PLAN.DOC. Note the placement of the common identifier—the first four characters. It is more effective to have the common identifier in the lead position, rather than embedded in the filename, so that DOS can efficiently search for related groups of files.

This approach makes it easier for you to search for related files when you look at a file directory listing. With the same naming convention, but without concern for placement, the filenames might look like BUDGNET1.WKS, P&LNET1.WKS, and NET1PLAN.DOC.

The first listing of filenames clearly is more effective, since you could list them all at once with a DIR NET1*.* command.

3.21 Trap: *Make sure both the source and destination devices you supply in a COPY command exist, and are on-line. Otherwise, DOS may hang.*

To copy information from the keyboard into a disk file, use the following command syntax:

```
COPY CON: A:MEMO.DOC
```

All keystrokes are entered until you press the F6 function key (which enters an end-of-file marker) followed by ENTER. The data will be saved as a file named "MEMO.DOC" on the disk in drive A.

The next example copies the contents of the named file onto the printer:

```
COPY B:MEMO.DOC LPT1
```

Other devices can be specified as source or target when you use the COPY command. But if DOS can't successfully make contact with the destination device, processing will stop and your system will be stalled. Sometimes this is a temporary condition; in other instances, you will have to turn off your system and then restart it, or restart it with CTRL-ALT-DEL. If you need to restart, you will lose programs or data held in system memory.

It is a good idea to check your devices before you issue any DOS commands that reference them. Make sure that they are turned on, plugged in, properly connected to your system unit, and on-line, which is usually indicated by a light on the device.

3.22 Tip: *You can recover accumulated dead space on a much-used diskette by using COPY *.* to copy its entire contents to a new diskette.*

Using a disk over and over slows down its performance. The process of saving and deleting files creates what is referred to as "dead space," small clusters of sectors scattered throughout the disk that are not sufficiently large to accommodate a file.

Consequently, files are saved in nonadjacent locations, and retrieval time is significantly increased. This can slow down DOS operations as well as applications programs. It also reduces the amount of the total disk space that you can use. The command COPY *.* will copy all files from the current subdirectory on one disk to another, with more effective placement of files on the destination disk. This process, in effect, can clean up your much-used diskettes, resulting in better performance.

3.23 Tip: *You can use the COPY command to concatenate (combine) several files, which will merge the source files and keep the originals intact.*

It is often useful to combine several small files into one file, since working with one larger file is easier and more convenient. The concatenation option of the COPY command appends the contents of subsequent files to the end of the first. For example, suppose that you want to combine two text files so that you can print them as a single document. The concatenation option of the COPY command makes this possible, without destroying the original versions. The syntax is as follows:

```
A>COPY FILE.TXT+OTHERFILE.TXT NEWFILE.TXT
```

Between the two source files, a + symbol indicates which files are to be combined or concatenated. The last filename in the command line, NEWFILE.TXT, becomes the name of the newly created file. DOS will display the name of each source file as it is copied and then report that one file was copied, since only one file was created. The procedure will look something like this on your screen:

```
A>COPY FILE1.TXT+FILE2.TXT HUGEFILE.TXT
FILE1.TXT
FILE2.TXT
        1 File(s) copied
```

If you look at the directory listing of your files, you will find the original files and the newly created file:

```
A>DIR
 Volume in drive A has no label
 Directory of  A:\

FILE1        TXT     11420     3-02-85   8:30a
FILE2        TXT     13476     3-05-85  10:33a
HUGEFILE     TXT     24896     3-10-85   9:21a
          3 File(s)   311,468 bytes free
```

This concatenation technique can be particularly useful when you want to combine two batch files.

3.24 Trap: *The concatenation option of the COPY command assumes that the source files will be ASCII text files. If they are binary, then a special switch must be set.*

The COPY command has the optional arguments /A and /B. Although rarely used for most COPY procedures, these parameters are important when you are using COPY to concatenate files. They are used to distinguish between ASCII files and binary files, thus the A and B. ASCII files contain only normal text characters, such as those used to create a letter or document. Binary files often contain special nontext characters. Programs and many data files are stored as binary files. Ordinarily, it would not make much sense to concatenate binary files. If you need to do so, however, you must use the /B option. If you specify neither /A nor /B, DOS assumes /A by default.

3.25 Trap: *Using wildcards in COPY commands can cause problems, especially when the concatenation option is used.*

The COPY command will proceed even if there is an existing file with the destination name, and valuable data can be overwritten. You will not be warned if you are about to destroy an existing file; once it is destroyed, it cannot be recovered. If you use wildcards (* and ?), do so carefully, because they force COPY to select any file that matches the global specification. Using wildcards in the destination name will create multiple destination files. Sometimes this is just what you want, but be sure you do.

The next example demonstrates another potential Trap when wildcards are used to name a destination file that is also a source file. The following represents a sample list of files, in the order they appear in the file directory:

```
JAN.ACC
FEB.ACC
MAR.ACC
QTR.ACC
YTD.ACC
```

Let's say you want to merge these source files into one destination file called QTR.ACC. One file is easier to deal with. It would seem appropriate to use

COPY to accomplish this task, so you issue the following command:

```
A>COPY *.ACC QTR.ACC
```

During the execution of this command, DOS displays this error message:

```
Contents of destination lost before copy
```

DOS has encountered a source filename that is identical to the destination filename. When wildcards are used in filenames, DOS scans all the files until it finds one that matches the wildcard name, and then processes that file. The first file, JAN.ACC, matches the *.ACC specification, and is included in the destination file known as QTR.ACC. DOS actually creates a new file, QTR.ACC, and considers it to be the destination file for all the source files that match the wildcard name.

The information stored in the source file QTR.ACC is erased when DOS creates the destination file by the same name (QTR.ACC), because files with the same name cannot exist on the same disk or in the same subdirectory. The source files JAN.ACC, FEB.ACC, and MAR.ACC are successfully appended to one another and placed in the newly created QTR.ACC destination file. However, when DOS attempts to append the source file named QTR.ACC, the error message is displayed because the original source file has been altered. At this point it is too late to salvage the contents of the original source file, QTR.ACC.

However, there is a way you can use the COPY command that will, in fact, successfully combine files under these circumstances. Use

```
A>COPY QTR.ACC+*.ACC
```

The last file without a plus sign in front of it is QTR.ACC, and so it becomes the destination file. QTR.ACC is also the first source file listed. Since the names are identical, DOS will skip this first source file, and then append all other .ACC files to the destination file.

3.26 Tip: *You can use the COPY command to change the DATE and TIME stamp of a file.*

When a file is copied, its original date and time are also copied. To change these notations, issue the COPY command in the following way:

```
A>COPY B:FILE.EXT+
```

DOS will combine all the files named FILE.EXT (there is only one in this case), and save them as one file on drive A, the default drive. Because DOS considers the results of this action to be a "new" file, new date and time stamps are applied. The contents of this file are unchanged.

Why would you want to change the date and time stamp? One reason is that you may have incorrectly set the date and time when you booted.

If you would like to change the date and time without moving the file from one diskette to another, use this command:

```
A>COPY B:FILE.EXT+,, B:
```

The two commas after the plus sign tell DOS not to look for another filename after the plus sign.

To perform the same operation on a program file or binary data file, the /B switch must be added when issuing the command:

```
A>COPY/B FILE.EXT+,,
```

or

```
A>COPY FILE.EXT/B+,,
```

This is important because a program file or binary data file may contain an end-of-file marker, ^Z (CTRL-Z). Using the /B setting ensures that DOS will copy the entire file. A file of this type that was copied without a /B may run inconsistently or not at all. A loss of data could also occur. DOS knows that it has found the last of the source filenames when it encounters the two commas. This syntax also makes sure that DOS will not confuse the destination filename with a nonexistent source filename.

3.27 Tip: *To determine whether a communications port is active, try copying to it.*

Occasionally something goes wrong with a serial (asynchronous) port that you are using for communications. Sometimes nothing physical is wrong, but DOS itself is confused about which logical device name belongs with which physical port. This can happen, for example, when you already have one serial port in your PC and you add an expansion board that contains a second serial port. Usually, this second serial port is configured by default as COM1. If so, DOS will think it has two COM1 ports, and neither of them may work. You may have to set

a "jumper" on the expansion board to configure the new port as COM2.

When you have an external modem with a visible Send Data light, there is a quick way to confirm that you have the right logical device names pointing to the right physical ports:

1. Connect an external modem to the port you want to test (the one that you think is COM1). Make sure that the modem is turned on.

2. Enter the following command:

```
COPY CON: COM1
xxxxxx                    (any string of characters you choose)
```

3. Press the F6 function key, then the ENTER key. If you see the Send Data light blink, then the modem has received the xxxxxx string you sent, and your COM1 is probably all right. If not, try the same copy procedure, but with a destination of COM2. If neither works, then you can conclude that there is a problem, and you will need to reset a jumper on the expansion board or replace one or more of your ports or cables.

3.28 Tip: *Use the /V option to verify that a copy has been executed successfully.*

The /V option instructs DOS to verify that the destination file was created accurately and is in fact an exact duplicate of the source file. Although PC-DOS rarely makes a mistake when copying files, you can be sure that your file was copied correctly by using the /V (for *verify*) when you issue the COPY command. The command syntax is

```
A>COPY 85BUDGET.WKS B:/V
```

The verifying process will slow down the command. However, it may be worth the extra time if you are working with important files, or if you suspect that your disk drive is malfunctioning. (Issuing the VERIFY ON command causes DOS to perform a verify on each subsequent disk write operation, including COPY operations, whether COPY/V has been specified or not.)

3.29 Tip: *You can also create a file by typing from DOS.*

You might find it helpful to be able to create a file without having to use a full-functioning word-processing program. To begin, issue this command:

```
COPY CON: "filename"  (the name of the file)
```

Type as many lines as you want. When you are finished, press the F6 function key and ENTER key to save your files. Files created this way can be edited with the EDLIN or any text or word-processing program.

3.30 Tip: *The COPY command can be used to transfer files between devices.*

The TYPE command displays the contents of a single file on your screen. By pressing the CTRL+PRTSC keys before you issue the TYPE command, you can send a single file to your printer. However, the TYPE command is limited, since it handles only one file at a time, and will not accept this kind of syntax: TYPE *.TXT. There are times when you might want to display or print all files in the current directory, one after another. Since TYPE does not accept wildcard characters, you will get the following message in response to this command:

```
File not found
```

The way around this is to use the COPY command. To copy all TXT files to the screen, use the command

```
COPY *.TXT CON:      (CON refers to console)
```

To print all of these files in one operation, enter

```
COPY *.TXT LPT1   (LPT1 refers to parallel printer #1)
```

3.31 Tip: *You can use the keyboard as a typewriter and type directly to the printer from DOS.*

Issue this command:

```
COPY CON: LPT1:
```

Start typing. You might want to create a simple memo. When finished, press the F6 function key and the ENTER key to terminate the process.

3.32 Tip: *If you want to erase files from a disk, you can use either of the two DOS commands, ERASE or DELETE (DEL).*

This is actually one command with two different names. Both will give you the same results: A particular file or group of files will be erased from your disk. The command syntax is DEL *filename.ext* or ERASE *filename.ext*.

You can erase several files by using wildcards; there is no limit to the number of files that will affected by a given command. Consider these wildcard applications:

```
ERASE *.COM
```

All files with the .COM extension will be erased from the current subdirectory.

```
ERASE ??BUDGET.WKS
```

All BUDGET worksheet (.WKS) files, regardless of the first two characters in the name, will be eliminated from the current subdirectory. See the following Trap.

3.33 Trap: *It is easy to erase the wrong file(s), because ERASE (DEL) proceeds without warning.*

It is very easy to accidentally erase files that you did not intend to erase. This can be particularly dangerous when wildcards are included in the command line (such as in ERASE *.*). In this case, DOS tries to prevent disastrous results by offering a precautionary message:

```
Are you sure (Y/N)?
```

However, this prompt is not adequate protection against total disaster. You must be aware of the potential dangers in using the ERASE command, and do so very carefully. You should always look at a directory listing (DIR) of your files before executing the ERASE command so that you can see a list of the files that are about to be erased.

3.34 Trap: *Be careful not to provide a subdirectory name in an ERASE command unless you want to erase all files in that subdirectory.*

Suppose you have a subdirectory called TEST under the root directory. Issuing the command ERASE \TEST would erase all files in that subdirectory. (The subdirectory itself would remain.)

3.35 Tip: *You can use the COMP command to determine whether two files are identical, to check whether a copy is free of defects, or to assure yourself that no damage has occurred since you last made a copy.*

You can also use this command to compare two supposedly identical files that were created at different times. One way to check the accuracy of transmission of a file over a communications line is to transmit it twice (with two different names) and then compare the two files. If they're not the same, you had errors in the transmission.

The command syntax is COMP *filename1.ext filename2.ext*, in which *filename1* is the first file and *filename2* is the second. The filename extension must be specified, or DOS will be unable to locate the files you want to compare and will give you an error message like this:

```
File 1 not found
```

If the comparison is successful, DOS displays this message:

```
Files compare OK
```

When the files are not identical, DOS displays an error message for each location that contains mismatched information in the two files. The message looks something like this:

```
Compare error at offset 0
File 1=61
Compare error at offset 1
File 1=62
File 2=63
```

The message indicates the offset (the number of bytes into the file before the mismatch occurs) and contents of the mismatched bytes, displayed in hexadecimal notation.

After ten pairs of mismatched bytes, DOS ceases execution and displays the message

```
10 mismatches - ending compare
```

If two files have the same name and reside on different disks, you do not have to specify the name of the second file; however, the drive specification is mandatory. If the extension is left off, you will see

```
Enter 2nd file name or drive id
```

If you do not include filenames, or give only one when two are required, the COMP command will prompt you with this:

```
Enter primary file name
```

You might respond with something like this:

```
BUDGET.WKS
```

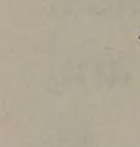
You will then see

```
Enter 2nd file name or drive id
```

Then specify your filename, drive id, or both:

```
B:BUDGET.BAK
B:
```

COMP will then compare the specified files.



3.36 Tip: *You can use wildcards to simplify the COMP procedure.*

You can use wildcards with the COMP command to compare all the files on two disks. The command is issued like this:

```
COMP A:*.* B:
```

This leaves it up to DOS to check both disks for identically named files. Another way to use wildcards with COMP is

```
COMP A:*.EXE B:*.BAK
```

This will cause each file on drive A with the extension .EXE to be compared with any file found on drive B with the same filename but with a .BAK extension.

3.37 Trap: *Files must be the same length to be compared.*

The COMP command won't work at all if one file is larger than another. If you compare files of unequal length, you will see this message:

```
Files are different sizes
```

3.38 Tip: *The RENAME (REN) command makes it possible for you to change the name of one or more files.*

Filenames and extensions can be changed, providing the new name isn't also in use by another file on the same disk. The location of the file on the disk remains unchanged. The command notation is

RENAME *oldname.xxx newname.yyy*

Why would you want to change the name of a file? One reason is that the contents of one of your files may have changed so much that the old name is no longer descriptive. For example, you have a file named SALES.WKS that contains sales data for the first three months of the current year. You decide to maintain quarterly files and select a filename that is more descriptive:

```
A>RENAME SALES.WKS SALESQ1.WKS
```

If you specify a name that already exists, will get this message:

```
Duplicate filename or file not found
```

You will get this same message if you attempt to rename a nonexistent file.

3.39 Trap: *Because RENAME does not change a file's location, destination drive designators are ignored, and if you specify a filename that already exists on the disk, the command will be aborted.*

Since the disk drive designator is ignored for the second filename, these commands produce the same results:

```
RENAME B:BUDGET.WKS B:Q1BUDGET.WKS

RENAME B:BUDGET.WKS A:Q1BUDGET.WKS
```

If you want to rename a file *and* transfer it to another disk or subdirectory, use the COPY command.

3.40 Tip: *If you have two or more files with the same name or extension on a disk, you can use wildcard characters to rename these files as a group.*

DOS 2 allows you to change the names of up to 20 files at once. (The DOS 1 limit is 12 files.) The asterisk (*) takes the place of the remaining part of a filename or extension, and the question mark (?) is used for individual characters. The following examples demonstrate how wildcards are typically used with the RENAME command.

In the first example, all files named ACCOUNTS will be renamed SALES, regardless of extension.

```
A>RENAME ACCOUNTS.* SALES.*
```

The next example renames all files with the extension .DOC to the extension .TXT, regardless of filename.

```
A>RENAME *.DOC *.TXT
```

3.41 Trap: *When you use wildcards with the RENAME command, it's a good idea first to copy your files in a temporary directory on your hard disk, or to a temporary disk.*

Create a temporary disk or directory and copy your files to it before using the RENAME command. After determining that your files have been successfully renamed, erase the contents of the temporary directory.

Using wildcards increases the chance of making a mistake when identifying which source file(s) are to be renamed. For this reason, it's a good idea to create a backup copy of your files before executing this command.

3.42 Trap: *Do not change the names of any files that are used by application programs.*

For example, the word-processing program WordStar will not run unless the file named WSOVLY1.OVR is accessible. If you were to change either the filename or the extension of this file, WordStar wouldn't know the new name.

Another example: Lotus 1-2-3 makes it possible for you to access certain DOS commands while executing the 1-2-3 program. This useful utility would not be possible if DOS commands were renamed, since 1-2-3 is programmed to look for these files by their original names.

3.43 Tip: *Data erased from files is still on the disk and can be recovered with an "un-erase" program.*

The idea that an erased file can be recovered may seem strange. However, when a file is erased, the data isn't destroyed; it is just inaccessible as far as standard DOS commands are concerned. A good analogy is that of a filing folder and wastebasket: If you remove the contents from the folder and throw them in the wastebasket, you could still retrieve the papers if the basket hasn't been emptied. An "un-erase" program can recover your data if no other file has been saved in the same location.

One such data recovery program is included on the Norton Utilities diskette. This set of utilities is worth owning, if just for insurance against accidental file erasure. Sooner or later, you will erase a file that you didn't mean to. Everbody does.

3.44 Tip: *The VERIFY command is used to check that data is accurately recorded when writes to the disk occur.*

When VERIFY is turned on by issuing VERIFY ON, DOS checks all write-to-disk operations for accuracy to be sure that data can be read correctly in the future. VERIFY stays on until you reboot or turn it OFF by issuing the VERIFY OFF or the SET VERIFY system call. Issuing a VERIFY without arguments will display the current VERIFY status.

Since disk drives are generally quite reliable, the VERIFY command is rarely used. However, if you are experiencing disk drive problems, it might be a good idea to use this command. But consider the following Trap.

3.45 Trap: *With VERIFY on, disk drive activity is slowed down. This affects both DOS operations and applications programs.*

If DOS has to verify all disk-writing procedures, many DOS commands and applications programs will run significantly more slowly. Once VERIFY is turned on, it remains in effect until you turn it off or restart your system.

The /V option of the COPY command is usually a better alternative than the global VERIFY command because it is file-specific.

4

Managing Disks And Directories

This chapter deals with managing groups of files, whether on a floppy system or on a hard disk system. Since the floppy and hard disk cases differ significantly, we include a special section on managing hard disks.

General Disk Management

FORMAT Command

4.01 Trap: *You must have your DOS diskette in a floppy drive (floppy system) or establish a path to the FORMAT.COM file (hard disk system) before issuing the FORMAT command.*

Some commands (called "internal") are part of the main DOS program, COMMAND.COM, which is loaded into memory when you boot your system. Others (called "external") reside in separate files on your DOS diskette. The FORMAT command resides on the diskette as an external file, FORMAT.COM, which means

that on a floppy system your DOS disk must be in a floppy drive. If the FORMAT.COM file is in a drive other than default, you must prefix the command name with the drive designation. On a hard disk system, you must issue the FORMAT command from either the directory where FORMAT.COM resides (usually the root) or a subdirectory that has a path established to FORMAT.COM. If the FORMAT.COM program file can't be found, then DOS will respond with the following message:

```
Bad command or filename
```

DOS is telling you that it was unable to locate the program file required to execute the command you entered. Of course, if you incorrectly spelled the name of the command, then you will also see this message, since DOS will be looking for a file with the name as you typed it.

4.02 Tip: *You can prepare a program diskette so that it can be used to start up your floppy system.*

It is often convenient on a floppy system to place a copy of DOS on program diskettes that you use again and again. That way, you can use the same diskette both to boot your system and to load a program (e.g., Lotus 1-2-3, dBASE III, and so on). A diskette prepared in this manner is sometimes referred to as a *system disk;* it can be used to start up the system and initiate a program, all in one step. To use this formatting option, issue the following command:

```
A>SYS B:
```

The SYS command transfers from your DOS diskette to your target diskette the two hidden files that are responsible for booting your system. These files must be the first files read and, therefore, must be placed on the first two tracks, Tracks 0 and 1. Before using the SYS command, then, you must ensure that these tracks are blank.

If you purchased a commercial software product designed to have DOS files added by the user, the manufacturer will have left the appropriate directory locations blank so that the SYS command will function properly. Because DOS is not included on commercial software, users can select the version of DOS that best meets their needs or change to an updated version. Manufacturers and users alike can prepare diskettes with space reserved for system files by issuing the /B option of the FORMAT command, FORMAT B:/B. The /B parameter causes FORMAT to produce a diskette that is compatible with all versions of PC-DOS: The two hidden DOS files and the command processor program are not placed on the disk at the time of

formatting, and Tracks 0 and 1 are left blank. After issuing the command, DOS responds with

```
A>FORMAT B:/B
Insert new diskette for drive B:
and strike any key when ready

Formatting...Format complete
  322560 bytes total disk space
    8704 bytes used by system
  313856 bytes available on disk

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

If you have a blank diskette that you want to format as a boot disk, you can use the /S option. This option transfers the hidden DOS files and the command processor file at the same time the diskette is formatted. This option should *never* be used on a commercial program since formatting erases all information. Command syntax is A>FORMAT *d:*/S, in which *d* stands for the drive containing the disk you want to format. Assuming that you want to format the disk in drive B, you will respond to the following prompt:

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

Formatting...Format complete
System transferred

  362496 bytes total disk space
   40960 bytes used by system
  321536 bytes available on disk

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

When formatting disks for data storage, having DOS on the disk takes up valuable storage space. A good rule of thumb is to include DOS on all program diskettes if there is room, but format without the /S option when creating data diskettes.

If you format a diskette without the optional /B or /S parameters, you cannot add the DOS files later unless you reformat, which would destroy any programs or data currently stored on the diskette. When you use the FORMAT command, nothing is issued to warn you that existing files will be destroyed.

4.03 Tip: *Using FORMAT together with COPY instead of DISKCOPY to copy a diskette makes it possible to use a diskette, even if some of the sectors are bad.*

The FORMAT command will check and verify all sectors. If a sector cannot be read by the system, then it is masked out, and no data is ever written to it. DISKCOPY, on the other hand, does not perform this check. This means that DISKCOPY may write data to bad sectors, and that this data may later be unreadable.

4.04 Trap: *Never format a commercial software program diskette.*

When you install a new software program, you are often asked to format a diskette. This must be a blank diskette that can then be used to copy program files onto it. If you mistakenly format a program diskette, the program files will be erased. Unless you have a backup copy, one way to rectify this is to contact the software manufacturer to obtain a new program diskette. Another way is to make sure that you have placed write-protection tabs on all your program disks.

4.05 Tip: *The FORMAT command's optional switches make it possible to format diskettes for use with older model PCs.*

The original IBM PCs were equipped only with single-sided disk drives. DOS 1.0 (the first version of DOS) therefore used only one side of a diskette; it also accessed only eight of the nine available sectors. The FORMAT command in DOS-2 includes the /1 and /8 switches. The /8 option instructs DOS to format a diskette so that only eight sectors per track will be usable, and the /1 option formats a diskette for single-sided use. If you have an older model IBM PC or diskettes formatted with DOS 1.0, these options could be handy. You may also need your DOS-2 diskettes to be compatible with DOS-1 diskettes on another system (at work or a friend's), and vice versa.

This is the FORMAT command if you were formatting a disk in a single-sided disk drive:

```
A>FORMAT B:/1/8
Insert new diskette for drive B:
and strike any key when ready

Formatting...Format complete

    160256 bytes total disk space
    160256 bytes available on disk

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

The number of available bytes is 160K, not 360K, because your diskette is formatted on just one side, with only eight sectors instead of nine.

4.06 Tip: *You can assign a volume label to your diskettes, giving you an internal as well as external label.*

When you format a diskette, you should affix a label identifying the files. It is also possible to name your diskette internally by employing the /V switch during the formatting process. The "V" stands for volume, which is synonymous with diskette.

```
A>FORMAT B:/V
Insert new diskette for drive B:
and strike any key when ready

Formatting...Format complete

Volume label (11 characters, ENTER for none)? ACCOUNTING

   362496 bytes total disk space
   362496 bytes available on disk

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

Whenever DOS commands interact with a diskette, the volume name is referenced. This feature is available with DOS-2 and DOS-3 only. The label can go up to 11 characters in length and it is best to use only alphabetic and numeric characters. Once you have named a diskette, you cannot remove or change the name unless you either reformat or use the DOS 3.X LABEL command.

When formatting a hard disk, the /V switch can specify a volume label to identify the hard disk by name. Again, under DOS-2, a volume label can be specified only during the formatting process, and it cannot be changed unless the hard disk is reformatted.

4.07 Trap: *You must be aware of the potential dangers of the FORMAT command: One mistake and all is lost!*

When executing the FORMAT command, be very careful to specify the correct drive, or you might erase programs or data by mistake. This is an irrecoverable error — all of your information is lost forever. This warning applies to both diskettes and hard

disks. You must also be careful to issue the FORMAT command with a drive designation. Otherwise, the current drive default will be assumed.

To avoid disaster on a floppy system, be sure to use a copy of your DOS diskette, not the original. It is also a good idea to issue the FORMAT command through a batch file. Refer to the next Tip for a suggested batch file and see Trap 4.38 for a batch file to use with a hard disk system.

4.08 Tip: *You should use the FORMAT command through a batch file to protect against inadvertently formatting valuable data.*

With so many potential hazards associated with the FORMAT command, we highly recommend that you use a batch file to protect against formatting over existing programs and data. The following batch file first checks the disk to determine if files exist. You are then given the option either to continue the formatting process or stop execution.

```
REM         Before FORMATTING, check files.
CHKDSK B:
DIR B:
PAUSE       If there are valuable files, CTRL/BREAK to stop!
FORMAT B:
```

A batch file like this offers some protection for your data: You are reminded to check for the existence of valuable files before making a disastrous error. In addition, you should always label your diskettes so that you know what was saved on them.

4.09 Trap: *Don't use the FORMAT command if you use DOS 2.0 with a date before 10/83. You need to obtain an update from IBM, or you may find that you can't retrieve your files.*

The FORMAT program on DOS 2.0 with a date before 10/83 does not correctly mask out bad sectors. This could result in files being written to bad areas on the disk. You will receive read errors when you attempt to retrieve them and lose the data. IBM has since corrected the problem and will supply you with what they call a ZAP disk free of charge.

4.10 Tip: *You can create a batch file that will let you format an entire box of diskettes without having to sit and stare at the screen during this long, tedious process.*

After each diskette is formatted by the FORMAT command, DOS waits for you to respond to the prompt:

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

Ordinarily, you must sit and watch for this prompt on your screen to know when the previous disk has been formatted. You can, however, create a batch file that "front-ends" the FORMAT command and beeps each time it is ready to format the next diskette. Named FORMALL, this batch file will free you to work on other things while your diskettes are being formatted:

```
:START
TYPE BEEP
PAUSE ***INSERT DISKETTE IN DRIVE B:***
FORMAT<NO B:
GOTO START
```

The colon (:) precedes the label, START, and serves as a marker for the GOTO command at the end of the batch file. The BEEP file referred to in the second line contains nothing but the beep character (ASCII 007). You can create it like this:

```
A>COPY CON: BEEP
<ALT--007>
```

(This line is created by holding down the ALT key and, using the numeric keypad, pressing the numerals 0 0 7)

```
<F6> <Enter>
```

The ALT-007 combination inserts an ASCII 007 into the file. When the TYPE BEEP command in the second line of the batch file is executed, the beep sound is produced, reminding you to insert a new diskette. Be sure to use the numeric keypad to key in these three digits; otherwise, you won't get the beep.

The FORMAT<NO B: command in the fourth line formats the diskette in drive B; keyboard input is supplied automatically by input redirection (see Chapter 6) from a file called NO, created earlier. The file NO contains only the lines X and N. The X, which could be any character, is a response to the "Insert new diskette for drive A: and strike any key when ready" prompt from the FORMAT command. The N is a response to the "Format another (Y/N)?" prompt from the FORMAT command. You need to intercept and pass automatically through these two prompts to create your own formatting loop complete with a beep; the input redirection from the NO file achieves this for you.

Since the last line of the batch file (GOTO START) automatically causes the process to repeat, you need to press CTRL BREAK in response to the PAUSE command on the third line when you are done formatting disks and want to stop the batch file.

DIR Command

4.11 Tip: *Use the DIR command to determine the amount of available space on your disk if you need to copy certain files to one diskette.*

Suppose you want to copy related files to one diskette, and you aren't sure whether there is sufficient disk space for all the files. You can use the DIR command to compare the total number of bytes of the files to be copied with the number of available bytes on the destination diskette.

4.12 Tip: *Using wildcards with the DIR command, you can list selected filenames that are related by content.*

If you want to get a directory listing of a group of related files, use the DIR command with wildcards, * and ?. The asterisk (*) is a substitute for any group of characters in the remaining portion of the filename or file extension. The question mark (?) takes the place of any single character. The following demonstrates how to get a selective directory listing with a wildcard specification:

```
A>DIR *.EXE

 Volume in drive A has no label
 Directory of  A:\

SORT      EXE      1408   10-20-83   12:00p
FIND      EXE      5888   10-20-83   12:00p
        2 File(s)      28672 bytes free
```

4.13 Tip: *If your list of files is too long to fit on a single screen display, use /P or /W with the DIR command.*

If your directory listing is too long to fit on a single screen display, the DIR command can be modified with the /P and /W switches. /P causes a directory to pause after it fills the screen. /W creates a directory with only the names of the files. Date and time stamps and the size of the file parameter are suppressed. The following is a directory listing of the DOS 2.1 diskette using the /W switch:

```
A>DIR/W

 Volume in drive A has no label
 Directory of  A:\

COMMAND  COM     ANSI     SYS     FORMAT   COM     CHKDSK   COM
DISKCOPY COM     DISKCOMP COM     COMP     COM     EDLIN    COM
FDISK    COM     BACKUP   COM     RESTORE  COM     PRINT    COM
ASSIGN   COM     TREE     COM     GRAPHICS COM     SORT     EXE
MORE     COM     BASIC    COM     BASICA   COM     SYS      COM
MODE     COM     RECOVER  COM     FIND     EXE
      23 File(s)     28672 bytes free
```

Using the /W switch is particularly helpful when you need to compare directory listings of more than one diskette without overflowing the screen.

4.14 Trap: *When you issue the DIR command without a drive specification, it lists the contents of the current default drive/subdirectory only.*

This could be dangerous if you are not careful, especially if you are using the DIR command to determine whether you can safely format a particular diskette. If you use a hard disk system, the default drive will be C (hard disk) unless you specify otherwise. If you want to obtain a directory listing of a floppy diskette, type **DIR A:.**

4.15 Tip: *You can create a batch file that will sort your directory filenames and pause them after every screenful.*

This batch file, called ASORT, automatically sorts your filenames alphabetically and displays only one screenful at a time:

```
A>COPY CON: ASORT.BAT
DIR|SORT|MORE
^Z
        1 File(s) copied
```

Using this batch file to get a directory listing of the DOS 2.1 diskette, you will see the following on your screen:

```
A>ASORT
A>DIR|SORT|MORE
       25 File(s)     202752 bytes free
 Directory of  A:\
```

```
 Volume in drive A has no label
ANSI     SYS      1664   10-20-83   12:00p
ASSIGN   COM       896   10-20-83   12:00p
BACKUP   COM      3687   10-20-83   12:00p
BASIC    COM     16256   10-20-83   12:00p
BASICA   COM     26112   10-20-83   12:00p
CHKDSK   COM      6400   10-20-83   12:00p
COMMAND  COM     17792   10-20-83   12:00p
COMP     COM      2534   10-20-83   12:00p
DISKCOMP COM      2188   10-20-83   12:00p
DISKCOPY COM      2576   10-20-83   12:00p
EDLIN    COM      4608   10-20-83   12:00p
FDISK    COM      6369   10-20-83   12:00p
FIND     EXE      5888   10-20-83   12:00p
FORMAT   COM      6912   10-20-83   12:00p
GRAPHICS COM       789   10-20-83   12:00p
MODE     COM      3139   10-20-83   12:00p
MORE     COM       384   10-20-83   12:00p
-- More --

PRINT    COM      4608   10-20-83   12:00p
RECOVER  COM      2304   10-20-83   12:00p
RESTORE  COM      4003   10-20-83   12:00p
SORT     EXE      1408   10-20-83   12:00p
SYS      COM      1680   10-20-83   12:00p
TREE     COM      1513   10-20-83   12:00p
```

The ¦ character used here activates the DOS piping facility, which automatically passes the output of one program as input to another. SORT and MORE are DOS "filter" programs. See Chapter 6 for more on piping and filters.

4.16 Tip: *You can create a batch file that will sort your directory filenames by date.*

This batch file, called DSORT, automatically sorts your filenames by the date they were saved, with the most recently saved files at the bottom of the list:

```
A>COPY CON: DSORT.BAT
DIR|SORT/+24|MORE
^Z
        1 File(s) copied
```

The /+24 parameter instructs DOS to key on the 24th character (the first digit in the date entry) for sorting the files. Using this batch file on a data diskette, you will see something like the following:

```
A>ASORT

A>DIR|SORT|MORE
       25 Files(s) 202752 bytes free
 Directory of A:\
 Volume in drive A has no label
BUDG85     WRK  1664       2-20-86   9:24a
PROP2      WRK   896       3-13-86  18:02p
LJG        DOC  3687       4-03-86  10:37p
JMG        DOC 16256       7-13-86   9:21a
PROP1      WRK 26112       8-04-86  21:30p
DNG        DOC  6400      11-07-86  11:29a
REPORT     WRK 17792      11-26-86   4:44a
YTD        WRK  2534      12-31-86  12:00p
```

DISKCOPY Command

4.17 Trap: *You can use DISKCOPY for quick and easy diskette duplication; however, you should be aware of potential problems.*

When you issue the DISKCOPY command, DOS responds with the following prompts:

```
A>DISKCOPY A: B:

Insert source diskette in drive A:

Insert target diskette in drive B:

Strike any key when ready

Copying 9 sectors per track, 2 side(s)

Copy complete

Copy another (Y/N)?N
```

DISKCOPY is often recommended as a quick and efficient way to create copies of diskettes; however, there are some potential traps. The DISKCOPY command reads all the formatting and information from one diskette and creates an exact duplicate on another diskette. If either the disk you are copying from or the one you are copying to contains bad sectors, then the DISKCOPY procedure will not be successfully completed. On the other hand, COPY handles bad spots by simply skipping over them, and the procedure is not interrupted.

DISKCOPY erases all previously saved files from the destination diskette. If your original disk has been subjected to considerable file creation and erasure, large blocks of contiguous sectors are no longer available. This results in DOS being forced to save files in fragments, significantly lengthening file access time, because it takes longer to find all the segments. Since DISKCOPY creates an exact duplicate of a diskette, files on the copy will also be fragmented. To avoid this, use COPY *.*, which will copy the files in the most efficient fashion, and fragmentation will be eliminated as much as possible. It is almost always better to use COPY *.* rather than DISKCOPY to create backup copies.

If you are using a hard disk system, you have to remember to use COPY *.* on each directory; the command can reference files in only one directory at a time. Refer to Chapter 3 for additional Tips and Traps on using the COPY *.* command.

4.18 Trap: *Using DISKCOPY always formats your diskette into the format of the original source diskette, regardless of any previous formatting on your target diskette.*

Most software manufacturers market their products to be compatible with any IBM PC system, and the lowest common denominator is the eight-track single-sided format. Software installation procedures instruct you to use your own blank diskette on which to DISKCOPY the program. Suppose this diskette is already formatted in nine-track double-sided format. DISKCOPY automatically formats your diskette just as the source diskette is formatted, ignoring any previous target disk formatting. If the commercial program diskette is in eight-sector single-sided format, DISKCOPY will duplicate this format to the destination diskette. This may not be desirable if your version of DOS uses nine-sector double-sided formatting, since only 160K bytes will be available for storage instead of the maximum of 360K bytes.

4.19 Tip: *You can use DISKCOPY to detect whether a diskette is damaged.*

You can use DISKCOPY to determine whether a diskette has been physically damaged. If you can DISKCOPY a diskette without errors, then you can usually conclude that it is not damaged. But if you encounter a bad spot on the diskette, the following message will appear on your screen:

```
DISK ERROR READING
```

4.20 Tip: *You can use the /1 option with DISKCOPY to create a single-sided diskette.*

Like FORMAT, the DISKCOPY command includes a /1 option that saves information on just one side of the diskette.

CHKDSK Command

4.21 Tip: *Use the CHKDSK command to obtain a status report on available disk space and system memory.*

CHKDSK is a very powerful DOS command and offers a lot of information about your disk. CHKDSK tells you two main things: how much disk space is available for saving files and the amount of available system memory (RAM). If DOS reports a lack of disk space or a problem reading a disk, your best diagnostic tool is the CHKDSK command. Let's look at a sample listing produced by CHKDSK:

```
A>CHKDSK

   362496 bytes total disk space
    22528 bytes in 2 hidden files
   311296 bytes in 36 user files
    28672 bytes available on disk

   655360 bytes total memory
   568112 bytes free
```

Here, the first group of lines indicates total disk space, the number and total size of hidden files, the number of user files and total number of bytes used to store them, and the remaining disk space. If your disk contains subdirectories, CHKDSK will also list the number of directories and total bytes consumed by them. The last two lines report on the total amount of RAM storage available in your system and the amount that is currently not in use.

4.22 Tip: *Use CHKDSK to determine whether a diskette is formatted as a boot disk.*

Using the CHKDSK command can be an easy way to determine whether a disk is system-formatted: System files are referred to as *hidden files* when you get a

CHKDSK status report. Refer to the following table:

Number of Hidden Files	Is Disk DOS System Disk?
0	No
1	No
2	Yes
3	Yes

See the next Trap for an explanation on why DOS might report the existence of a third hidden file.

4.23 Trap: *If CHKDSK indicates the existence of a third hidden file, don't be concerned. Your disk was formatted with the /V switch, and CHKDSK identifies the VOLUME ID label as a hidden file.*

When you format and name a disk with the FORMAT/V command, DOS records the Volume ID label as a hidden file. When you run CHKDSK on a start-up disk that has a volume name, DOS identifies a total of three hidden files instead of the two you might expect.

4.24 Tip: *If the disk-writing process is interrupted (e.g., when saving a file), access possible damage with CHKDSK.*

Suppose that you are saving a file and the electricity shuts off before the process is completed. Data can be destroyed if the process of writing to a disk is interrupted before completion. Usually damage is confined to one file, but it is possible for the entire disk to become scrambled. If the CHKDSK command does not return error messages, you can usually conclude the disk as a whole is in good shape.

You should then use the DIR command to determine whether the file that was being saved when the interruption occurred is in fact complete. Check the size of the file: If zero bytes or a number much smaller than anticipated is reported, then you know that the file was not successfully saved. You may have to repeat the saving process.

4.25 Trap: *Using CHKDSK does not always guarantee that bad sectors will be detected and reported.*

It is possible for CHKDSK to report no errors and yet have bad sectors exist on the disk. Only when you try to read a file that has been saved on the damaged part of the disk will you be made aware of bad sectors. The CHKDSK command can only determine whether the file directory is complete and all entries are consistent.

4.26 Tip: *You can use the CHKDSK command to determine whether an entire file has been saved in contiguous sectors; fragmented files increase retrieval time.*

The CHKDSK command can determine whether an entire file has been saved in contiguous sectors. Fragmented files are inefficient, with access time greatly increased. The command syntax is CHKDSK *filespecification*. This procedure addresses one file at a time as DOS checks whether a particular file is contiguous. More than likely, DOS will respond with the following message:

```
All specified files are contiguous
```

If DOS indicates that your file(s) are fragmented, you can easily correct this problem with a blank formatted diskette and COPY *.* all files to the new diskette. Do *not* use DISKCOPY to transfer the files. Refer to Trap 4.17.

4.27 Trap: *Using CHKDSK with the /F argument to fix disk errors could result in losing information; be sure to back up your files first.*

When your files are written to disk, DOS uses the File Allocation Table (FAT) to note the disk address of files. Without this information, you cannot retrieve your files. Sometimes the FAT becomes corrupted and doesn't accurately track files on a disk. The "Data Error Reading" message often indicates this type of problem. CHKDSK can detect problems with the FAT table, and the /F argument can try to fix errors.

However, the FAT option could result in a loss of data. When DOS attempts to fix disk errors, parts of files are sometimes deleted. Should you suspect disk errors,

first run CHKDSK without the /F option to learn if errors indeed exist. If errors are reported, then use COPY *.* to create a backup disk. Run CHKDSK again on your original disk, this time including the /F argument. Refer to the next Tip.

4.28 Tip: *If CHKDSK finds errors, use the /V option to obtain a detailed analysis, including the location of errors.*

The /V option completes the examination of your disk by specifying where the errors are located. It causes the CHKDSK command to work in what is referred to as *verbose mode,* with the affected directories and files listed by name. Refer to the previous Trap.

Managing Hard Disks

Formatting and Partitioning

Before you can use a new hard disk, you must use the Fixed Disk Setup Program (FDISK) command. This command is used only with a hard disk and is found on DOS-2 and DOS-3 versions; DOS-1 does not contain the FDISK command, because it was not designed to run a hard disk system. (It may be that someone else in your organization or the vendor from whom you purchased your hard disk has already formatted it. If you get a C> prompt when you boot, then the disk has already been formatted, and you can skip the rest of this Tip.)

You use the FDISK command to divide your fixed disk space into separate areas called *partitions,* so that one or more operating systems can be used. Different operating systems can't share the same disk space, but they can coexist on a hard disk if the disk has been partitioned. The following is a list of rules governing the creation and management of partitions:

- You can create from one to four partitions.
- Partitions can be different sizes and set up in any order.
- You can specify the default partition, which is the partition DOS will access when you start or restart your system.
- An operating system can access only one partition.
- You cannot transfer data directly from one partition to another.

Since DOS is used almost exclusively by PC users, the following exercise will

create a single partition dedicated to DOS. Put DOS-2 or DOS-3 in drive A and then issue the FDISK command. DOS will respond with

```
IBM Personal Computer
Fixed Disk Setup Program Version 1.00
(C)  Copyright IBM Corp. 1983
```

Depending on your version of DOS, the program version number and date may differ. DOS goes on to display the available options:

```
FDISK Options

Choose one of the following:

1.Create DOS Partition
2.Change Active Partition
3.Delete DOS Partition
4.Display Partition Data

Enter choice: [1]

Press Esc to return to DOS
```

Accept the default, option 1, by pressing the ENTER key, because you want to create a DOS partition. You will see

```
IBM Personal Computer
Fixed Disk Setup Program Version 1.00
(C) Copyright IBM Corp. 1983

Create DOS Partition

Do you wish to use the entire fixed
disk for DOS (Y/N).....................?  [Y]
```

Press ENTER because you want to dedicate your entire fixed disk to DOS. The next prompt is

```
Insert DOS diskette in drive A:
Press any key when ready . . .
```

DOS should still be in drive A, so you must press a key to complete the process. The next step is to format the DOS partition so that it is usable. Issue the following command:

```
FORMAT C:/S/V

Press any key to begin formatting drive C:

Formatting . . .
```

The formatting procedure takes time. Eventually, you will see

```
Format complete
System transferred
Volume label (11 characters, ENTER for none)?
```

You can enter a label for the hard disk or press the ENTER key if you don't want to label it. Remember, you cannot add a volume label later with DOS-2. You will now want to transfer the remaining files from your DOS diskette to the hard disk and enter **COPY *.* C:**.

You have now completed the processes of partitioning, formatting, and copying the DOS utilities onto your hard disk. To determine whether your hard disk is functional, remove the DOS diskette from drive A, and press the CTRL, ALT, and DEL keys simultaneously to restart your system. If all goes well, you will be prompted to enter the date and time, and then you will see your hard disk prompt:

```
C>
```

Your hard disk is now ready to use; you will be able to boot your system from the hard disk without having to use a floppy diskette copy of DOS.

Warning: Once you have created a DOS partition and saved a file to the hard disk, be careful not to destroy it with the FDISK program. If you use FDISK to change your partition(s), you may lose everything stored on the entire hard disk. Be careful!

The DOS Hierarchical Directory Structure

4.29 Tip: *Use subdirectories to organize your files on a hard disk.*

DOS uses a directory to keep track of the name and location of the files on your disk. Think of the relationship between a book and its table of contents; the same parallel relationship applies to a disk and its file directory. Just as you would refer to the table of contents of a book to locate a particular chapter, DOS refers to the file directory to find out where a particular file is stored on the disk. Before DOS-2, a single directory was adequate for managing files on diskettes; each diskette contains only one directory.

When you format a floppy diskette with either DOS-1 or -2, a single directory is created (known as the root, or system, directory) and can store up to 112 files. To search for a particular file, you issue the DIR command and scan the directory shown on the screen. If the number of files on the diskette is close to capacity (112 files), you

may need as many as four screenfuls to display the entire directory listing. Though somewhat bothersome, this is still manageable. However, with the introduction of hard disks, the problem is greatly magnified: Ten megabytes of storage lets you store hundreds or even thousands of files. It takes minutes instead of seconds for hundreds of files to scroll by on the screen, and many files could have similar filenames, which makes identifying a specific file almost impossible. It's also much harder to back up or clean up a particular group of files if they're mixed up with many others in the same directory.

With the release of DOS-2, IBM solved the problem by adopting a tree-structured file directory system. The main, or root, directory branches into subdirectories, which, if you desire, can contain more subdirectories. You assign each subdirectory a name, following the same rules as with filenames. You can use subdirectories on both floppy diskettes and hard disks. However, under most circumstances, subdirectories aren't useful on floppies: The additional work involved for DOS to locate files in a subdirectory structure significantly slows down the file-retrieval and file-saving procedures on a floppy diskette. Because hard disks work substantially faster than floppy diskettes, the subdirectories on a hard disk are worthwhile and highly recommended.

Subdirectories allow you to organize your programs and files electronically, much as you would manually by using separate disks with labels on them. The labels are the subdirectory names; however, you are not limited to the 360K capacity per diskette.

You can use the following DOS commands to create and maintain subdirectories:

MKDIR (MD)	Makes a new subdirectory.
CHDIR (CD)	Changes to a subdirectory other than the current one.
RMDIR (RD)	Removes an existing subdirectory.
TREE	Displays complete directory tree from a disk.
TREE/F	By adding /F can use TREE to get listing of files in each subdirectory.
PATH	Instructs DOS where to look for program files (.COM, .EXE, or .BAT file extensions). Can be used to tell DOS to search in several directories and on several disks.

With a hierarchical file structure, you cannot access any file just by entering the filename and extension. You must specify the path for DOS to look for the file if it resides in another subdirectory. The PATH command instructs DOS to find a particular subdirectory by identifying the route to follow from one directory to

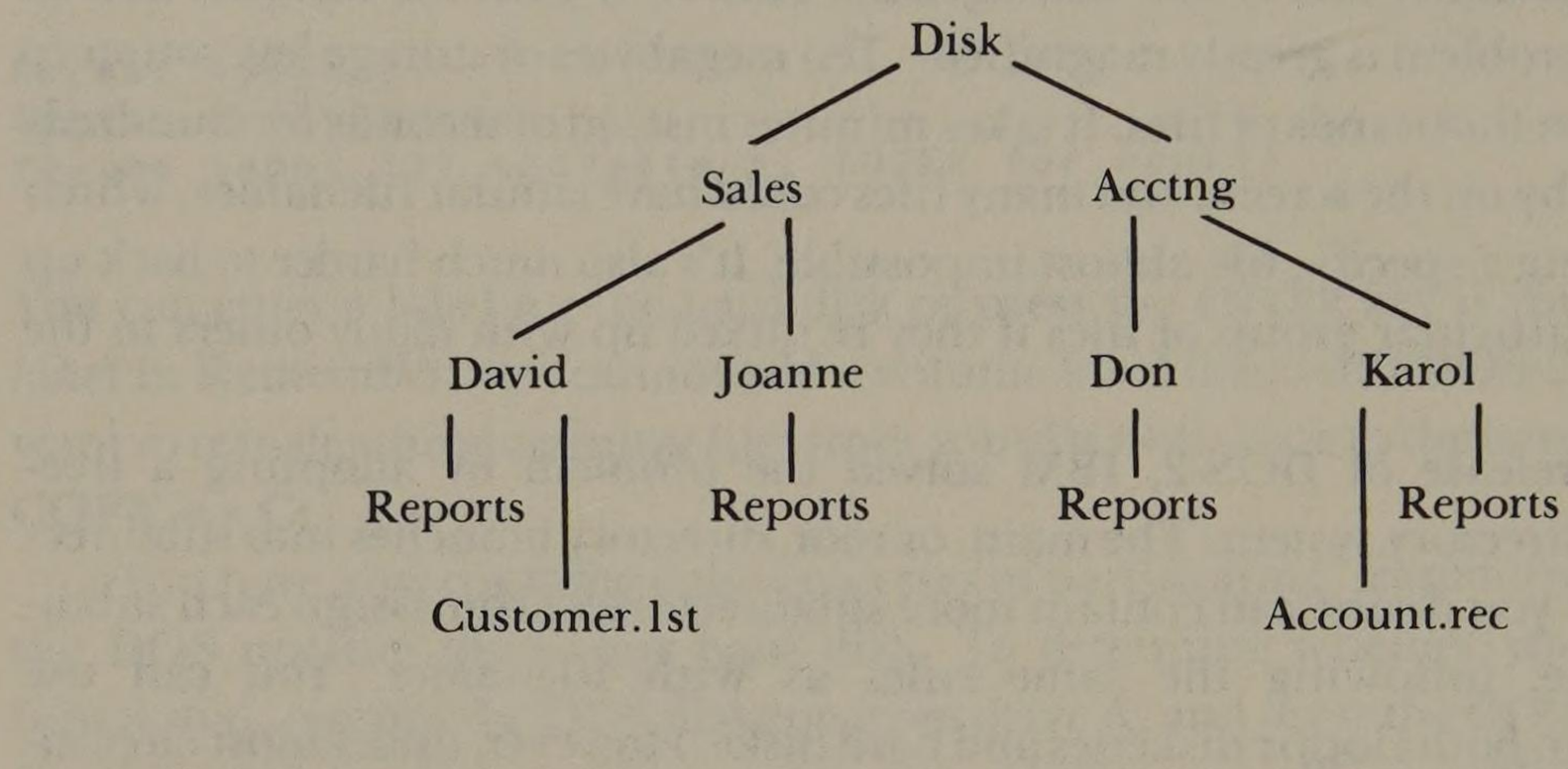


Figure 4-1. Sample subdirectory structure

another. Consider the sample subdirectory structure in Figure 4-1. Suppose that you want to access one of Joanne's reports. The path could be described something like this:

1. Start with Disk (the root directory)
2. Locate the subdirectory named Sales
3. Locate its subdirectory named Joanne
4. Locate its subdirectory named Reports
5. Locate the file named Summary (one of Joanne's reports).

We can simplify this with the path name \DISK \SALES \JOANNE \REPORTS \ SUMMARY.

Notes about using the PATH command:

- A path name consists of a list of subdirectory names separated by backslashes (\). If a filename is listed at the end of a path name, it must be separated from the last subdirectory by a backslash.
- Start the path name with a backslash if you want DOS to start with the root directory. If you omit the backslash that denotes the root directory, DOS assumes that you want to start with the current directory, which may or may not be the root directory.
- Issuing a PATH command does not change the current directory.

- Alternate search paths specified by a PATH command are in effect until you issue another PATH command.
- Unless a PATH is specified, DOS expects to find programs you ask it to execute in the current directory of the default drive.
- You can use PATH to lead DOS either to a file or a subdirectory.
- DOS always begins searching for a particular program in the current directory; however, if the file isn't found there, then DOS continues to search, following any paths you have specified with the PATH command.
- Since DOS keeps track of the current directory, you need not write the path name each time you use a file in the current directory.
- The specified path from the root directory to the subdirectory or file must be less than 63 characters long, including the backslashes. This is the full path, not just a path segment. If the full path exceeds 63 characters, you cannot add a new subdirectory.

4.30 Trap: *Do not assume that the MKDIR (MD) command affects the current directory setting.*

Using the MKDIR (MD) command to create a new subdirectory does not change the current directory setting. If you want to access the newly created subdirectory, you need to issue the CHDIR (CD) command. The following sequence of commands first creates a new subdirectory and then makes it current:

```
MD\ACCTG\AR

CD\ACCTG\AR
```

4.31 Trap: *You cannot remove a subdirectory until you first erase all files contained in that subdirectory.*

The RMDIR (RD) command allows you to remove subdirectories; however, there are two restrictions on what directories can be removed:

- The directory must be empty; all files must first be erased.
- Neither the root directory nor the current directory can be removed.

4.32 Trap: *You cannot use the PATH command to search for data files; PATH can only be used to search for files with .COM, .EXE, and .BAT extensions.*

The PATH command instructs DOS to search subdirectories other than the current one for a program, command, or batch file you have requested. Unfortunately, the PATH command is limited just to files with the .COM, .EXE, and .BAT extensions. This means data files can't be located in other subdirectories with the PATH command. If you are using a program that doesn't recognize path names, you must place both program files and data files in the same subdirectory. If you don't, your program will be unable to access related data files, and the PATH command won't help.

4.33 Trap: *If you do not specify the previous level subdirectory to be the current one before creating a new subdirectory, the new subdirectory will be created at the wrong place in the hierarchy.*

Consider the subdirectory structure in Figure 4-1. Suppose that you want to create a subdirectory, LETTERS, at the fourth level under Joanne. Before you create the new directory, you first make Joanne the current directory. If you don't, then the command MD LETTERS will cause LETTERS to be a subordinate subdirectory to the current directory, which will not be Joanne, the one you want. You can also make sure that the new subdirectory ends up in the same position by specifying the full path name:

```
MD \DISK \SALES \JOANNE \LETTERS
CD \DISK \SALES \JOANNE \LETTERS
```

This method is of course more cumbersome than the command MD LETTERS, issued from JOANNE.

4.34 Trap: *If your subdirectory structure is too complex, DOS slows down, and you may have difficulty maintaining your directories.*

To use a hard disk system successfully, you need to create a simple directory tree, rather than one with an elaborate and complex organizational scheme. The main problem you face is deciding when to separate files into distinct subdirectories and when to keep them together in the root directory. To some extent, your particular

application will dictate your tree structure. However, you should follow a basic strategy: Keep it simple!

Let's consider the sample tree structure shown in Figure 4-1, which is a tree-shaped subdirectory structure that appears to be well planned and effective. However, a four-level tree like this is extremely inefficient in organizing a hard disk. Just to change default subdirectories requires extensive typing: Suppose that you want to change from the ACCOUNTS.REC on the fourth level to the REPORTS subdirectory under KAROL. You must type the command **CD \DISK ACCTNG \KAROL \REPORTS**.

Changing subdirectories is a frequent occurrence, and you will want to simplify the process by keeping the amount of typing to a minimum. With multitiered subdirectories, it becomes increasingly difficult to recall where your different programs and corresponding data are stored. The search process can be lengthy, and it is possible to lose a particular file on the limbs of a complex tree structure.

The main reason for creating a subdirectory is to track certain files in it. While DOS does a good job keeping track of hundreds of files in a single directory, we humans simply are incapable of doing so ourselves. By grouping related files into a separate subdirectory of more manageable size, you can readily identify any files that are missing or in need of updating, and so forth. However, if you don't really need to track a group of files, then don't create a separate subdirectory for them.

4.35 Tip: *By putting all your subdirectories under the root directory, you maximize efficiency and accessibility.*

A complex tree structure that matches the logical relationship of your files sounds like a good idea; however, it doesn't work well. There are two major problems with grouping files in a multitiered subdirectory scheme: performance and tracking. DOS has to work harder when dealing with a complex tree structure, which reduces response time when you save and retrieve files. You have to work harder to keep track of your directories when you put subdirectories under subdirectories. The TREE command is supposed to assist you in tracking your directories; however, it is a limited, poorly written utility.

There are two possible solutions: One is to create a menu structure to handle the task of tracking subdirectories and paths. For a detailed description of how to set up this type of structure, see Tip 4.36. A second approach, which is highly recommended if a menu structure isn't used, is to put all your subdirectories under the root directory. If you use the PATH command to keep the path set to C: \ in all subdirectories, you have immediate access to all programs in the root directory, whatever the current subdirectory.

Usually you want to create a separate subdirectory for every group of data files that are part of the same application. One such case is an accounting system created with Lotus Symphony that requires access to multiple data files and constant switching between them. It is more effective to keep these related files in a separate subdirectory: You can easily track your latest accounting transactions by checking date entries on a directory listing to be sure that your program file has immediate access to necessary data files. You can also easily delete outdated files, perform backups, and so forth, if all the files related to a particular application are in the same subdirectory.

4.36 Tip: *You can create a hierarchical menu structure to help you navigate a multilevel hierarchy of subdirectories.*

As the previous Tip suggests, you may have difficulty finding your way around a subdirectory structure with more than one level below the root directory. You may not remember the exact structure, and find yourself using the Change Directory command with long path names to go from one subdirectory to another. You can, however, create a batch file system that produces a hierarchy of menus to match your hierarchy of subdirectories so that you can easily navigate from one to the other. The menu structure we will use as an example can access three different programs and their associated data subdirectories: Symphony, dBase III, and Timeline. Figure 4-2 shows the subdirectory structure on the hard disk we're using. In addition to these three program choices, a fourth allows the user to return easily to DOS control. Setting up this menu structure involves creating a series of simple batch and text files. The first one is an autoexec batch file that boots your system and then displays a root directory menu from which to select the program you wish to run:

```
DATE
TIME
CLS
TYPE MENU.TXT
```

Before you can execute this autoexec file, you need to create the MENU.TXT file. In our example, the text file looks like this:

```
                    ROOT DIRECTORY MENU

1. SYMPHONY.

2. DBASE III.

3. TIMELINE.

4. END CURRENT SESSION.
```

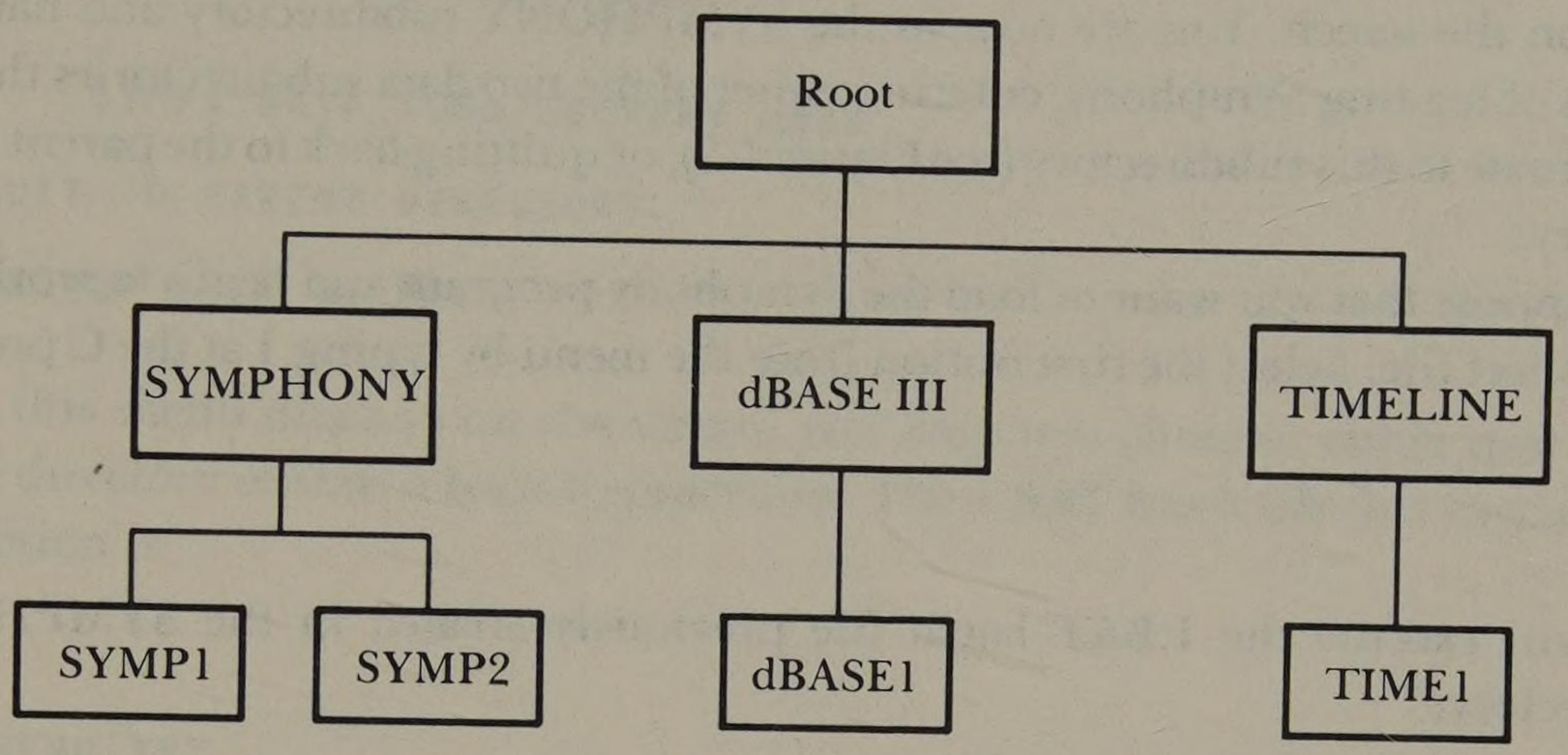


Figure 4-2. Subdirectory structure for the hierarchical menu structure in Tip 4.36

When the autoexec batch file has completed, the TYPE MENU.TXT command will display this menu on the screen.

Suppose you want to access Symphony. At the C prompt, type

```
C>1
```

This executes the root directory 1.BAT batch file, which we created to look like this:

```
CD\SYMPHONY
CLS
TYPE MENU.TXT
```

The CD \SYMPHONY command here makes the subdirectory containing Symphony the current one. The last line of the batch file, TYPE MENU.TXT, displays the Symphony Subdirectory Menu. While this line is identical to the last line of the previous batch file, it refers to a different menu; thus, a MENU.TXT file is unique to each subdirectory and contains the menu for that subdirectory. This is the MENU.TXT file we have placed in the SYMPHONY subdirectory:

```
                SYMPHONY SUBDIRECTORY MENU

1. LOAD SYMPHONY.

2. SYMP1 DATA SUBDIRECTORY.

3. COMMON SYMPHONY DATA SUBDIRECTORY.

4. QUIT TO PARENT DIRECTORY.
```

Once the 1.BAT batch file from the root directory finishes, you will see the above menu on the screen. You are now in the SYMPHONY subdirectory and have the options of loading Symphony, entering either of the two data subdirectories that are subordinate to this subdirectory (see Figure 4-2), or quitting back to the parent (root) directory.

Suppose that you want to load the Symphony program and begin to work on a spreadsheet file. Select the first option from the menu by typing 1 at the C prompt:

```
C>1
```

This will execute the 1.BAT batch file previously created in the SYMPHONY subdirectory:

```
PAUSE ***PLEASE INSERT SYMPHONY 1.1 DISK IN DRIVE A:***
SYMPHONY
DIR *.
REM RUN BACKUP FOR EACH DATA DIRECTORY WHERE YOU'VE MADE CHANGES
PAUSE ABOVE ARE DATA DIRECTORIES FOR THIS PROGRAM:
CLS
TYPE MENU.TXT
```

Note that this is totally different from the 1.BAT batch file we created in the root directory. To make this system work, you need to create unique 1.BAT and 2.BAT batch files and so on for the root directory and each subdirectory. The first line of this 1.BAT file pauses the execution of the batch file, and you are prompted to insert the Symphony diskette in drive A. If you don't, Symphony will fail to load, and you will be returned to DOS control (C> prompt). The next line, SYMPHONY, is responsible for loading the Symphony program. The last five lines execute after you exit from Symphony: The DIR *. will list all subdirectories in the current directory if they have no file extension (as is usual). The REM and PAUSE lines serve as reminders to back up any data files you created or modified during your Symphony session. Finally, the TYPE MENU.TXT at the end will redisplay the menu from which you chose Symphony.

Suppose that you now want to access a Symphony data file that you just created in the SYMP1 subdirectory, so that you can back it up. Selecting menu option two executes a 2.BAT batch file previously placed in the SYMPHONY subdirectory. This batch file changes the current directory to SYMP1 and then uses the familiar TYPE MENU.TXT command to display the menu for the SYMP1 subdirectory:

```
CD\SYMPHONY\SYMP1
CLS
TYPE MENU.TXT
```

The MENU.TXT file looks like this:

```
        SYMP1 DATA SUBDIRECTORY MENU
1.  QUIT TO PARENT DIRECTORY.
2.  BACKUP DATA FILES.
```

When this menu displays on the screen, you have two choices: either quit to the parent directory or start a backup procedure. The 1.BAT batch file that executes the first option is

```
CD ..
CLS
TYPE MENU.TXT
```

The CD.. command in the first line makes current the current directory's parent. In this case, it allows us to back up in the directory hierarchy from SYMP1 to SYMPHONY (see Figure 4-2). Once we've done that, the TYPE MENU.TXT again displays the menu for the SYMPHONY subdirectory. As long as we have a Quit batch file identical to this one in each subdirectory, we can always navigate backward in the hierarchy to any directory in the system.

The 2.BAT file in the SYMP1 data subdirectory reminds us about the backup options we have.

REM ISSUE BACKUP *filename* TO BACK UP A NEW OR MODIFIED FILE

This assumes that we have created a batch file called BACKUP.BAT to control the backup process in whatever way we wish. See the following Tip for one way of doing this.

Suppose that when we are done with our backup procedure we want to see the menu for the current subdirectory on the screen again. We can always obtain the menu for the current subdirectory by issuing the commands:

```
CLS
TYPE MENU.TXT
```

It is possible to place these two commands in a batch file in each subdirectory called MENU.BAT. Then entering MENU will always get you the current menu.

Although we have not described the batch files you would need for every menu choice, we have covered a representative sample to show all the important techniques. You should be able to use these techniques to construct a menu system that will serve your special needs.

You could refine this menu system by adding PATH commands to some of the menu choice batch files to make sure you had established program search paths for DOS where required. For instance, if you want to be sure that you can access certain root directory programs from the SYMPHONY subdirectory, you add a PATH \ command to the 1.BAT file in the root directory, immediately after the CD command.

4.37 Tip: *You can construct a batch file to automate the process of backing up your data files.*

Because so many things can happen to harm important data files or the disks on which they reside, it is important to establish and then follow a good procedure for regular backup. The safest approach is to back up each important data file as soon as it is created or modified. The previous Tip showed how you can use a batch file to remind yourself automatically to make backups each time you exit from a program that may have created or modified a data file.

The best backup technique makes sure you always have two generations of backups of important files at any one time. This means that before you create a new backup file for data file X, you make sure that you save the previous backup of X and mark it as the previous one. The reason for saving the previous generation before creating a new one from the original is that the latest original occasionally becomes corrupted in some way without your realizing it. If you then create a backup from the corrupted original without first saving the old backup, you have two copies of the corrupted file; you will probably have a difficult time reconstructing what the file looked like before corruption.

One batch file, called BACKUP.BAT, can help you make sure that you always keep two generations of backups:

```
REM ***THIS BATCH FILE MAINTAINS A 2 GENERATION SYSTEM***
PAUSE TO BACKUP, INSERT BACKUP DISK IN B: AND ISSUE BACKUP filename
COPY B:%11.* B:%12.*
COPY %1.* B:%11.*
REM ***BACKUP COMPLETE ***
REM ***ISSUE BACKUP filename FOR MORE BACKUPS.***
```

You execute this batch file by first making current the subdirectory containing the file you wish to back up. You then issue the command BACKUP X, in which X is the name of the data file. This filename must be less than eight characters because, as we'll see, the eighth character marks the backup generation. (For this version of the batch file, do not supply the file extension.) The PAUSE command in the second line prompts you to place your backup diskette in drive B. The COPY command that

follows substitutes the filename you provided for the %1 parameter marker in both the source and target specifications. The result is COPY B:X1.* B:X2.*. This command copies the current copy (if any) of backup generation 1 for X to backup generation 2, thus saving the previous backup generation as a safety measure. After parameter substitution, the next command becomes COPY X.* B:X1.*. This command copies the new original of X to the generation 1 backup file, completing the backup.

4.38 Trap: *Do not run FORMAT inadvertently on a hard disk, or else disastrous results will occur. You will lose all your files.*

After a hard disk has been used to store information, do not issue the FORMAT command for drive C (hard disk), or you could lose all the programs and data stored on it. The consequences of this potentially dangerous mistake are so serious that IBM advises users to remove the FORMAT program from the hard disk. This is not an effective solution because there are times when you need to format floppy diskettes. A batch file can be created to alleviate this danger by defaulting the FORMAT procedure to drive A unless the user specifies otherwise. The following is a small batch file that will do the trick:

```
C>RENAME FORMAT.COM FORMATT.COM

C>COPY CON:FORMAT.BAT
FORMATT A:
^Z
```

The name of the batch file is FORMAT.BAT; now, whenever anyone types FORMAT, a command that formats the disk in drive A will be executed. The important step here is renaming the regular FORMAT.COM command to FORMATT.COM.

4.39 Trap: *You cannot assume that all programs are written to recognize and utilize a hard disk.*

Some programs are written to address only disk drives A and B. Suppose that you have one of these programs that assumes a dual floppy system (drives A and B). However, your system is equipped with a hard disk (drive C). With a hard disk system, all of your data resides on the C drive, but your program expects to find the data on drive A or B. This will break down communication between you and your

program. The best solution is to use the ASSIGN command to reroute references temporarily from one disk to another. With this command you can fool a program that expects to work with two diskettes into working with one hard disk. To assign both drives A and B to your hard disk drive C, issue the command ASSIGN A=C B=C.

ASSIGN is only a temporary solution, requiring that you reassign the drives each time you use the program. A potential trap associated with the ASSIGN command is forgetting to set the drives back to their default settings when you are finished with a particular program. Taking this into account, you should use ASSIGN in a batch file that looks like this:

```
C>COPY CON:ACCT.BAT
REM  Switch to accounting program
ASSIGN A=C B=C
BASIC ACCT
ASSIGN
^Z
```

The fourth line first loads up the BASIC program and then loads the ACCT program written in BASIC. The fifth line resets the drive defaults when you are finished running your program.

4.40 Trap: *If your hard disk will not boot your system, it might be physically worn out.*

After several years of constant use, a hard disk can actually wear out. Tracks 0 and 1 are likely to wear out first, since they are accessed every time you start your system. If this happens, you may start to get a Boot Error message when you boot from the hard disk. If the damage is restricted to Tracks 0 and 1, your disk will still function. However, you will have to use a floppy diskette containing DOS to boot your system. To simplify the process, you should create an autoexec batch file on your floppy boot disk like

```
DATE
TIME
C:
```

This boots from the floppy, gets the date and time, and then makes C: current. You should also back up your entire hard disk immediately, just in case further damage occurs. Then, to be absolutely safe, have your hard disk serviced. You may need to replace it.

5

Batch Files

Batch files are a major capability of the DOS operating system. A simple batch file can automate a recurring function and make the routine use of the computer much easier and safer. The purpose of this chapter is to delve into the mysteries of batch files and mine a few jewels that will increase your PC productivity. Because the batch file capability is described so sketchily in the DOS manual, we will depart somewhat from the Tips and Traps format in this section to present you with an introduction on how to create batch files and use them effectively.

What Is a Batch File?

A batch file is simply a text file. This text file can be created by many word processors or any ASCII text editor that doesn't put special characters in the file (like the EDLIN editor on your DOS disk). You can also create a batch file from the DOS prompt by typing **COPY CON: filename.BAT** and **your text**, in which "filename" is whatever name you wish to assign, and "your text" is the one or more DOS and special batch file commands that you wish to place in the file. When you have entered the last line, press the F6 key to place an end-of-file marker (^Z) after the last line, and press ENTER. The COPY command will then

create the file and return you to the DOS prompt. To make sure you have created the file correctly, use the TYPE command to display its contents.

The main constraints on a batch file are that its name ends in the .BAT suffix and that it is an "ordinary" text file without any special characters, which may confuse the batch file interpreter. The text contained in a batch file is formed from a combination of standard DOS commands (such as COPY, TYPE, and so on) and a group of special batch file commands, which are defined in the Batch Commands section of your DOS manual. These commands look a little like the standard DOS commands you type after a DOS prompt (e.g., A>), but they are specifically designed to be executed as part of a batch file.

The DOS operating system treats batch files as if they were kitchen recipes. Just as a cook will follow instructions for baking a cake rigidly and in the proper sequence, so will the DOS operating system follow the command instructions in a batch file rigidly and in sequence. It is a simple matter to tell DOS to start following the batch file recipe—you simply type the name of the batch file without the .BAT suffix.

What Are They Used For?

Batch files are primarily used to automate a series of tasks that you want to perform time after time. Suppose that you have an accounting chore you perform once or even several times a day. You could create a batch file describing the series of tasks to accomplish this chore and call it POST.BAT. You could then type POST after the DOS prompt and the computer would chug away on your chore, leaving you free to work on something else.

One common use of batch files is ensuring protection from error. Many tasks you perform on your computer are subject to serious consequences if they are done wrong. These tasks may often be long, complex, and filled with "computerese" that is hard for uninitiated users to understand. If complex tasks are done manually, the possibility of error is higher. But if they can be automated and hidden from less skillful users who need to get the job done, you can feel more secure about the tasks being performed correctly.

Another use of batch files is the AUTOEXEC.BAT file. If DOS finds a file with this name on the disk you booted from, it is executed automatically. This capability allows you to set certain conditions automatically and run programs whenever you turn the machine on or reboot DOS. Since most DOS owners will have to look at AUTOEXEC.BAT files at one time or another, some knowledge of batch files and their operation will prove helpful even if you never create a batch file on your own.

Building Batch Files

Now that we have convinced you that batch files can be useful, how about learning to build one? You know that you must create a file with a .BAT file extension, and you also know how to do this, but what goes in the file? You can see all your choices in great detail in the Batch Commands section of your DOS manual, but we will go through the basics here. All you have to do is choose from the commands available and assemble your "recipe" for the task to be accomplished. Once you know what you want to do, you can start constructing a batch file to help you do it.

Creating a Batch File

Let's pick a simple example and elaborate on it step by step. As we go along, we will try to work in each of the batch commands. Suppose that your office uses Lotus 1-2-3 for graphs and spreadsheets and WordStar for letters. Some of your computer operators bounce from one of these programs to the other all day long. Because they perform these tasks so often, you would like to create a batch file that sets everything up. This way your operators can go from one program to the other without needing to know much about DOS. Creating a good batch file will help them.

If you start with a simple batch file to run Lotus and WordStar, you should be able to get moving. Begin by typing

```
COPY CON: GO.BAT
LOTUS
B:WS
```

You then press the F6 and ENTER keys in turn so that the COPY command will actually write the GO.BAT file to the disk. When you execute this file by typing **GO** at the DOS prompt, LOTUS will be executed from the current default drive and directory. When you exit from LOTUS, the batch file is still in control, and WordStar will be executed from the disk in drive B. (You must be sure, of course, that you have the proper disks in the proper drives.)

Now that you have established the basic file, you can go on to make it fancy. In the next subsection we will discuss the various batch file commands and how to use them.

The Building Blocks

DOS supports two batch file commands that allow you to document your files and make them more readable and usable. The first is the REM command. The

REM, or remark, command allows you to put messages into your batch files to tell the operator something. In your file you might add the following REM commands:

```
REM XYZ OFFICE PROCEDURE 11327 - IBM PC OPERATIONS
REM THIS SYSTEM WILL NOW RUN LOTUS AND WORDSTAR
REM BE SURE TO HAVE THE WORDSTAR DISK IN DRIVE B
LOTUS
B:WS
```

In this case the remarks would be displayed before running LOTUS.

You also have to access a command called ECHO that works a bit like REM in one of its variations but also had quite a different function. ECHO can be followed either by "ON" or "OFF" or by a message. If you put ECHO OFF in your batch file, DOS will not display all of the commands that are being processed by the operator. DOS will also suppress any REM messages you may have placed in the file. The ECHO command affects only those commands that follow it in the file, and the ECHO OFF command can be reversed by placing an ECHO ON command later in the file. If you use the ECHO command with a message after it, the message will be displayed to the operator even if you have said ECHO OFF. You could change your file to include the ECHO commands like this:

```
ECHO OFF
REM XYZ OFFICE PROCEDURE 11327 - IBM PC OPERATIONS
ECHO THIS SYSTEM WILL NOW RUN LOTUS AND WORDSTAR
ECHO BE SURE TO HAVE THE WORDSTAR DISK IN DRIVE B
REM NOW RUN LOTUS
LOTUS
REM THEN RUN WORDSTAR
B:WS
```

Here the ECHO OFF command would get displayed, the first REM command would not, the two ECHO messages would get displayed, and the LOTUS and B:WS commands would not.

In the example you are building, your operator may have difficulty reading your message about what the file does. Fortunately DOS provides a method for pausing the system long enough for the operator to do something like read a message or change a disk. The command is logically named PAUSE. The PAUSE command can be used with no following parameters and will cause the batch file to pause in the middle of execution, displaying the message

```
Strike a key when ready...
```

This would allow the operator time to read your message and insert the correct disk(s) before the batch file resumes execution. As modified with a PAUSE command, the file would look like this:

```
ECHO OFF
REM XYZ OFFICE PROCEDURE 11327 - IBM PC OPERATIONS
ECHO THIS SYSTEM WILL NOW RUN LOTUS AND WORDSTAR
ECHO BE SURE TO HAVE THE WORDSTAR DISK IN DRIVE B
REM STOP AND WAIT FOR THE USER
PAUSE
REM NOW RUN LOTUS
LOTUS
REM THEN RUN WORDSTAR
B:WS
```

Your first goal was to establish a procedure for your operator that would eliminate the typing of DOS commands as long as you used either Lotus 1-2-3 or WordStar. To do this, you need to tell the batch file to repeat part of itself. The GOTO command will allow you to do so. This command tells DOS to change the line of the batch file that should be executed next. Left to its own devices, DOS will simply execute the commands in order from top to bottom until it reaches the end of the file. The GOTO command, however, allows you to alter the order to anything you want. You need to be able to tell DOS where you want to go, so it provides you with a LABEL command. The form of these two constructs is shown in your modified batch file:

```
ECHO OFF
REM XYZ OFFICE PROCEDURE 11327 - IBM PC OPERATIONS
ECHO THIS SYSTEM WILL NOW RUN LOTUS AND WORDSTAR
ECHO BE SURE TO HAVE THE WORDSTAR DISK IN DRIVE B
REM WE START HERE
:START
REM WAIT FOR THE USER
PAUSE
REM NOW RUN LOTUS
LOTUS
REM WAIT FOR THE USER
PAUSE
REM THEN RUN WORDSTAR
B:WS
REM START OVER AGAIN
GOTO START
```

Note that the GOTO START at the end refers to the :START label just before the first PAUSE command. With these changes, the execution of your batch file will cycle forever between LOTUS and WS. There is a way out, though. You

can press CONTROL-BREAK or CONTROL-C to interrupt execution whenever you reach one of the PAUSE statements.

Your batch file will now do what you wanted, but you can protect it a bit by testing to see if the WordStar program is indeed in drive B. You do this by using another batch file command, the IF command. The IF command allows you to test for things in and around the batch file and execute a batch file command if the test is successful. In your case you want to test to see if the WordStar program is on the disk in drive B, and if it is not, to tell the operator to put it there and try the execution again. You can modify your file to allow using the IF, GOTO, PAUSE, and ECHO commands.

```
ECHO OFF
REM XYZ OFFICE PROCEDURE 11327 - IBM PC OPERATIONS
ECHO THIS SYSTEM WILL NOW RUN LOTUS AND WORDSTAR
ECHO BE SURE TO HAVE THE WORDSTAR DISK IN DRIVE B
REM WE START HERE
:START
REM WAIT FOR THE USER
PAUSE
REM NOW RUN LOTUS
LOTUS
REM SEE IF WORDSTAR IS ON B DRIVE
IF NOT EXIST B:WS.COM GOTO GETWS
REM GO HERE TO RUN WORDSTAR
:WS
REM WAIT FOR THE USER
PAUSE
REM THEN RUN WORDSTAR
B:WS
REM START OVER AGAIN
GOTO START
REM COULDN'T FIND WORDSTAR
:GETWS
REM TELL THE USER TO PUT WORDSTAR IN B
ECHO PLEASE INSERT WORDSTAR DISK IN DRIVE B!
REM TRY RUNNING WORDSTAR AGAIN
GOTO WS
```

There are many other forms of the IF command, including ones that check for two ASCII character strings being equal to each other. The IF command is often used with parameters to make batch files more flexible.

The last batch file command we will explain in this section is the FOR command. This command lets you perform a single operation on a group of items. For instance, if you needed to sort five files before running a report program, the FOR command would do the sorting for you in one operation. In the batch file you are building, you will use the FOR command to give the operator a directory listing of the A and B drives, before beginning the day's work. Thus, if the wrong disks are in the drives, changes can be made. With the FOR command added you have

```
ECHO OFF
REM XYZ OFFICE PROCEDURE 11327 - IBM PC OPERATIONS
ECHO THIS SYSTEM WILL NOW RUN LOTUS AND WORDSTAR
ECHO BE SURE TO HAVE THE WORDSTAR DISK IN DRIVE B
ECHO THE FILES YOU HAVE ONLINE ARE:
REM FOR ALL DISKS DISPLAY DIRECTORY
FOR %%D IN (A B) DO DIR %%D /W:
REM WE START HERE
:START
REM WAIT FOR THE USER
PAUSE
REM NOW RUN LOTUS
LOTUS
REM SEE IF WORDSTAR IS ON B DRIVE
IF NOT EXIST B:WS.COM GOTO GETWS
REM GO HERE TO RUN WORDSTAR
:WS
REM WAIT FOR THE USER
PAUSE
REM THEN RUN WORDSTAR
B:WS
REM START OVER AGAIN
GOTO START
REM COULDN'T FIND WORDSTAR
:GETWS
REM TELL THE USER TO PUT WORDSTAR IN B
ECHO PLEASE INSERT WORDSTAR DISK IN DRIVE B!
REM TRY RUNNING WORDSTAR AGAIN
GOTO WS
```

The way this FOR command works is that DOS sets the %%D variable in turn to each of the values within parentheses and then executes the command after the DO. The variable may be any single character preceded by %%. Thus, this particular command results in the execution of a DIR A:, followed by a DIR B:. DOS then proceeds with the execution of the next statement in the batch file, the PAUSE. You may more easily understand this batch file by referring to the flowchart shown in Figure 5-1.

We have now looked at all but one of the batch file commands, the SHIFT command. We have omitted it because it is obscure and rarely used. Its purpose is to help make batch files with many parameters work well.

Batch Files That Change

DOS allows you to have parts of your batch files that can change each time that you run them. The mechanism for this changeability is the "parameter" construct. A batch file parameter is a piece of information DOS saves from the prompt command line (in our case, A>GO). Each element on the command line

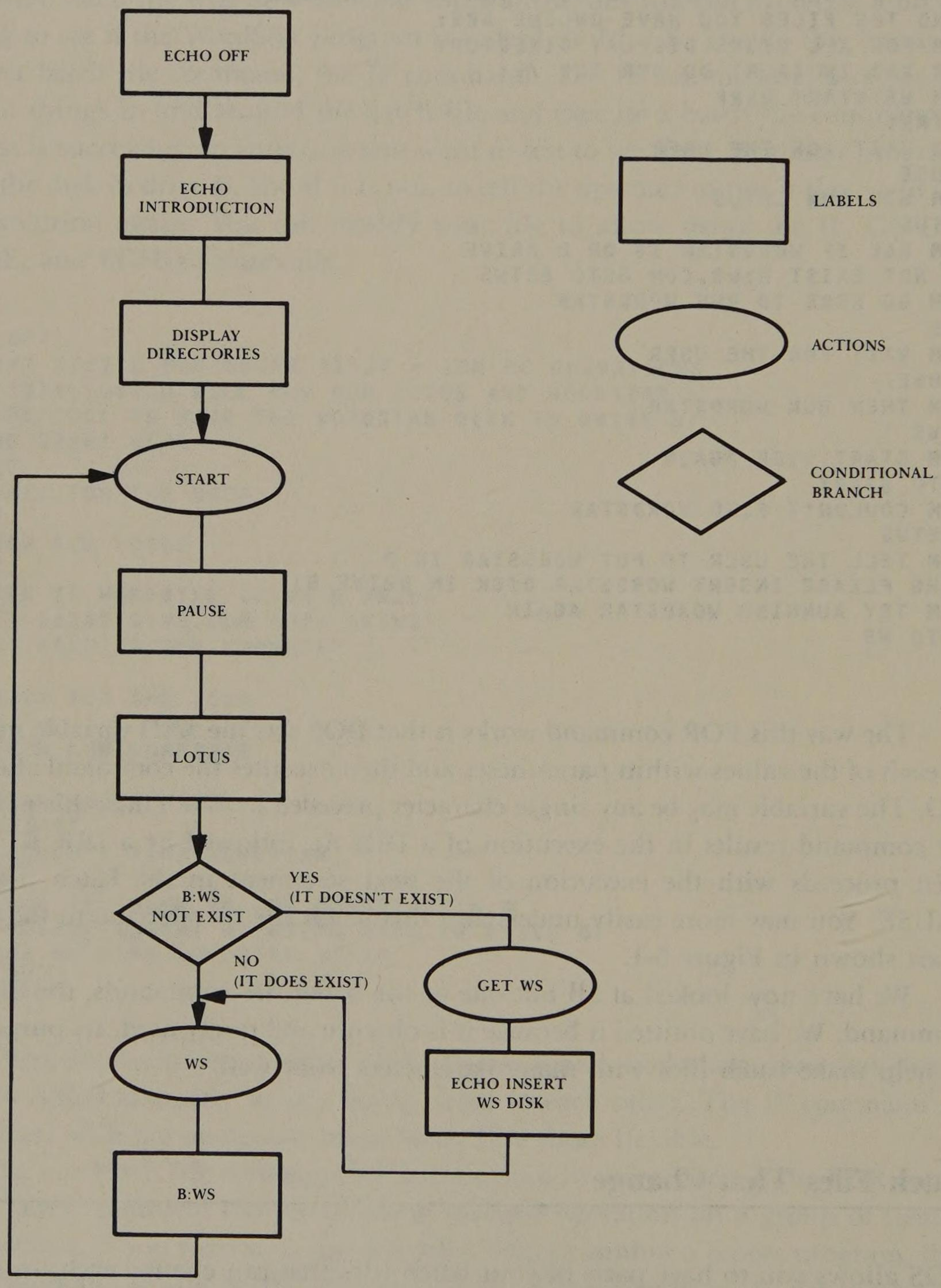


Figure 5-1. The GO.BAT flowchart

is separated out by DOS and kept aside. These pieces of the command you typed are available in the batch file by using parameters. The parameters are numbered from 0 to 9. The SHIFT command can extend this limit beyond 9 (see the SHIFT subcommand in the Batch Commands section of your DOS manual). In the batch file, a parameter is preceded by the % character (e.g., %0 %1 %2 ... %9).

You could add parameters to your file by allowing a different word processor to be used with LOTUS. Your example would then be

```
ECHO OFF
REM XYZ OFFICE PROCEDURE 11327 - IBM PC OPERATIONS
ECHO THIS SYSTEM WILL NOW RUN LOTUS AND %1
ECHO BE SURE TO HAVE THE %1 DISK IN DRIVE B
ECHO THE FILES YOU HAVE ONLINE ARE:
REM FOR ALL DISKS DISPLAY DIRECTORY
FOR %%D IN (A B) DO DIR %%D /W:
REM WE START HERE
:START
REM WAIT FOR THE USER
PAUSE
REM NOW RUN LOTUS
LOTUS
REM SEE IF THE NAMED COMFILE IS ON THE B DRIVE
IF NOT EXIST B:%1.COM GOTO GETWS
REM GO HERE TO RUN PROGRAM
:WS
REM WAIT FOR THE USER
PAUSE
REM RUN PROGRAM ON B DRIVE
B:%1
REM START OVER AGAIN
GOTO START
REM COULDN'T FIND PROGRAM
:GETWS
REM TELL THE USER TO PUT PROGRAM IN B
ECHO PLEASE INSERT %1 DISK IN DRIVE B!
REM TRY RUNNING PROGRAM AGAIN
GOTO WS
```

With a change to the IF statement, this batch file will now run any .COM file or .EXE file with LOTUS and will check to see if it is in the B drive. You would execute the file for WordStar by supplying GO WS at the DOS prompt. You could also execute it for VolksWriter by supplying GO VW at the DOS prompt. It should be clear that if you add other parameters, perhaps in various orders, you could extend this batch file. You could have the operator specify which drive the word-processing program was in. You could even change directories or do complex setups based on the operator's inputs. The availability of parameters makes the batch file capability of DOS very powerful and helpful in making your work easier and simpler. Several of the Tips we will discuss in this chapter use parameters, so you will have more examples to study.

COMMAND.COM—The Program That Does It ALL

Although we have been casually referring to DOS as executing the batch files, a particular program in fact does all the work. This is the program called COMMAND.COM, which is found on the DOS system disk. You may have wondered why you needed COMMAND.COM on your disk for this or that program to run. The COMMAND.COM Program is what actually reads your batch files and tells the DOS operating system what to do to accomplish the tasks you have established in the batch file. COMMAND.COM is a "command processor" and must be available to DOS if you wish to run a batch file or use many DOS commands. The program is brought into the computer's memory when you run a batch file and tends to stay there unless DOS needs the space for something else. When DOS has used some of COMMAND.COM's memory, you get this strange message:

```
Install the disk containing COMMAND.COM in drive x:
```

Understanding a little bit about COMMAND.COM and how it works makes some of the more interesting Tips in this chapter possible. The program COMMAND.COM can be executed from the DOS prompt or from within a batch file. A full description of the command is provided in the DOS manual, but here is a summary of the syntax of its execution:

COMMAND [D:][*path-definition*][/P][/C*command-name*]

COMMAND.COM allows you to tell DOS where it will reside via the [D:][path-definition] section of the command line. The /P option tells DOS to bring a new copy of COMMAND.COM into memory and make it permanent. After that, you won't get those strange messages telling you to install the COMMAND.COM disk. Finally, the /C part of the command instructs the program to go and execute "command-name" and return. Not all of these parameters to COMMAND.COM are required, and different ones serve different purposes. Many interesting uses are possible with the COMMAND.COM program.

COMMAND.COM also supports the DOS environment variables. These variables are named places inside COMMAND.COM where short bits of text information may be stored and retrieved by commands. The user command that sets up these environment variables is the SET command. Its syntax is SET [name=[parameter]]. We will use the SET command and COMMAND.COM in several of the following tips.

5.01 Tip: *Let PATH direct DOS to batch files in other directories.*

The DOS PATH command can allow you to use batch files in other directories. Let us suppose that you have issued the following SET command, perhaps in your AUTOEXEC.BAT file:

```
SET PATH=C:\MYSTUF;C:\DOSSTUF;A:\;B:\
```

After this statement is executed, DOS will look around for batch files that you may want to run. If you had, say, an accounting batch file in your MYSTUF directory, and you were currently in a different directory, you could type the name of the batch file, say POST, and DOS would try to find it. It would first look in your local directory, and if no POST.BAT file was found, it would then look in each of the directories you specified in the SET PATH command. The second place DOS would look is your MYSTUF directory; if it found a POST.BAT file there, it would execute it.

5.02 Tip: *Protect your user (and yourself) by controlling the use of a dangerous command.*

Some of the DOS commands can be very harmful if they are misused. Batch files provide you with a way to protect unsophisticated users from doing serious harm to the computer system. For example, the FORMAT command allows you to clear all the information from a disk and then begin again from scratch, which is useful for recycling floppy disks. It can be disastrous, however, if you accidentally use this command on a hard disk, which would reformat it and wipe out all of its current contents. One way to protect yourself against such a disaster is to substitute your own FORMAT command for the one provided by DOS. You do this by renaming the FORMAT.COM file with a new name such as #$.COM. With such a strange name it is very unlikely that anyone would execute it by accident. You then can create a front-end batch file for your private format command that only allows the user to do safe and sane things. An example of such a command would be

```
ECHO OFF
REM IF DRIVE C ISSUE WARNING
IF %1 ==C: GOTO WARN
REM IF LOWERCASE C ISSUE WARNING
IF %1 ==c: GOTO WARN
```

```
REM RUN FORMAT PROGRAM WITH PARAMETERS
#$ %1 %2 %3 %4 %5
REM LEAVE VIA END
GOTO END
REM GO HERE IF DRIVE C
:WARN
REM DISPLAY WARNING
ECHO WARNING!  YOU MUST NOT FORMAT THE HARD DISK
REM EXIT BATCH FILE
:END
```

In this batch file you check to see whether the user has inadvertently asked to format drive C, the hard disk. If that is what has been requested, you display a warning and do not execute the format. If you wanted to format the disk in drive B, for instance, you would supply FORMAT B: at the DOS prompt in the usual way. In this case, both IF statements in the batch file would fail, and the FORMAT command (now renamed as #$ in your example) would be executed on the disk in drive B.

This technique can be applied to any external DOS command that you wish to protect. You can't use it in exactly this way with any internal commands such as COPY and DIR. These commands are not separate programs, but part of the COMMAND.COM command processor itself. But you can create your own custom batch files to substitute for internal commands in a slightly different way: For your own custom front end to the COPY command, create a batch file with the name COPYIT.BAT and then use the COPYIT command whenever you want to make a copy.

5.03 Tip: *Use AUTOEXEC.BAT to set up your system before you start working.*

The file AUTOEXEC.BAT is executed whenever DOS is booted, if such a file exists on the boot disk. If you put commands into this file, they will be automatically executed at boot time. AUTOEXEC.BAT is a good place to put things like setting the date and time, moving to the proper directory, and running a standard program. Many of the Tips in this chapter can be applied to the file AUTOEXEC.BAT to allow you to insulate the computer system user from DOS and its commands.

5.04 Trap: *You cannot use path names in the IF EXIST command.*

One of the most powerful enhancements to DOS introduced in version 2.0 was the tree-structured directories. This support feature enables an operator to use large storage devices like hard disks effectively. Not every command in the system fully supports the structured directory's path specification, and the IF EXIST batch file command is one of them. It is not possible to issue a command like the following:

```
IF EXIST DOWN\MYFILE.WRK ECHO DO SOMETHING
```

Even if the file MYFILE.WRK exists in the DOWN directory, the command will fail. If you are sure that the DOWN directory is there, the following batch file can help work around this limitation:

```
REM CHANGE DIRECTORY TO DOWN
CD DOWN
REM CHECK FOR FILE
IF EXIST MYFILE.WRK ECHO DO SOMETHING
REM BACKUP TO PARENT DIRECTORY
CD ..
```

By creating a batch file like this one you can check for the existence of files in other directories. The CD .. command allows you to back up one level in the directory structure without having to know the path name of the target directory, thus making your batch files location independent and more usable in other directory structures. This batch file could be even more interesting if parameters were used for the directory and filenames.

5.05 Tip: *You can use CTRL-BREAK, or CTRL-C, to terminate a batch file.*

If your batch file is executing for a long time and you wish to end the processing early, or if your batch file is doing something you do not want it to do, like repeating an operation endlessly, you can type the key sequence CTRL-BREAK, or CTRL-C, and DOS will ask "Terminate Batch Job?" This will give you the opportunity to stop the processing by answering **Y**. If you respond with an **N**, the processing will continue. Although DOS accepts the keystrokes immediately, it may not act on them as quickly, since control must return to the command processor before the termination message can be displayed. In other words, DOS has to get to a breaking point before it can respond to the request.

However, you should keep in mind that when you type **Y** to the termination request, the process that was executing may not complete before executing the next line of the batch file. DOS will tend to abort the current running process to ask if you want to terminate, assuming that you would not have pressed the CTRL-BREAK or CTRL-C keys if you didn't want to stop execution of the batch file as soon as possible.

5.06 Trap: *If you remove a disk containing a batch file being processed, you will usually have to insert it again before the next command can be processed.*

DOS executes a batch file one line at a time and reads the batch file each time it needs to find out the next line of the file. If it tries to read the batch file and can't find it, it issues the message "Insert disk with batch file and press any key when ready." Since DOS reads the file each time it needs another command from it, if you insert a disk that has a file with the same name it will happily read the next record of that file. This could produce some potentially hazardous problems if the two batch files had the same name but not identical contents. Therefore, you should not give two batch files the same name, even when they reside on different disks, unless they actually are identical.

5.07 Tip: *You can change the current directory at will within a batch file.*

The change directory command (CD or CHDIR) allows you to change directories from within batch files. This capability allows you to manage your files more effectively by automating the process of moving around in the directory structure. If you organize your directories so that each type of file (e.g., word-processing files, spreadsheet files, accounting files, and so on) is placed in a directory containing only that type of file, you then can process different types easily by changing to the proper directory area within the batch files. You can even subdivide a given class of files like accounting files into separate subdirectories. For instance, you might have a subdirectory for each of your accounting applications, such as accounts receivable, accounts payable, inventory, general ledger, and so forth. A single batch file could then be constructed that changes directory to each of the proper subdirectories and performs the required functions and then moves on to the next. With the CHDIR command such flexibility is easy to implement.

5.08 Tip: *The last command in a batch file can be the name of another batch file, allowing you to chain from one to another.*

If DOS encounters the name of a batch file as a line in another batch file, then it will transfer processing to the named command. (This transfer acts as a branch, not a subroutine call, so control will never return to the batch file from which the branch took place. But see Tip 5.15.) Such a capability allows for the building of very long batch file groups and for creating pairs of batch files that cycle back and forth between one another. Several of the more advanced tips in this chapter use this feature to build batch files that support menus and isolate the user from the DOS system.

5.09 Tip: *You can set up a special load batch file for each piece of software you use frequently.*

Batch files allow you to establish an "alias" for software that you run regularly. In such batch files, you can do all the setup the software requires, invoke the applications program with its normal parameters, and clean up things after it finishes executing. This is one of the most common and powerful uses for batch files. It helps to isolate people from the complexities of the computer system and lets them concentrate on the jobs they wish to accomplish.

Suppose, for instance, that you have a dBASE III accounting application that creates a temporary log of the current session's activity against the accounting data bases. With each new session, you want to copy the latest log file to a backup file and then erase it so that it can be replaced with a new log. You might create an ACCT.BAT batch file that would look something like this:

```
REM SAVE A BACKUP COPY OF THE FILE
COPY ACCTLOG.DBF ACCTLOG.BAK
REM REMOVE THE WORKING FILE
ERASE ACCTLOG.DBF
REM CREATE A NEW FILE
DBASE ACCT
```

This would copy the current log to the backup file, erase it to prepare for its re-creation, and execute DBASE with a parameter instructing it to execute the DBASE ACCT command file automatically. Each time you want to start this ACCT application, all you have to do is enter ACCT at the DOS prompt.

5.10 Tip: *You can execute a DOS command more than once, with variations, by using the batch file FOR command.*

As we mentioned in the first section of this chapter, the batch file FOR command is very powerful. It allows you to perform DOS functions on a group of files rather than having to deal with each file separately. Here is an example:

```
FOR %%A IN (*.TXT) DO TYPE %%A
```

This command will TYPE all the .TXT files in the local directory one after another.

There are some restrictions on the FOR command, however. It is not possible to execute the command with another FOR command. DOS protests if you try this. You must use the double percent sign (%%) to identify your variable name (e.g., %%A), since DOS will interpret a single % as the beginning of a parameter.

Batch file parameters are allowed in a FOR command and in the DOS command that it executes. An example of this would be

```
FOR %%A IN (%1.TXT) DO TYPE %%A
```

Here, you can invoke the batch file with the filename parameter corresponding to the .TXT files you want to see. You could even pass wildcard characters in the filename parameter. For instance, supplying ???RPT would cause a TYPE on all files with names beginning with any three characters and ending with RPT.TXT.

5.11 Tip: *You can supply a parameter to a batch file that will determine the number of times a part of it will execute.*

In some situations it is helpful to be able to execute a batch file or part of it a number of times. For example, suppose you want to make from one to three different backup copies of a file under batch file control and want to be able to supply a different number each time you execute the batch file. There is no direct way of setting up a counter, incrementing it, and testing it in a batch file. However, there is a trick that will do the job indirectly. Suppose that you have a batch file called "BACK.BAT." We assume the existence of a dummy file called INIT, the contents of which are irrelevant. Here is what BACK.BAT looks like:

```
ECHO OFF
REM BACK WILL BACKUP THE FILE %1 TIMES
REM UP TO A LIMIT OF 3 TIMES
REM REMOVE ALL OLD COUNT FILES
ERASE *.CNT
REM CREATE THE COUNT FILE
COPY INIT %1.CNT
REM IS IT ONE?
IF EXIST 1.CNT GOTO ONE
REM IS IT TWO?
IF EXIST 2.CNT GOTO TWO
REM MUST BE THREE
COPY MYFILE.WRK \BACKUP\MYFILE.B3
REM TWO COPIES WANTED
:TWO
COPY MYFILE.WRK \BACKUP\MYFILE.B2
REM ONE COPY WANTED
:ONE
COPY MYFILE.WRK \BACKUP\MYFILE.B1
```

When you invoke BACK with the parameter "3" you will get three copies backed up to your backup directory. When you invoke BACK with "2" or "1," you will get either two or one backup copy, respectively.

It is not too hard to see that this approach can be generalized to have more operations depend upon the passed count. The trick is to use the temporary %1.CNT to contain the count as part of its name, and allow the IF EXIST command to let you look for all the possibilities.

5.12 Tip: *You can execute special batch commands from the DOS prompt.*

Although the most common use of batch file commands is to place them inside batch files for execution as part of the execution of the whole file, they are equally available at the DOS prompt. Perhaps the most useful is the FOR batch file command. It can be used to perform a single operation on a group of files in a single command to DOS. An example would be the display of all the files in the current directory that end in the suffix .TXT. The command allowing you to do this would be

```
A>FOR %A IN (*.TXT) DO TYPE %A
```

In this FOR command you wish to cover all the files in the group (*.TXT) and perform the TYPE command on each one. The %A indicates what you will call the elements in the group when the FOR command tries to execute your task.

It should be obvious that this technique is not limited to simple DOS commands; batch files can be executed this way as well. The FOR command can be a very powerful tool in working with groups of related files.

5.13 Tip: *You can combine several techniques to make the screen display cleaner and clearer while batch files are executing.*

The techniques we describe in this Tip are designed to reduce the "computer clutter" that typical applications programs dump on the screen, so that the display will appear as clean and to the point as possible. The examples we will use rely on the redirection capabilities of DOS. (See Chapter 6, "Redirection, Piping, and Filters.")

DOS supports a "dummy" device called NUL: that allows you to suppress unwanted output from programs. Any output to this device is simply "eaten" by the system and not displayed or placed in any file. It is DOS's "black hole." Where NUL: really starts to be helpful is in batch files, particularly the AUTOEXEC.BAT file that sets up your system. Let's look at a typical AUTOEXEC.BAT file that uses the NUL: device and redirection in interesting ways:

```
ECHO OFF
REM CLEAR THE SCREEN
CLS
REM COMFORTING MESSAGE
ECHO SETTING UP RAM DISK C:
REM DO THE WORK
SUPERDRIVE C:>NUL:
REM SET PROMPT TO SHOW CURRENT DIRECTORY
PROMPT $P$G
REM MORE COMFORT
ECHO NOW SETTING UP KEYBOARD
REM KEYBOARD MACROS SETUP
PROKEY DOS.PRO /R > NUL:
REM KEEP USER INFORMED
ECHO COPYING B: TO C:
REM COPY DISK
DISKCOPY B: C: < DCANS.TXT> NUL:
```

In this file you set up a RAM disk and route all the output to NUL:. (See Chapter 8, "Advanced Topics," for a discussion of RAM disks.) You do the same for the keyboard setup, which in this case uses the PROKEY keyboard macro program, and then provide DISKCOPY with a file of all your responses so that the user doesn't have to know what to type in. When all the dust clears, you have your system set up correctly, and the user saw only your two echo messages.

5.14 Tip: *You can create sorted directory listings with batch files.*

It is possible to create a batch file that allows you to have a variety of sorts of the directory listing provided by the DIR command. The principle here is that you can do both piping and sorting from within a batch file, which allows you to create a customized directory command. Let's create a general-purpose directory sort command called "SDIR.BAT":

```
DIR %1: ¦ SORT /+%2
```

In this batch file the first parameter to be passed is the drive letter. The second parameter is the first column the SORT command will do the sort on. For more information about the SORT command, see Chapter 6, "Redirection, Piping, and Filters."

You can then create a group of batch files that use SDIR to sort on various fields of the directory list. Some examples might be

By name:

```
SDIR %1 1
```

By extension:

```
SDIR %1 9
```

By size:

```
SDIR %1 13
```

In reverse order:

```
SDIR %1 13/R
```

For any of these to work, of course, you must have the SORT.COM program in your default directory. It is also possible to print these listings on the line printer by redirecting the output of SDIR to PRN:.

5.15 Tip: *You can call batch files from other batch files, return to the point of call, and continue execution.*

As we have mentioned, the COMMAND.COM program executes batch files for DOS. It can also execute batch files for us from within another batch file. In some

circumstances it can prove handy to have one batch file execute another and then return to the caller. For example, an environment setup that may be used in several situations could be defined in a batch file and then executed whenever needed as a called subroutine. The mechanism that makes this possible is the /C option of COMMAND.COM. COMMAND.COM will accept the /C option followed by any legal DOS command, including a batch file name with parameters. If, for instance, you had three batch files you wanted to execute in order and then do a few other tasks, you could create a batch file that contained the following:

```
ECHO OFF
REM EXECUTE FIRST WITH TWO PARMS
COMMAND /C FIRST 1 4
REM EXECUTE SECOND BATCH FILE
COMMAND /C SECOND
REM EXECUTE THIRD WITH B:
COMMAND /C THIRD B:
REM DO A DIRECTORY LIST OF B
DIR B:
REM WAIT FOR USER
PAUSE
```

This batch file would execute FIRST.BAT with the two parameters 1 and 4. SECOND.BAT would then be executed, followed by THIRD.BAT with B: passed as a parameter. After THIRD was finished, control would return to your main file and the DIR B: and PAUSE commands would execute.

You could embellish this approach with IF and GOTO commands to provide a powerful batch file environment indeed.

5.16 Tip: *You can create and modify batch files from any program that can create an ASCII text file.*

One of the common tasks required of a computer system is to allow for the selection of various options by a user of the system. This selection process often takes the form of a menu on the screen from which the user selects a choice. We will explore several ways to implement this capability in this chapter, one of which we will discuss here. This technique involves the use of BASIC to provide the prompting for our menus and batch files to carry out the work. We will begin with a simple example of how a BASIC program can modify a batch file and then extend the example to show how menus can be added. For these examples we will have three basic files. The first is a batch file we will call GO.BAT, which is the main starting point for our journey. GO.BAT will include a call to BASIC and one to another batch file called TASK.BAT.

Let's look first at the GO.BAT file:

```
BASICA BUILD
TASK
```

As you can see, there is not much to GO.BAT. BASIC executes the BASIC program BUILD.BAS and then transfers control to the TASK.BAT batch file. The reason for separating GO from TASK is to allow us to go back and forth between the two batch files. DOS easily allows a transfer from one batch file to another at the end of a batch file.

Now let's look at the BASIC program BUILD.BAS. This program is created by typing the text into BASIC and doing a SAVE. The BASIC text for the BUILD program is

```
10 REM "THIS PROGRAM CONSTRUCTS "
11 REM "THE BATCH FILE TASK.BAT"
20 REM "A MENU CAN BE CREATED TO SELECT "
21 REM "WHAT IS PUT IN TASK.BAT"
30 OPEN "O",#1,"TASK.BAT"
40 PRINT #1, "DIR"
50 PRINT #1, "PAUSE"
60 PRINT #1, "GO"
70 CLOSE #1
80 CLS
90 SYSTEM
```

The BUILD program creates the TASK.BAT file and writes some batch file commands into it. Thus, when the GO batch file gets to the line that executes TASK, TASK will contain whatever was just written into it. It should be clear that this technique allows the forming of batch files on the fly, and thus allows for dynamic changing of what will be done.

In our example, when we type GO, BASIC executes the BUILD program and creates the TASK batch file with the directory and pause commands in it. Control then passes to the TASK batch file, and it executes the DIR and PAUSE commands. After these commands have executed, control passes back to a new copy of our GO batch file and we start all over. In this way, we have created an isolated environment on our computer that will never allow DOS back in to give a prompt. We have effectively isolated our user from DOS and have provided the commands needed to do the tasks required.

We can also extend this example and put real menus on the screen and allow for choices as to which commands are to be executed. We still use the GO and TASK batch files, but our BASIC program gets a bit more complicated. Following is an example of this type of BASIC menu program.

```
10 REM "THIS PROGRAM CONSTRUCTS "
11 REM "THE BATCH FILE TASK.BAT"
20 REM "IT PRESENTS A MENU OF "
21 REM "CHOICES TO THE USER"
30 OPEN "O",#1,"TASK.BAT"
100 PRINT "SELECT ONE OF THE FOLLOWING:"
110 PRINT " 1.  DIRECTORY OF THE CURRENT DRIVE"
120 PRINT " 2.  DIRECTORY OF THE B DRIVE"
130 PRINT " 3.  DIRECTORY OF THE C DRIVE"
140 PRINT " 4.  GO BACK TO DOS"
150 PRINT " "
160 INPUT "WHICH DO YOU WANT";A
170 IF A = 1 GOTO 300
180 IF A = 2 GOTO 350
190 IF A = 3 GOTO 400
200 IF A = 4 GOTO 450
210 GOTO 100
300 PRINT #1, "DIR"
310 GOTO 500
350 PRINT #1, "DIR B:"
360 GOTO 500
400 PRINT #1, "DIR C:"
410 GOTO 500
450 PRINT #1, "EXIT"
460 GOTO 500
500 PRINT #1, "PAUSE"
510 PRINT #1, "GO"
520 CLOSE #1
530 SYSTEM
```

In this more advanced BUILD program we display a menu of choices on the screen and write different entries into the TASK batch file, depending on what the user selects. Our example gives a variety of directory commands, but clearly shows that a command or group of commands could be written to the file in response to the user. Of course, the actual contents of the menu and what commands are written depend on what tasks are executed regularly on your computer system.

This approach can yield a very friendly front end to DOS that makes things easier for the inexperienced user.

5.17 Tip: *You can create interactive batch files that ask the user for a response and then proceed accordingly.*

As you develop more sophisticated batch file applications, one deficiency in the batch file language becomes very obvious. There is no direct provision for asking the user whether a specific operation should be performed and for altering the course of the batch file operation based on the answer. In this Tip we present one technique as a possible solution to this problem. This technique requires the cooperation of the BASICA (or BASIC) interpreter. The possible applications for

this type of batch file are many. The only reason we have used BASIC is that it is guaranteed to be available to everyone. You can use any other programming language if you prefer.

The principal strategy here is to write a program that will read the user's response and create one of two files, a *yes* file or a *no* file. Once these files are created, the batch file IF EXIST command can make decisions about what to do next in the batch file flow.

Let's begin by looking at the BASIC program. We call this program YN. It is short enough to be typed into BASIC and saved to the disk with the BASIC command SAVE YN. Our yes/no program looks like this:

```
10 ON ERROR GO TO 500
20 KILL "ANSWER.YES"
30 KILL "ANSWER.NO"
40 A$ = INPUT$(1)
50 IF A$="Y" OR A$="y" THEN GOTO 200
60 IF A$="N" OR A$="n" THEN GOTO 300
70 GOTO 40
200 OPEN "O",#1,"ANSWER.YES"
210 WRITE #1,A$
220 CLOSE
230 SYSTEM
300 OPEN "O",#1,"ANSWER.NO"
310 WRITE #1,A$
320 CLOSE
330 SYSTEM
500 RESUME NEXT
```

The YN program erases the files ANSWER.YES and ANSWER.NO before obtaining the user's response. It checks for either an N or a Y and creates the appropriate file. Once the file is created, it is written to and then closed. The appropriate file then exists and can be checked for in the batch file.

The batch file that we present as an example asks the user whether specific functions are desired and then performs them. You should note that our BASIC program does not display any prompt messages; this is done by ECHO statements in the batch file. We must also use the PAUSE command after the prompts, since BASIC insists on clearing the screen when it begins to run. Here is what the GO.BAT file looks like:

```
ECHO OFF
REM THIS BATCH FILE WILL ASK THE USER
REM WHAT HE WANTS TO DO AND THEN DO IT
:START
ECHO DO YOU WANT A DIRECTORY LIST?
ECHO TYPE SPACE THEN Y OR N:
REM WAIT FOR THE USER
PAUSE
REM GET THE Y OR N AND MAKE THE FILE
```

```
BASICA YN
REM CHECK FOR YES
IF EXIST ANSWER.YES GO TO DIR:
REM ASK NEXT QUESTION
ECHO DO YOU WANT TO KNOW YOUR DOS VERSION?
ECHO TYPE SPACE THEN Y OR N:
REM WAIT
PAUSE
REM GET ANSWER
BASICA YN
REM CHECK FOR NO
IF EXIST ANSWER.NO GOTO END
REM DO VER
VER
REM RESTART
GOTO START
REM WANT DIRECTORY LISTING
:DIR
REM DO DIR
DIR
REM RESTART
GOTO START
REM EXIT
:END
```

This technique can provide a very flexible yet carefully controlled environment for nontechnical computer users. It can be improved by using a programming language other than BASIC.

5.18 Tip: *You can make sure DOS can always find COMMAND.COM.*

As we have mentioned, the COMMAND.COM program actually performs the operations that you specify in batch files. DOS must be able to find a copy of COMMAND.COM somewhere when a batch file terminates in order to reestablish the connection to the normal command line operation of the system. Should any operation force DOS to reuse part of the memory in which it was storing COMMAND.COM, it must load a new copy of COMMAND.COM into memory. When DOS can't find the copy of the program in memory, it issues a message telling you to insert the disk that has the COMMAND.COM program on it. This message can be most frustrating to users of hard disks and RAM disks, because DOS will not find COMMAND.COM on those devices unless explicitly told where to look. It is possible to tell DOS where to look for your COMMAND.COM and also to indicate that you would like to have COMMAND.COM permanently loaded into memory so that no time is lost finding and loading it. Let's look at an AUTOEXEC.BAT file and its partner, FINISH.BAT, which order the world the way we want it. The AUTOEXEC.BAT file is

```
ECHO OFF
REM SET THE DATE AND TIME
ASTCLOCK
REM COPY TO RAM DISK
COPY A:*.* C:
REM CHANGE TO RAM DISK
C:
REM SET UP NEXT AUTOEXEC FILE
COPY C:FINISH.BAT C:AUTOEXEC.BAT
REM BRING IN NEW COMMAND PROCESSOR
COMMAND C:\ /P
```

In this batch file we are initializing our clock (in this case the clock is on an AST board), copying all our files to drive C, and changing the default drive to C. The COPY command copies our partner FINISH.BAT to the AUTOEXEC.BAT name on drive C. We then execute COMMAND.COM with two parameters. The first says that it should assume its home is on drive C, and the second tells it to stay in memory when a batch file completes. All this is fairly straightforward except for the COPY command line. This is required because when COMMAND executes, it will process any AUTOEXEC.BAT file on its home disk and directory. If it tried processing our original AUTOEXEC.BAT, it would simply repeat all the commands in this file, including starting up a new copy of itself. This would result in what is called in computer programming an *infinite loop*, a process that would go on forever or until available memory was exhausted. To avoid this problem, we split our AUTOEXEC.BAT functions into two parts. The first gets us started and the second finishes the process. In our case, the FINISH.BAT file looks like this:

```
ECHO OFF
REM LOOK FOR PROGRAMS IN ORDER
SET PATH=C:\;A:\;B:\
REM WHERE TO FIND COMMAND.COM
SET COMSPEC=C:COMMAND.COM
REM WHAT PROGRAM PROCESSES COMMANDS
SET SHELL=C:COMMAND.COM
REM PROMPT WITH CURRENT DIRECTORY AND >
SET PROMPT=$P$G
REM RUN WORDSTAR
WS
```

In our FINISH.BAT we set up the environment the way we want it with SET commands, since the new copy of COMMAND.COM will not remember the commands set up before starting. Finally, we execute our WordStar applications program, using the WS command line. When the applications program terminates, control will return to our new COMMAND.COM, and no message about inserting disks will appear. All this will remain in effect until we type EXIT,

which returns to the previous copy of COMMAND.COM. Thus, we will lose any environment variables that were established.

5.19 Tip: *You can use environment variables in batch files.*

We recommend this advanced Tip for the experienced DOS user. An undocumented DOS feature can be used to make batch files much more flexible and powerful. This feature is the availability of DOS environment variables within batch files. Environment variables are small named spaces in the operating system that can store some text. Whenever the names of these variables are referred to, the text that was assigned to them is substituted for the name. These same environment variables are available in batch files if their names are enclosed in % signs. The environment variables used must be SET before they can be referred to from the DOS command line, an AUTOEXEC.BAT file, or any other batch file. The recommended approach is to set them up in the AUTOEXEC.BAT file so that space is reserved for them. Loading resident portions of DOS (e.g., graphics, print, and so on) takes up space allotted to environment variables. The example that we will use comes to us courtesy of Bob Becksted's public domain contributions. We will demonstrate how to call one batch file from another and return to the caller without using the COMMAND.COM program.

For this we will need two batch files. The first is named CALL.BAT and the second CALLED.BAT. In the first file, we have statements that test and set up some environment variables and in the second, we use one of these variables to return to the caller. We need to look at some tricks here, but let's start by looking at the files. The first is CALL.BAT:

```
ECHO OFF
. test to see if environment variable exists
IF %0%BATLABEL%==%0BATLABEL%% GOTO DEFAULT
IF NOT %BATLABEL% ==.  GOTO %BATLABEL%
. note BATLABEL is assumed to be set to '.'
:DEFAULT
ECHO CALL.BAT EXECUTED
SET BATCALR=%0
. give the called batch file my name to get back
. then leave word where to come back to
SET BATLABEL=RETURN
. set further parms to save %1 thru %9 if
necessary
%1CALLED
:RETURN
ECHO FINISHING UP CALL.BAT
SET BATLABEL=.
```

In the CALL.BAT file you will note that we first test to see if the BATLABEL environment variable exists. We do this by comparing the program's name (%0) and the environment variable with the name and the characters "BATLABEL%%." The double percent signs are a batch file convention that allows us to use the actual % character rather than having the system do a substitution. We also check to see if the environment variable is initialized to a ".". If it is, then we know that the batch file is being entered for the first time. This file takes advantage of being able to operate differently when it is invoked, depending upon what state the BATLABEL environment variable is in. This all may seem tricky, and it is, but the results are very interesting and useful.

After the label :DEFAULT there is space for the file to do some useful work. In our example we let you know that we ran with an ECHO statement, but it could well have been a menu of choices of tasks for the user. The next interesting line is the SET BATCALR=%0 statement, which sets into an environment variable BATCALR the name of the calling batch file, in this case CALL. We also set the "communication variable" BATLABEL to hold the label name that will be "gone to" when the CALL file is invoked the second time. We will see how that all works later.

The next line is %1CALLED, which actually invokes the second batch file. The %1 variable implies that when CALL is invoked from the keyboard, we will specify the disk the CALLED file is on if it is not on our current default disk.

The last three lines in the file are executed when CALL is reentered after the CALLED file is executed. We do some more useful work and reset our key communication variable to an initial state. This covers the CALL file. Our next batch file is the CALLED.BAT file, which gets invoked from CALL. It looks like this:

```
ECHO OFF
ECHO CALLED BATCH FILE EXECUTES
. return to the caller
%BATCALR%
```

The CALLED batch file is much simpler than the CALL file. Here we only have to do the work we want and then invoke %BATCALR%, which in our case decodes into the characters "CALL." We could have several different CALLED files, each doing a different task, such as those for a menu. See Table 5-1 for some examples.

When the CALLED batch file executes the statement CALL, a fresh copy of the CALL batch file is invoked. The new CALL batch file first checks to see if any RETURNs are pending by testing the BATLABEL environment variable. If it is set to RETURN, the batch file will logically take the RETURN branch and do the correct final activity and cleanup. Thus, even though the original CALL

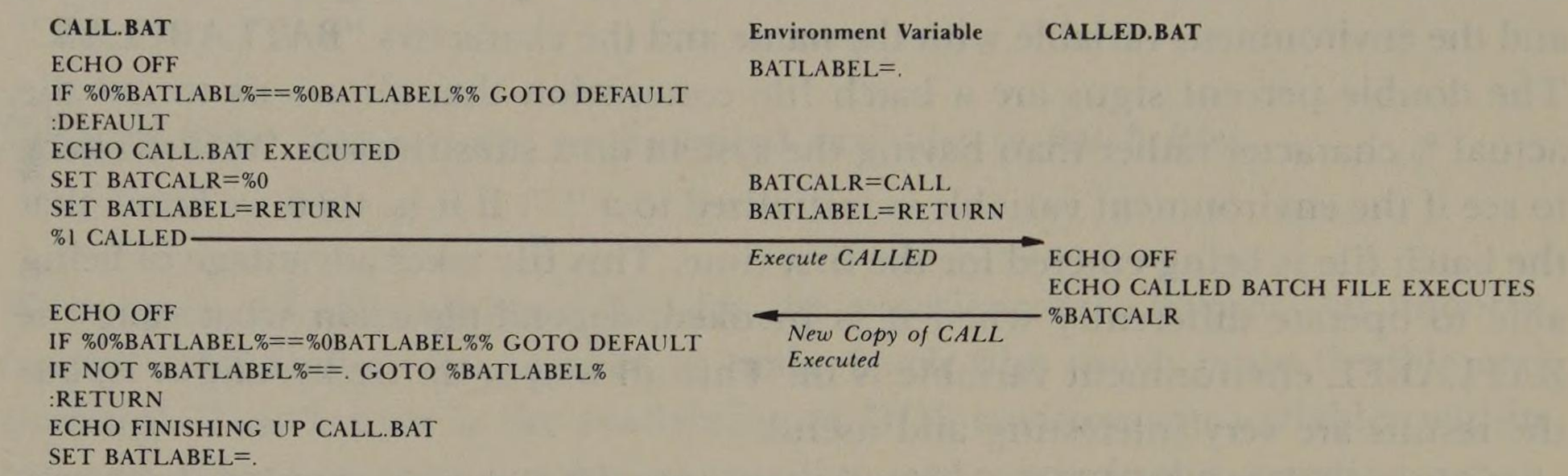


Table 5-1. Examples of Environment Variables and Batch Files

batch file died when CALLED was invoked, the new CALL file can take its place and do what we want. This basic capability of environment variables to hold information across executions of batch files is very powerful and potentially useful.

If you are interested in using this technique in your batch files, there are a few things you will want to know. The first is that environment variables are very sensitive to blank characters. The command SET BATLABEL=RETURN and the command SET BATLABEL = RETURN are *not* equivalent. In the first case, BATLABEL gets set to "RETURN" and in the second, it gets set to " RETURN." This feature can cause many hours of needless debugging. You should also watch out for a new set of environment variables being created when a new copy of COMMAND.COM program is executed. This means that you cannot pass information to batch files that are invoked by a call to COMMAND with the /C parameter.

You also should know that the environment variables are expanded only once at the beginning of the batch file execution. That means that SET commands within the batch file can reset a variable during the execution of that file, but any tests on that variable will use the original value of the variable. Our example works because a fresh copy of CALL is invoked and the variables are expanded again.

6

Redirection, Piping, And Filters

This chapter covers a group of features that are standard with the UNIX operating system used on many minicomputer systems. We describe how you can use them individually and in various combinations to simplify your day-to-day routine. These features enable you to take full advantage of other PC-DOS commands by making them easier to use or by enhancing them.

Several of the Tips in this chapter employ redirection, pipes, and filters inside of batch files to enhance DOS facilities or create new ones. The best way to understand how each section of a batch file works is to type it in a line at a time directly after the A> prompt. This way you can watch the program work in slow motion and stop to examine any files that are changing, with the TYPE command. The only trick here is that when you come to the parameters (%1, %2, and so on), you have to type in the same test string everywhere the parameter appears in the batch file. See Chapter 5 for details on batch file programming techniques.

Redirection

With I/O redirection you tell DOS to change the standard source or destination for input and output. Standard input is the keyboard. When you redirect input, you tell DOS to get data from a disk file or some other device. Standard output is the display screen. When you redirect output, you tell DOS to display data into a disk file, a printer, or some other device. DOS handles the mapping of the logical devices (standard input and standard output) to the physical devices in a way transparent to the programs whose input or output is being redirected. That is, the programs don't really have to know which physical device is being used.

Standard error is another logical device that maps to the display screen. Programs and DOS commands use standard error to report error conditions. Standard error cannot always be redirected. The reason is to make certain you see error conditions reported. For example, if you enter the command

```
DIR A:
```

you see the current directory displayed on your screen, then the prompt. If you enter the command

```
DIR A: > NUL
```

you do not see the directory displayed. DOS redirected the directory display away from the screen. DOS returns only the prompt to the screen.

If you enter the command

```
DIR A: > NUL
```

and you leave the A: floppy disk drive door open, DOS ignores the redirection and returns this error message to the screen since the screen is the standard error device.

```
Not ready error reading drive A
Abort, Retry, Ignore?
```

DOS is not consistent in which error messages it will or will not redirect. You will learn more about the ">" and the "NUL" in the following tips.

6.01 Tip: *You can think of the CTTY (Change Console) command as a type of redirection.*

The main purpose of this command is to redirect both input and output to one of your communication ports: COM1, (AUX), or COM2. This enables you to control your computer from a remote location with another computer or terminal.

CTTY will only work with a device that inputs and outputs. Do not specify a printer—DOS will attempt to read from it, which is not possible. See Chapter 8, "Advanced Topics," for more on CTTY.

6.02 Tip: *You can create a directory file by redirecting the output from the DIR command to a file.*

The directory displayed on your screen by the DIR command is standard output. You can redirect the display to any valid filename. You can then TYPE the file. This will enable you to view the file as needed. Also, you can PRINT the file for a hard copy of your directory.

```
A>DIR > DIRFILE.EXT
A>TYPE DIRFILE.EXT
A>PRINT DIRFILE.EXT
```

Another alternative is to redirect the listing to the printer to give you an immediate hard-copy directory listing.

```
A>DIR > LPT1     [ PRN or LPT2 ]
```

6.03 Tip: *You can automate daily routine procedure responses.*

You can create a file that contains your normal keyboard responses to a series of prompts from a program. There are two kinds of keyboard responses to a program:

1. You key one or more characters, and the program waits for you to press the ENTER key to accept the data. When you create your response file to hold these characters, you then press the ENTER key after you key the prepared response.

2. You press one key, and the program acts upon it immediately. For example, DOS prompts "strike any key when ready." This is "Inkey" type data. For the

"Inkey" type, not followed by the ENTER key, you enter the next response immediately after the "Inkey" response, on the same line.

You may want to do this with an applications program that requires a standard procedure for startup or logoff each day. We will use the COMP command to illustrate both kinds of responses. The COMP command compares two files. The first step is to create the response file. COMP requires three separate responses:

1. Filename1 followed by ENTER.

2. Filename2 followed by ENTER.

3. Y or N to compare another file. This is an "Inkey" type response character.

Let's suppose that we want to compare the AUTOEXEC.BAT files on two floppy disks. Create the response file COMPIN from the keyboard:

```
COPY CON: COMPIN
A:AUTOEXEC.BAT [Enter]
B:AUTOEXEC.BAT [Enter]
N [F6]                          (N followed by function key F6)
```

Next, execute the COMP command with the keyboard responses redirected from the response file.

```
COMP  < COMPIN
```

Two responses you will encounter over and over again are "N" (or "no more"), and the ENTER key (or carriage return key). You can create a NO response file for redirected input of "N":

```
COPY CON: NO
N [Enter]
[F6]                    (Function key F6)
```

You can create a CR response file for redirected input of the ENTER key:

```
COPY CON: CR
[Enter]                 (Press the ENTER key)
[F6]                    (Function key F6)
```

6.04 Tip: *You can automate testing procedures for debugging programs.*

Suppose that you are testing a program that requires a set of user responses. You could create an input response file for each test you needed to run and use redirection to direct the input from that file into your program. You could also redirect any output to the screen to a file so that you could examine it later, instead of losing it as it scrolls across your screen. The DOS command line to perform both kinds of redirection for a program called TEST might look like this:

```
A>TEST <INPUT >OUTPUT
```

This will cause the program TEST to read its standard input from a file called INPUT and write its standard output to a file called OUTPUT.

6.05 Trap: *DOS can lock up if you automate program input responses but do not provide all the responses in the response file.*

When you redirect input to a program, you must provide all of the responses the program expects. If the program looks for more input after the end of your input file, DOS will stop processing. You may either have to hit CTRL-BREAK or "warm-boot" with CTRL-ALT-DEL.

For example, the COMP command, which compares two files, requires three separate responses:

1. Filename1 followed by ENTER.
2. Filename2 followed by ENTER.
3. Y or N to compare another file.

To see what will happen if you don't include all the responses in a response file, create the response file COMPIN from the keyboard, but "forget" to include the third response of "N":

```
COPY CON: COMPIN
AUTOEXEC.BAT [Enter]
AUTOEXEC.BAT [Enter]
[F6]                          (Function key F6 ends the file)
```

Next execute the COMP command with the incomplete keyboard responses redirected from the response file.

```
COMP  < COMPIN
```

DOS will suspend processing and hang up with just the cursor blinking. You cannot regain control of the cursor or the command. You can break out of the command by pressing CTRL-BREAK.

6.06 Trap: *You can unexpectedly erase an existing file with the create-only ">" output redirection.*

There are two ways to make this mistake:

1. You redirect output to a file that already exists, thus replacing the old contents of that file with the redirected output.

2. You use ">" to pipe output to another program instead of "¦", the correct pipe character. You then overwrite the second program instead of passing data to it. Be particularly careful not to overwrite FIND.EXE, MORE.COM, and SORT.EXE programs in this way.

For example, you intend to enter the statement

```
DIR ¦ MORE.COM
```

to get a screen-by-screen directory. By mistake you enter

```
DIR > MORE.COM
```

which will not give you a nice screen-by-screen directory. It will instead overwrite the program MORE.COM with the image of your directory. You will then need to restore MORE.COM from your DOS backup diskette. Normally you would enter only MORE, not MORE.COM. In that case you would simply create a new file called MORE and not overwrite MORE.COM. This example does apply to any program to which you might pipe a file.

6.07 Tip: *By using the add-or-create ">>" output redirection, you can append files and avoid erasing a cumulative file.*

Where you have a daily or weekly cumulative file for transaction data, you can use ">>" to guarantee you will not erase the to-date data by mistake.

For example, you create an end-of-day logoff batch file to save your daily transactions to a week-to-date cumulative file:

```
COPY CON LOGOFF
TYPE DAILY.TRN >> WEEKLY.TRN
[F6]                                    (Function key F6)
```

Suppose that each weekend, the WEEKLY.TRN file is deleted. On Monday you run Logoff to create the WEEKLY.TRN file. On Tuesday through Friday you run Logoff to append the day's transactions to the end of the WEEKLY.TRN file.

6.08 Tip: *You can redirect program messages for later viewing.*

Some program screen output is too important to miss. You might not see it because it scrolls too quickly or because you are away from the computer when the output is displayed. You can be sure to see everything if you redirect output to a disk file for viewing later. TREE is an example of a fast-scrolling report you would want to review:

```
TREE > TREE.SAV
```

will save the TREE report for you in the file TREE.SAV.

6.09 Trap: *BASIC V1.1 will not allow you to redirect input or output.*

This is a feature that first became available in V2.00.

6.10 Trap: *Some programs will not allow you to redirect input or output.*

Programs that use the standard input/output feature (i.e., DOS function calls to perform standard input or output) to be able to redirect input or output are called "well behaved." Programs that write directly to the screen and bypass standard output are called "not well behaved." You may need to experiment to determine whether you can redirect input or output for a particular program.

6.11 Trap: *You cannot redirect input or output at the batch file level.*

You can redirect input and output for DOS commands and programs within a batch file. The same redirection will not work if you use it on the line where you execute the batch file. A batch file does not treat the redirection operator ">" and filename as parameters; it simply ignores them. The first example shows a batch file with the output redirected from the line in the batch file:

```
COPY CON: REDIRECT.BAT
COPY AUTOEXEC.BAT TEMP > NUL
^Z
```

Execute the batch file. Note that "1 File(s) copied" does not appear on the screen.

```
REDIRECT
COPY AUTOEXEC.BAT TEMP > NUL
```

Now create a batch file with no internal redirection.

```
COPY CON: REDIRECT.BAT
COPY AUTOEXEC.BAT TEMP
^Z
```

Then execute this batch file using redirection of the output to a file called TESTOUT.

```
REDIRECT>TESTOUT
```

Note that the "1 File(s) copied" now appears on the screen. DOS has ignored the redirection expression after the batch filename. The batch file is one type of program that redirection will not work with.

6.12 Trap: *BASIC program redirection may seem inconsistent to you.*

If you redirect both input and output, the output will be redirected as expected, and no print output will print on the screen. If you do not redirect input and you redirect output only, print output will be displayed on the screen in addition to being sent to the redirected destination.

6.13 Tip: *You can suppress unwanted output messages by redirecting them to the special NUL device.*

NUL is a special reserved device name for unwanted data. It can be helpful in suppressing DOS command output that is not suppressed by the batch command ECHO OFF. When you have a standard batch file (e.g., for daily logon or logoff), you may want to do file copies or other DOS commands that display messages. Unless you redirect these messages to NUL, you and your users will be forced to look at them each time you invoke the batch file.

```
REM ************************************
REM ****          NULEXAMP.BAT        ****
REM **** Redirect-to-NUL example.   ****
REM ************************************
REM
ECHO ON: Display DOS command output.
REM
     COPY AUTOEXEC.BAT
REM ************************************
REM * With ECHO ON, both the COPY      *
REM * command and the message display:*
REM*
REM* COPY AUTOEXEC.BAT                 *
REM* File cannot be copied onto itself*
REM* 0 file(s) copied                  *
REM ************************************
REM
ECHO OFF: Do not display DOS commands.
ECHO *
ECHO *  "COPY AUTOEXEC.BAT"  is next.
ECHO *
       COPY AUTOEXEC.BAT
ECHO ON:  Resume display of commands.
REM *
REM ************************************
REM * Though you suppress display of   *
REM * the COPY command with ECHO OFF,  *
REM * the COPY message still displays  *
REM *                                  *
REM *File cannot be copied onto itself*
REM *   0 File(s) copied               *
REM ************************************
REM *
REM *  "COPY AUTOEXEC.BAT > NUL"  next.
REM *
ECHO OFF: Do not display DOS commands.
       COPY AUTOEXEC.BAT > NUL
REM *
REM ************************************
REM * By redirecting to NUL, you       *
REM * suppress the COPY message also.  *
REM ************************************
```

6.14 Trap: *Redirecting to NUL can make you think DOS is hung up.*

If you redirect a DOS command output to NUL, the DOS prompts are not visible on the screen. The cursor blinking on a blank line appears to be a DOS hang-up. In reality, the DOS command is waiting for you to enter a response.

For example, if the default drive is C, and you redirect the FORMAT output to NUL:

```
FORMAT > NUL
```

What you see next is a blank line with the cursor blinking.

Actually, the FORMAT command has issued the prompt

```
WARNING, ALL DATA ON NON-REMOVABLE DISK
DRIVE C: WILL BE LOST!
Proceed with Format (Y/N)?
```

But you do not see the prompt because it has gone to NUL. If you should just happen to press "y" or "Y," you would erase all the files from your hard disk. Exercise great care when redirecting screen output to NUL!

6.15 Tip: *You can save the current path and restore it later automatically.*

When you are using different programs, you may need to set up different paths for a program to run and then want to return to the original paths when the program is completed. You can do this by redirecting the current path to a holding file, then retrieving it later. To save the current path, execute the following statement:

```
PATH > SAVEPATH.BAT
```

If the current path is A:\DIR1;A:\DIR2, this creates a new file SAVEPATH.BAT with the following:

```
PATH=A:\DIR1;A:\DIR2
```

Conveniently, this is also the command syntax for the PATH command. You could then change the path for the new program, and restore the original path by executing SAVEPATH.BAT. For example, from any directory you could execute

the following batch file to save the current path, make a new path, run a program, and restore the original path:

```
COPY CON EXECPATH.BAT       (Create the batch file.)
PATH > SAVEPATH.BAT         (Create the current path file.)
PATH=C:\DIR5;C:\DIR6        (Set a new path for your program.)
PROGRAM                     (Execute your program.)
SAVEPATH                    (Restore the original path.)
[F6]                        (Function key F6)
```

6.16 Tip: *You can save the current directory and return to it later automatically.*

When you are using different programs, you may need to use a different directory for a program to run and then want to return to the original directory when the program is completed. You can do this by redirecting the current directory name to a holding file and retrieving it later. Although this Tip is similar to the previous Tip on saving the current path, it requires another step.

To place the name of the current directory into a holding file, enter the following statement:

```
CD > SVCURDIR.BAT
```

If the current directory is A:\DIR1, it creates a new file SVCURDIR.TXT with the name of the current directory:

```
A:\DIR1
```

However, the entire command syntax needed for the CHDIR command to change the current directory to A:\DIR1 would be

```
CHDIR A:\DIR1
```

To get the entire command, append the name of the current directory to an already existing BAT file with just the "CHDIR." Each time it is used, the BAT file will be re-created with just the "CHDIR." Otherwise the BAT file would accumulate directory lines with each execution. Now create the first half of the command:

```
COPY CON CHDIR
CHDIR [ ] [F6] [Enter]   (CHDIR, space, function key F6. ENTER key)
```

Execute the batch file EXECHDIR.BAT from any directory to save the current directory, make a new directory, run a program, and restore the original directory. Now create EXECHDIR.BAT:

```
COPY CON EXECHDIR.BAT          (Create the batch file.)
TYPE CHDIR > SVCURDIR.BAT      (Create a fresh CHDIR command.)
CD >> SVCURDIR.BAT             (Append the current directory to CHDIR.)
CD A:\DIR3                     (Set new directory for your program.)
PROGRAM                        (Execute your program.)
SVCURDIR                       (Restore the original directory.)
[F6]                           (Function key F6)
```

6.17 Tip: *You can automatically keep a log file of each time someone boots your PC.*

Your AUTOEXEC.BAT file can automatically build a cumulative log file to show the date and time of the start of each session with your PC. The log file creation does not have to be apparent to the user. As an option, you could request the user's name for the log file.

Here is how you could create one that is not apparent to the user: First create an ENTER key/carriage return (CR) response file for redirected response input of the ENTER key:

```
COPY CON: CR
[Enter]                (Press the ENTER key)
[F6]                   (Function key F6)
```

Create the AUTOEXEC.BAT file to append the date and time to the automatic log file. The ">>" (the append output redirection parameter) creates the log file initially and adds a new line to the file thereafter. The "< CR" supplies the ENTER key or carriage return that is expected by the DOS DATE and TIME commands.

```
COPY CON AUTOEXEC.BAT          (Create the batch file.)
ECHO OFF                       (Suppress DOS command display.)
DATE >> LOGFILE.LOG < CR       (Append the DOS date to the log file.)
TIME >> LOGFILE.LOG < CR       (Append the DOS time to the log file.)
[F6]                           (Function key F6)
```

Here is how you can create a log file that captures the user's name.

```
COPY CON AUTOEXEC.BAT                    (Create the batch file.)
DATE >> LOGFILE.LOG < CR                 (Append the DOS date to the log file.)
TIME >> LOGFILE.LOG < CR                 (Append the DOS time to the log file.)
REM
REM ************************************************************
REM **  Enter LOGON followed by your name.                      **
REM **  Separate your first and last name with a dash.          **
REM **                                                          **
REM **        LOGON  Firstname-Lastname                         **
REM **                                                          **
REM ************************************************************
[F6]                                     (Function key F6)
```

Create the LOGON.BAT file to append the user's name to the automatic log file.

```
COPY CON LOGON.BAT          (Create the batch file.)
TYPE %1 >> LOGFILE.LOG      (Append the user name to the log file.)
[F6]                        (Function key F6)
```

With this method, the AUTOEXEC.BAT file will prompt for the user's name. The user then enters, for instance,

```
LOGON John-Jones
```

Note that the dash between first and last names is necessary so DOS will consider the entire name as one parameter.

Piping

Piping is telling DOS to pass the output from one program to be input for another program. DOS accomplishes this by creating a temporary file on the default disk for each piped information set. The first program writes out to a %PIPEx.$$$ file, which is then read in by the second program. You can think of this as a type of redirection, with the first program redirecting output to a file that is later used as redirected input to a second program. The difference is that with piping, DOS itself creates the intermediate file(s) automatically.

```
DIR ¦ MORE
```

is the same as

```
DIR    >   %PIPE1.$$$
MORE   <   %PIPE1.$$$
```

6.18 Trap: *The message "INTERMEDIATE FILE ERROR DURING PIPE" may be caused by several different conditions.*

If you get this message, DOS tried unsuccessfully to create a %PIPEx.$$$ temporary piping file. You may be out of directory space or file space on the disk. Another possibility is that the first program did not create the file you expected it to create, with the result that nothing for DOS was put through the pipe.

Filters

Filters are programs that take data in, perform a standard and perhaps simple process on the data, and then send the data out again. There are three standard filters included in DOS V2.0, V2.1, and V3.1:

- FIND Search a file for a specified string.
- MORE Display one screen of data, then wait for the user to strike a key to display more.
- SORT Sort files according to ASCII sequence. (DOS 3.1 uses a modified ASCII sequence.)

6.19 Tip: *Use the FIND filter together with the DIR command to look for certain files on one directory.*

Use DIR as input to FIND to locate files with a certain creation date:

```
DIR ¦ FIND "12-31-86"
```

Another example is

```
DIR ¦ FIND "LET"
```

In the first instance, the DIR command creates a current directory listing. Instead of displaying on the screen, the directory listing is "piped" as an input file to the FIND command. The FIND command will display on the screen only those files with a creation date of 12-31-86.

In the second instance, the FIND command will display on the screen only those files that have the string "LET" anywhere in the filename or an extension of "LET".

6.20 Trap: *FIND gives you a message "File not found" instead of the result you want if you put a parameter in the wrong position.*

Unlike most DOS commands, the FIND parameters /V/C/N go directly after the FIND command. The FIND parameters must be entered right after the FIND command and before the search string. For example:

```
FIND/V "DAT"
```

If you put the /V or /C or /N at the end, DOS will try to find a file named "V", "C", or "N".

6.21 Trap: *You may get unexpected results if you do not take into account the fact that FIND distinguishes between upper-case and lowercase letters.*

FIND treats uppercase and lowercase letters differently. You need to consider whether a string has uppercase or lowercase letters. For example, to make sure you find all occurrences of the string "different":

1. Use FIND for each possible combination:

```
FIND "different"
FIND "Different"
```

2. Eliminate characters where you can still define a unique string:

```
FIND "ifferent"
```

6.22 Trap: *You will get unexpected results with FIND if the string you specify is not sufficiently unique.*

You need to use enough characters to make the search string unique and keep the unwanted matches away. For instance, if you want a month for a date, use " 12-", not just "12".

6.23 Trap: *The FIND /C parameter does not count the number of times the character string occurs in a file.*

What /C really does is tell you how many lines contain the string at least once. If a string appears three times on one line, it counts only one time for /C.

You can demonstrate this by having FIND search a text file for a string that appears more than once in a line. For example, create the file MEN.TXT:

```
COPY CON MEN.TXT
Now is the time for all good men
to come to the aid of their party.
[F6]                                    (Function key F6)
```

Execute FIND to count the occurrences of "to":

```
FIND /C "to" MEN.TXT
```

If FIND were to count the occurrences of "to," it would return a count of 2. But FIND counts the lines where "to" occurs, and returns the line count:

```
----------MEN.TXT:    1
```

6.24 Tip: *You can get around the FIND wildcard prohibition.*

FIND does not let you use the wildcards "*" or "?" to search several files. You must enter each filename. By using the MULTFIND batch file, you can search multiple files to find data you know you have somewhere on one of the files in a directory. The MULTFIND batch file has only one statement you need to have for your batch file—the one that begins "FOR %%F IN." The rest just comments to help explain what it does.

```
REM*************************************************************
REM**************              MULTFIND.BAT         *************
REM**************  FIND A STRING IN MULTIPLE FILES   *************
REM*************************************************************
REM*                                                            *
REM*           MULTFIND   string   filename.ext   /findparms    *
REM*                                                            *
REM*************************************************************
REM* The FIND command does not permit wildcard (*, ?) filenames. *
REM* This batch file program allows you to search all of the    *
REM* files you want in one directory for a given string.        *
REM* Wildcards (*, ?) are allowed in the filename. FIND cannot  *
REM* search FIND.EXE. This causes the FIND program to fail.     *
REM* MULTFIND guards against that problem for you.              *
REM*                                                            *
REM* These parameters enable MULTFIND to search multiple files: *
REM*                                                            *
REM* %%1  "string"        Required: Search string enclosed in    *
REM*                                double quotes.              *
REM* %%2  filename.ext. Required: Use the full path name if the *
REM*                                file is not in the current  *
REM*                                directory. You may use "*" and *
REM*                                "?" in the filename.ext.    *
REM* %%3  /findparms      Optional: Parameter string of one or   *
REM*                                more of /V, /C, or /N.      *
REM*                                                            *
REM*************************************************************
REM*
   FOR %%F IN(%2)DO  IF NOT FIND.EXE==%%F    FIND  %3  %1  %%F
REM*
REM************************************************************
REM*******************  END OF MULTFIND.BAT  *******************
REM************************************************************
```

Here is an example of this batch file's use. Suppose you want to search all files that have the extension "txt" for the string "men." There are two files, WOMEN.TXT and MEN.TXT:

WOMEN.TXT	Now is the time for all good women to come to the aid of their party.
MEN.TXT	Now is the time for all good men to come to the aid of their party.

Note that the DOS FIND command expects the /V, /C, or /N right after FIND. MULTFIND expects them at the end of the statement line. Examples of using these commands follow:

```
MULTFIND  "men"  *.txt  /N
IF NOT FIND.EXE==WOMEN.TXT     FIND  /N  "men"  WOMEN.TXT
---------- WOMEN.TXT
[2]women to come to the aid
IF NOT FIND.EXE==MEN.TXT     FIND  /N  "men"  MEN.TXT
---------- MEN.TXT
[1]Now is the time for all good men
```

In this instance, MULTFIND searches all files with an extension of "txt" for the string "men." Additionally, the "/N" requests the line number for each line that contains the string "men."

6.25 Trap: *FIND will go blank if you forget a file specification.*

If you do not specify a source filename and path with the FIND command, FIND thinks you are entering the data through the keyboard. The screen will show only a blinking cursor while FIND waits for you to enter data. (You can then enter characters from the keyboard. If you enter the string you specified, DOS will echo back the string. This may be interesting, but not of much use.) To get out, press CTRL-BREAK. Then try again.

6.26 Trap: *You cannot FIND and redirect to the same file.*

DOS first allocates the new file, which destroys the existing data. You then FIND against an empty file, which leaves you with an empty result. If you want to change an existing file to have only lines that contain a specified string, you need to first redirect to a new file:

```
FIND "string" OLDFILE > NEWFILE
DEL OLDFILE
REN NEWFILE OLDFILE
```

6.27 Trap: *You cannot FIND "string" FIND.EXE.*

FIND will run against all files, even program files and other binary coded files. Although you may see some strange characters on the screen for COM and EXE files, FIND will work. One exception is the FIND program itself, FIND.EXE. If you try this, the program will fail, and you will need to reboot.

6.28 Tip: *You can use MORE even if the lines in the file are longer than 80 characters.*

DOS allows you to enter lines as long as 127 characters. MORE wraps lines longer than 80 characters to the next line (40 if your screen is 40 columns). You can demonstrate this by creating a 127-character file and redirecting it as input to the MORE command.

```
COPY CON 127CHAR.TXT
12345678901234567890123456789012345678901234567890123456789012...to col 80
12345678901234567890123456789012345678901234567
[F6]
```

Now display this file with the 127-character record using MORE:

```
MORE < 127CHAR.TXT
```

MORE will not truncate or overwrite the line, but will wrap the last 47 characters to the second line.

6.29 Trap: *If you redirect a text file with lines longer than 80 characters, MORE will wrap the print line over itself.*

You need to set the printer to 132 characters to print a line that exceeds 80 characters.

6.30 Tip: *SORT sorts on the internal ASCII code values.*

You need to be aware of the internal ASCII values that correspond to the printed characters, or you may be surprised when you sort a file. You can find a complete

list in an appendix in your BASIC manual and others. In particular, you should be aware that numbers precede uppercase letters, which precede lowercase letters. (DOS 3.1 uses a modified ASCII sequence. See Chapter 8 for details.)

6.31 Trap: *The SORT filter can hide your text files.*

SORT sorts the entire file using the size provided in the directory. Some text editors pad the last sector of a file with end-of-file markers (the CTRL-Z character), which will sort to the top of the file, because the ASCII print characters all have a higher internal numeric value. When another program, like TYPE, attempts to read the file, the first character encountered is end-of-file, and the program quits, assuming the file is ended. To get around this, you might try editing the file before sorting it, using a word processor or text editor that can delete the end-of-file markers.

Combining Redirection, Pipes, And Filters to Create Utility Functions

6.32 Tip: *You can create a directory sequenced by creation date, showing the files created most recently first.*

Suppose you want to know which files were created most recently when you do backups. By using DIR and FIND, you can identify files with a specific date. You can combine redirection, piping, FIND, SORT, and MORE to create and list the directory in reverse-date order.

The batch file REVDTDIR.BAT (REVerse-DaTe-DIRectory) will create such a directory listing and display it on the screen. REVDTDIR is a lengthy batch file with numerous REM statements. You do not need to enter the REM statements. Here is a description of the redirection, pipe, and filter techniques used in REVDTDIR:

- DIR ¦ SORT /R /+24 > REVDTDIR.WRK
  The directory listing contains the file creation date, month and day (MM-DD), beginning in column 24. This statement creates a directory listing and pipes it to the SORT filter. SORT sorts the directory in reverse by month and day, and the sorted output is redirected to the work file REVDTDIR.WRK.

- The new directory is built line by line by the FIND command. The first line is the "Volume" line, which creates the new directory file with the ">" redirection.
- From here on, lines are appended to the new directory with the ">>" append redirection.
- With the work file already in reverse month-day order, finding records in reverse year order builds a file in reverse year-month-day order. Note that beginning in 1991 you will need to add additional FIND statements to select future file years.
- FIND /V "----------" REVDTDIR.DIR > REVDTDIR.WRK
  Filters out the DOS FIND output comments, showing the filename where it found a string. This is only a cosmetic statement to improve the final appearance. (See the FIND/N example output for the MULTFIND.BAT batch file in Tip 6.24.)
- TYPE REVDTDIR.DIR ¦ MORE
  Redirects the rebuilt file to the MORE filter for a screen-by-screen display of the reverse directory. You can print REVDTDIR.DIR for a hard-copy directory:

```
REM ***************************************************
REM *****      R E V D T D I R . B A T          ******
REM *****      REVERSE DATE DIRECTORY           ******
REM ***************************************************
REM * REVDTDIR  creates a file of the current      *
REM * directory in reverse order of the file       *
REM * creation  date.  Because the standard DIR    *
REM * prints the date in MM-DD-YY order, you are   *
REM * not able to do a simple SORT /R, which you   *
REM * could have if it printed in YY-MM-DD order.  *
REM ***************************************************
REM
REM *** Sort by month,day in reverse order.      ***
REM *** Redirect sorted output to the work file.***
DIR  ¦ SORT /R/+24 >   REVDTDIR.WRK
REM *** Volume Label line creates a new file ">" ***
FIND "Volume in"       REVDTDIR.WRK  >  REVDTDIR.DIR
REM *** From now on we add to existing file ">>" ***
FIND "Directory"       REVDTDIR.WRK  >> REVDTDIR.DIR
REM *** Select, append years in descending order.***
FIND "-90"             REVDTDIR.WRK  >> REVDTDIR.DIR
FIND "-89"             REVDTDIR.WRK  >> REVDTDIR.DIR
FIND "-88"             REVDTDIR.WRK  >> REVDTDIR.DIR
FIND "-87"             REVDTDIR.WRK  >> REVDTDIR.DIR
FIND "-86"             REVDTDIR.WRK  >> REVDTDIR.DIR
FIND "-85"             REVDTDIR.WRK  >> REVDTDIR.DIR
FIND "-84"             REVDTDIR.WRK  >> REVDTDIR.DIR
FIND "-83"             REVDTDIR.WRK  >> REVDTDIR.DIR
FIND "-82"             REVDTDIR.WRK  >> REVDTDIR.DIR
```

```
FIND "-81"                  REVDTDIR.WRK  >> REVDTDIR.DIR
FIND "-80"                  REVDTDIR.WRK  >> REVDTDIR.DIR
REM *** This FIND /V copies everything BUT the    ***
REM *** references to the REVDTDIR.WRK file.      ***
FIND /V "___________" REVDTDIR.DIR  >  REVDTDIR.WRK
COPY REVDTDIR.WRK REVDTDIR.DIR
DEL  REVDTDIR.WRK
REM *** Get file count for end of reverse DIR.    ***
DIR ¦ FIND "File(s)" >> REVDTDIR.DIR
TYPE REVDTDIR.DIR ¦ MORE
REM*****************************************************
REM*****          END OF REVDTDIR.BAT                *****
REM*****************************************************
```

See Chapter 5 for Tips on some of the batch file techniques used here.

6.33 Tip: *You can create a directory listing file that lets you see the root directory, subdirectories, and all files.*

The DOS 2.X DIR command lists root (current directory) filenames and directory names, but does not show the filenames in the subdirectories. The DOS 2.X TREE /F command lists the subdirectories and subdirectory files, but does not show root directory files. You can't win! But you can. Put them together with redirection and filtering and you have a comprehensive directory tree file and report. The batch file DIRTREE2.BAT will create the complete directory for DOS 2.X.

DIRTREE2.BAT, which follows in the next listing, is a lengthy batch file with numerous REM statements. You do not need to enter the REM statements. Here is a description of the redirection, pipe, and filter techniques used in DIRTREE2.BAT:

- DIR > DIRTREE2.WK1
  The directory listing is placed in a work file.
- FIND "Volume in" DIRTREE2.WK1 > DIRTREE2.DIR
  The root directory portion is built line by line by the FIND command. The first line is the "Volume" line, which creates the new file with the ">" redirection.
- From here on, lines are appended to the new directory with the ">>" redirection.
- FIND "–9" DIRTREE2.WK1 > DIRTREE2.WK3
  FIND "–8" DIRTREE2.WK1 >> DIRTREE2.WK3
  SORT < DIRTREE2.WK3 > DIRTREE2.WK2
  These statements find all of the file lines and directory lines by finding the

- create year—any file created in 198x or 199x. You will need to add a FIND "—0" statement in the year 2000!

- FIND "<DIR>"
  Creates the summary listing of subdirectory names from the root directory.
  FIND /R "<DIR>"
  Creates the summary list of the root files that are not directory names, but are only regular filenames.

- FIND /V "---------" DIRTREE2.DIR > DIRTREE2.WK1
  Filters out the DOS FIND output comments, showing the filename where it found a string. This is only a cosmetic statement to improve the final appearance.

- TREE /F >> DIRTREE2.DIR
  Appends the regular TREE command display output, containing all subdirectory files, to the enhanced root directory file to make a combined root directory/tree directory.

- TYPE DIRTREE2.DIR ¦ MORE
  Redirects the combined file to the MORE filter for a screen-by-screen display of the complete directory. You can print DIRTREE2.DIR for a hard-copy directory.

DIRTREE2.BAT looks like this:

```
REM ***********************************************
REM *****       D I R T R E E 2 . B A T          ******
REM *****    COMPLETE  DIRECTORY for DOS 2.x     ******
REM ***********************************************
REM * DIRTREE2  creates a file of the current      *
REM * directory including names of all files       *
REM * and subdirectories, along with the TREE      *
REM * format report for names of subdirectories    *
REM * and files within each subdirectory.          *
REM ***********************************************
REM *
REM *** Create a directory listing work file     ***
DIR                                  >  DIRTREE2.WK1
REM *
REM *** Volume Label line creates a new file ">" ***
FIND "Volume in"        DIRTREE2.WK1 > DIRTREE2.DIR
REM *
REM *** From now on we add to existing file ">>" ***
FIND "Directory"       DIRTREE2.WK1  >> DIRTREE2.DIR
REM *
REM *** Select all files and directory lines by       ***
REM *** finding the create year (198x, 199x).         ***
REM *** SORT file and directory names to a work file. ***
FIND "-9"              DIRTREE2.WK1  >  DIRTREE2.WK3
FIND "-8"              DIRTREE2.WK1  >> DIRTREE2.WK3
```

```
SORT                    < DIRTREE2.WK3  >  DIRTREE2.WK2
REM *
REM *
REM *** First list just the subdirectory names.  ***
REM *** Select, append directory names in alpha order.***
FIND "<DIR>"              DIRTREE2.WK2  >> DIRTREE2.DIR
REM *
REM *** This FIND /V copies the root (current  ***
REM *** directory)  names to DIRTREE.DIR.       ***
FIND /V "<DIR>"           DIRTREE2.WK2  >> DIRTREE2.DIR
REM *
REM *** This FIND /V copies everything BUT the   ***
REM *** references to "DIRTREE2.WK1 and WK2"     ***
FIND /V "__________" DIRTREE2.DIR  >  DIRTREE2.WK1
REM *
COPY DIRTREE2.WK1 DIRTREE2.DIR
DEL  DIRTREE2.WK1
DEL  DIRTREE2.WK2
DEL  DIRTREE2.WK3
REM *
REM *** Get file count for end of root DIR.   ***
DIR ¦ FIND "File(s)"                    >>  DIRTREE2.DIR
REM *
TREE /F                                 >>  DIRTREE2.DIR
REM *
TYPE DIRTREE2.DIR ¦ MORE
```

The DOS 3.X DIR command lists root (current directory) filenames and directory names, but does not show the filenames in the subdirectories. The DOS 3.X TREE /F command lists root directory files, but does not have the concise list of root subdirectory names as it does for the lower subdirectories. With redirection, piping, and filtering, you can have a directory tree file and report the addition of the summary list of the subdirectories that appear in the root directory.

DIRTREE3.BAT, which follows, is a lengthy batch file with numerous REM statements. You do not need to enter the REM statements to obtain the tree report with the summary list of subdirectories that appear in the root directory. Here is a description of the redirection, pipe, and filter techniques used in DIRTREE3.BAT:

- DIR ¦ SORT > DIRTREE3.WRK
  The directory listing is sorted by filename and the sorted listing is placed in a work file.
- FIND "Volume in" DIRTREE3.WRK > DIRTREE3.DIR
  The root directory portion is built line by line by the FIND command. The first line is the "Volume" line, which creates the file with the ">" redirection.
- From here on, lines are appended to the new directory with the ">>" redirection.
- FIND "<DIR>" DIRTREE3.WRK >> DIRTREE3.DIR
  Creates the summary listing of subdirectory names from the root directory.

- FIND /V "----------" DIRTREE3.DIR > DIRTREE3.WRK
  Filters out the DOS FIND output comments, showing the filename where it found a string. This is only a cosmetic statement to improve the final appearance.

- TREE /F >> DIRTREE3.DIR
  Appends the regular TREE command display output, containing all subdirectory files, to the enhanced root directory file to make a combined root directory/tree directory.

- TYPE DIRTREE3.DIR ¦ MORE
  Redirects the combined file to the MORE filter for a screen-by-screen display of the complete directory. You can print DIRTREE3.DIR for a hard-copy directory.

DIRTREE3.BAT looks like this:

```
REM ***********************************************
REM *****       D I R T R E E 3 . B A T       ******
REM *****    COMPLETE  DIRECTORY for DOS 3.x  ******
REM ***********************************************
REM * DIRTREE3  creates a file of the current     *
REM * directory including names of all files      *
REM * and subdirectories, along with the TREE     *
REM * format report for names of subdirectories   *
REM * and files within each subdirectory.         *
REM ***********************************************
REM *
REM *** Sort by filename. Redirect to work file ***
DIR  ¦ SORT                          >  DIRTREE3.WRK
REM *
REM *** Volume Label line creates a new file ">" ***
FIND "Volume in"       DIRTREE3.WRK  >  DIRTREE3.DIR
REM *
REM *** From now on we add to existing file ">>" ***
FIND "Directory"       DIRTREE3.WRK  >> DIRTREE3.DIR
REM *
REM *** Select, append directory names in alpha order.***
FIND "<DIR>"           DIRTREE3.WRK  >> DIRTREE3.DIR
REM *
REM *** This FIND /V copies everything BUT the    ***
REM *** references to "DIRTREE.WRK"               ***
FIND /V "__________" DIRTREE3.DIR    >  DIRTREE3.WRK
REM *
COPY DIRTREE3.WRK DIRTREE3.DIR
DEL  DIRTREE3.WRK
REM *
REM *** Get file count for end of root DIR.   ***
DIR ¦ FIND "File(s)"                 >>  DIRTREE3.DIR
REM *
TREE /F                              >>  DIRTREE3.DIR
REM *
TYPE DIRTREE3.DIR ¦ MORE
```

6.34 Tip: *You can FIND all the filenames on your disk that contain any combination of one- or two-character strings.*

When you want to locate a file or files where you know some part(s) of the name, you can search all of the disk directories and subdirectories, hoping to see a familiar name. The batch file FILEFIND will identify all files that have

"string1"
"string1" and "string2"
"string1" or "string2".

FILEFIND.BAT is a lengthy batch file with numerous REM statements. You do not need to enter the REM statements when you copy FILEFIND.BAT. Here is a description of the redirection, pipe, and filter techniques used in FILEFIND.BAT:

- IF !==%1! GOTO ENDFILFD
  If the first parameter, %1, is blank, then ! will be equal to !. By using the one-character placeholder, you avoid the "Syntax error" message from DOS you would get if you just used "IF %1==%1" with no parameter entered.
- CHKDSK /V > TEMP1.$$$
  Creates a file with all of the filenames in TEMP1.$$$
- FIND %1 TEMP1.$$$ > TEMP2.$$$
  Creates a file in TEMP2.$$$ with all the filenames from TEMP1.$$$ containing "string1".
- FIND /V %1 TEMP1.$$$ > TEMP3.$$$
  Creates a file in TEMP2.$$$ with all the filenames from TEMP1.$$$ not containing "string 1".
- FIND %3 TEMP3.$$$ >> TEMP2.$$$
  Takes all filenames not containing "string1", but that do contain "string2", and adds them to the list.
- FIND /V "----------" FILEFIND.LST > TEMP2.$$$
  Filters out the DOS FIND output comments, showing the filename where it found a string. This statement only improves the final appearance.
- TYPE FILEFIND.LST ¦ MORE
  Redirects the selected filenames to the MORE filter for a screen-by-screen

display. You can print FILEFIND.LST for a hard-copy list of filenames. FILEFIND.LST looks like this:

```
REM ************************************************************************
REM ***************                FILEFIND.BAT                 ***************
REM ***************  FIND A FILE ANYWHERE ON A DISK   ***************
REM ************************************************************************
REM *            FILEFIND   "string1"                                      *
REM *            FILEFIND   "string1"   and    "string2"                   *
REM *            FILEFIND   "string1"   or     "string2"                   *
REM ************************************************************************
REM *
REM * You must enter at least one nonblank parameter.
REM * The statement below will be equal only if no parm is entered.
      IF !==%1! GOTO ENDFILFD
REM *
REM * CHKDSK /V creates a complete list of pathname\filenames on disk.
      CHKDSK /V > TEMP1.$$$
REM *
REM * Create a temporary file of filenames with "string1" in TEMP2.$$$
      FIND %1 TEMP1.$$$ > TEMP2.$$$
REM *
REM * If "string2" is blank, the final list is now in TEMP2.$$$
      IF !==%3! GOTO DONETMP2
REM *
REM * Identify which of "and" or "or" is requested.
      IF %2==AND GOTO HAVEAND
      IF %2==and GOTO HAVEAND
      IF %2==And GOTO HAVEAND
      IF %2==OR  GOTO HAVEOR
      IF %2==or  GOTO HAVEOR
      IF %2==Or  GOTO HAVEOR
REM *
REM * Need "and" or "or," so "string1" file is final.
      GOTO DONETMP2
REM *
:HAVEAND
REM * FIND  "string2" names in the file containing "string1" names.
      FIND %3 TEMP2.$$$ >   TEMP3.$$$
REM *
REM * The final list is now in TEMP3.  Put it in TEMP2.
      DEL  TEMP2.$$$
      REN  TEMP3.$$$  TEMP2.$$$
      GOTO DONETMP2
REM *
:HAVEOR
REM * FIND names in the file not containing "string1" into TEMP3.$$$.
      FIND /V %1 TEMP1.$$$  > TEMP3.$$$
REM *
REM * TEMP2.$$$ contains names with just "string1" both "string1"
REM * and "string2."  Add names with "string2" but not "string1."
      FIND    %3 TEMP3.$$$ >> TEMP2.$$$
REM *
:DONETMP2
REM * The final list is now in TEMP2.$$$
REM * Renaming is to have "FILEFIND.LST" in the final print.
      REN  TEMP2.$$$          FILEFIND.LST
```

```
      FIND /V "--------"        FILEFIND.LST > TEMP2.$$$
      DEL  FILEFIND.LST
      REN  TEMP2.$$$            FILEFIND.LST
REM *
:CLEANUP
      IF EXIST TEMP1.$$$ DEL TEMP1.$$$
      IF EXIST TEMP2.$$$ DEL TEMP2.$$$
      IF EXIST TEMP3.$$$ DEL TEMP3.$$$
REM *
      TYPE FILEFIND.LST ¦ MORE
:ENDFILFD
```

Below is a TREE listing for drive A. There are 14 files in the root directory and two files in subdirectory DIR1.

```
DIRECTORY PATH LISTING
Files:              AUTOEXEC.BAT
                    CHDIR
                    DIRTREE3.BAT
                    SVCURDIR.BAT
                    DOSRPF01.TIP
                    DATELOG
                    CR
                    DIRTREE3.DIR
                    127CHAR .TXT
                    TREE    .COM
                    DIRTREE2.DIR
                    DIRTREE2.BAT
                    FILEFIND.BAT
                    MORE    .COM
Path: \DIR1
Sub-directories:    None

Files:              NO
                    TREE    .COM
```

For example, if you execute FILEFIND.BAT for the previous tree:

```
FILEFIND "TREE" or "COM"
```

The result is

```
---------- FILEFIND.LST

      A:\DIR1\TREE.COM
      A:\DIRTREE3.BAT
      A:\DIRTREE3.DIR
      A:\TREE.COM
      A:\DIRTREE2.DIR
      A:\DIRTREE2.BAT

      A:\MORE.COM
```

6.35 Tip: *You can create a functional name and address database with DOS redirection, piping, and filters.*

PC-DOS can provide you with a simple database system where you can add records, delete records, find specific records, and list the entire file in sequence. You can do this with the four batch files given below.

PC-DOS accepts up to 127 characters as a batch file parameter. (It will wrap the characters around when you reach the right side of the screen.) Allowing for the batch file name, you can have records of about 120 characters long. You may choose to have records only 80 characters long. They will have a better appearance when displayed on the screen because they will keep data fields in a straight column. The records are being entered as a single parameter to a batch file, so you cannot key any delimiters into a record (space, comma, semicolon, equal sign, or the TAB key). Any data that follows a delimiter will be ignored by the batch file.

Create the batch file ADDREC to ADD a RECord to the file. The ECHO statement appends the record to the file.

```
COPY CON ADDREC.BAT
ECHO %1 >> DATABASE.TXT
[F6]                                (Function key F6)
```

Create records with fixed positions. You must fill in the gaps between fields with a nondelimiter like a period. Fixed positions allow you to sort on different data fields.

```
ADDREC Able.James.........ATTY..123.Main.St........San.Fran...CA.
94163.415-989-1234.....This.area.for.scratch.pad.notes.other.

ADDREC Aaron.David.........BROKR.456.Calif.St.......San.Fran...CA.
94105.415-989-4567.....This.area.for.scratch.pad.notes.other.

ADDREC Baker.Charles.......BARBR.789.Francisco......San.Jose...CA.
94101.415-989-8901.....This.area.for.scratch.pad.notes.other.
```

Another way to create your name and address records would be to not have fixed positions, and separate fields with a slash (/).

```
ADDREC Able.James/ATTY/123.Main.St/San.Fran/CA./94163/415-989-1234
/This.area.for.scratch.pad.notes.other/Additional chars used.

ADDREC Aaron.David/BROKR/456.Calif.St/San.Fran/CA.
94105.415-989-4567/This.area.for.scratch.pad.notes.other.

ADDREC Baker.Charles/BARBR/789.Francisco/San.Jose/CA.
94101/415-989-8901/This.area.for.scratch.pad.notes.other.
```

Create the batch file DELREC to DELete a RECord with a given string. FIND with the V option (Versus) keeps all records that do not have the string specified. This may be the long way around, but it works.

```
COPY CON DELREC.BAT
FIND /V %1 DATABASE.TXT > TEMP1.$$$
DEL DATABASE.TXT
REN TEMP1.$$$ DATABASE.TXT
[F6]
```

(Function key F6)

The following example FINDs Aaron and Baker to the new file. Able is not retained by the FIND command and is thus deleted.

```
DELREC "Able"
```

Before you DELREC, you should FINDREC first to see what records you will be deleting. You may want to change the string you specify to prevent unintentional deletion of records that happen to have a similar character string.

Create the batch file FINDREC to FIND a RECord with a given string.

```
COPY CON FINDREC.BAT
FIND    %1 DATABASE.TXT
[F6]
```

(Function key F6)

FIND (and display) all records for attorneys.

```
FINDREC "ATTY"
```

Create the batch file LISTREC to LIST the RECords in sequence. If you enter your fields in specific columns, you may optionally sort on one of the data fields within the record by entering a parameter to the LISTREC batch file of /+[column number]:

```
COPY CON LISTREC.BAT
SORT  %1  <   DATABASE.TXT ¦ MORE
[F6]
```

(Function key F6)

List the file in name sequence:

```
LISTREC
```

List the file in profession sequence. The profession field begins in column 21 of the record.

```
LISTREC /+21
```

If your database records are more than 80 characters long, you can create a batch file to print the database file in compressed mode (132 characters per line), then resume normal print mode:

```
COPY CON PRINT132.BAT
ECHO <Alt>015   >  LPT1     (Hold down ALT key, enter 015 from
                            the numeric keypad, not from the top row.)
COPY DATABASE.TXT LPT1      (Print the file.)

ECHO <Alt>146   >  LPT1     (Turn off the compressed mode:
                            Hold down ALT key, enter 146 from the numeric keypad, not
                            from the top row.)
[F6]                        (Function key F6.)
```

To print the database in 132-character mode, enter the statement:

```
PRINT132
```

By redirecting the ECHO of special characters to the printer, you can use printer features. For more information, see Chapter 7, "Controlling Peripherals."

7

Controlling Peripherals

Your personal computer is able to process, calculate, and move data internally. What makes your computer useful is the data you enter into it and the information it returns to you after processing. The devices that enable you to provide input and receive output are the *I/O peripherals.* I/O is an abbreviation for input/output. An example of input is pressing a key on the keyboard, which moves a character from the keyboard to the CPU and memory. An example of output is when a character moves from memory and the CPU to a printer. They are called peripherals because in the early days of computing, the CPU was always located in the center of the computer machine room, and the input/output devices were located around the sides, or periphery, of the machine room. Today, they are often built into the same box as the CPU, but the name "peripherals" is here to stay.

Peripherals are an integral part of your computer system. They are used by the CPU for any operation that requires gathering data externally to send data into the CPU or send it out. Because the peripherals are separate from the CPU and a peripheral device requires special controls, you need to instruct the CPU on how to communicate with and control each peripheral device. Often this is handled by whatever software you use, but sometimes you need DOS commands or facilities. This chapter explores Tips and Traps that will help you.

The peripherals we discuss here are the keyboard, monitor, printer, and communication ports.

Keyboard

7.01 Tip: *You can use the arrow and function keys for reentry and editing of DOS commands.*

If you have been frustrated when you made a one-keystroke error entering DOS commands on the command line and could not correct it by simply backspacing and overtyping or inserting, then you will appreciate this Tip. In the IBM *Guide to Operations* manual and the EDLIN section of the IBM PC-DOS manual are descriptions of the DOS editing keys. These keys enable you to correct mistakes quickly by entering or reentering a command. In addition, you can use certain commands in a safe logical sequence with only minor modification. The editing keys are listed here with definitions and examples.

F1 *Redisplays the previously entered line, one character at a time. (Cursor-right does this also.)*

This is helpful when you press one wrong key, as follows:

```
DEK *.txt
Bad command or file name
```

1. Press the F1 key twice:

```
DE
```

2. Enter the correct letter **L**:

```
DEL
```

3. Then press F3 once for the complete corrected command:

```
DEL *.txt
```

But what if you don't really want to delete one of those files? Before you DEL, do a DIR, then change the DIR to a DEL, using the above procedure only after you are sure you want to delete all of the files.

A second use for the F1 key is inserting characters into the command line.

1. Press F1 until you are at the position where you want to insert a character or characters.

2. Press the INSERT key. You won't see any change in the cursor, but have faith. You will insert.

3. Insert the character(s).

4. Press the F3 key to complete the command.

One more use for the F1 key is deleting characters from a command line. Use the F1 key to place the cursor on the line position corresponding to the offending character on the original line, then press the DEL key once for each character to be removed. You will not see anything here, but again, have faith. The characters are being deleted. Then press the F3 key to complete the remainder of the original command.

F2 *Redisplays all of the previously entered line up to the character you entered. Press F2 followed by a character.*

F3 *Redisplays the entire previously entered line. This is useful to repeat the previous command, say, if you want a second look at a directory list. F3 is more often used for showing a just modified command.*

F4 *The screen skips all of the characters in the previously entered line up to the character you enter. Press F4 followed by a character.*

F5 *Saves the currently displayed line for further editing. Use this key if you realize your mistake before you have executed the command. If you press the ESC key, you lose the command line and must start from scratch.*

You may be asking yourself whether it is worth the trouble to learn F1 through F5. F1 (or the cursor-right key) and F3 will definitely save you much time entering your DOS commands. F2 and F4 are marginally faster than pressing F1 a few times. If you can catch your mistakes before you press the ENTER key, you will appreciate F5 allowing you to correct them gracefully.

7.02 Tip: *You can enter any ASCII character, including graphics characters, from DOS into a file using the ALT key and the numeric keypad.*

To enter an ASCII character into the computer directly, hold down the ALT key, then enter the ASCII code for the character you want. A complete list of ASCII codes is in an appendix in your BASIC manual. Some examples of ASCII codes are

Decimal Value	ASCII Code Equivalent
007	sound the beep on the monitor
027	escape
048-057	0 through 9
065-090	A through Z
097-122	a through z
186	box, sides
187	box, upper-right corner
188	box, lower-right corner
200	box, lower-left corner
201	box, upper-left corner
205	box, top and bottom

This Tip can be extremely useful in creating batch files to produce special graphics images that can be imported to other programs.

You can create a batch file that will show a graphics box on the screen. With the proper printer, you can also print the box by sending the file containing the graphics codes to your printer. The characters for the monitor and the IBM printer are compatible. If you have a non-IBM printer, what you see on the printer may differ from what you see on the screen. You may be able to print the graphics image, however, if you identify what ASCII values to enter for your particular printer. Check your printer manual for details.

In the following example, you enter special ASCII codes into a file directly from the keyboard. This involves using the ALT key together with the numbers from the numeric keypad to enter the decimal equivalent of the special code. Here is the step-by-step method for entering these special codes:

1. You enter a special code when you see ALT followed immediately by three numeric digits:

ALT 000
through
ALT 255

If there is more than one special code, you will see the sequence repeated for each code:

ALT 255 ALT 255

2. Hold down the ALT key.

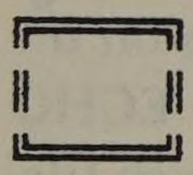
3. While keeping the ALT key depressed, type in the three numeric digits using the numeric keypad on the right side of the keyboard. Do not use the number keys on the top row of the keyboard.

4. After the three digits have been entered, release the ALT key. The screen will display the ^ and another character or will display an unusual-looking graphics character.

5. Only the first three digits that follow the ALT are keyed in this manner. After keying the first three numeric digits, resume key entry in the normal way.

To create a box image and save it in a file:

```
COPY CON BOX
<Alt>201<Alt>205<Alt>205<Alt>187    (top side and corners of box)
<Alt>186<Alt>032<Alt>032<Alt>186    (left, right sides)
<Alt>186<Alt>032<Alt>032<Alt>186    (left, right sides)
<Alt>200<Alt>205<Alt>205<Alt>188    (bottom side and corners of box)
F6                                  (Function key F6 for end-of-file)
```



This is the resulting "BOX" file. You can enlarge the box and create several files of boxes and other images for your graphics image library. Other graphics characters can lend more variety to your images. To send this to the printer, just issue the command TYPE BOX>PRN.

Monitor

7.03 Tip: *You can enlarge the cursor size to make the cursor easier to find.*

The standard two-scan line cursor can be difficult to find on a full page of text.

The following DOS DEBUG program will create a large cursor. Then, whenever you want a big cursor, execute the program BIGCURSR.COM:

```
DEBUG
-NBIGCURSR.COM
-E100 B4 01 B5 00 B1 07 CD 10 31 C0 CD 21
-RCX
CX 0000
:10
-W
-Q
```

7.04 Tip: *You can exercise control over what appears on the screen with the CLS and ECHO commands in batch files.*

When you execute a batch file, the batch file statements may appear in the middle of text that is already on the screen. For a clean appearance of the batch file, make the first statement in the batch file CLS. This will enable you to begin with a blank screen.

Another "clean-look" command for the batch file is the ECHO statement. ECHO OFF suppresses printing of DOS commands and batch file commands except for the PAUSE prompt. (To suppress output of some DOS statements, you may need to redirect the command output to "NUL." For details, see Chapter 6, "Redirection, Piping, and Filters.") You can selectively display messages on the screen by another use of the ECHO—it will display any text that follows on the same line regardless of the ECHO OFF or ECHO ON status. However, ECHO has one minor flaw: You cannot totally suppress screen display from a batch file. The initial ECHO OFF statement will show on the screen.

7.05 Tip: *You can use the MODE command to switch display adapters or set the display width for the color/graphics adapter.*

When you boot your PC, the settings on switch 1 of the system board indicate which monitor you have. You can reset the monitor with MODE. If you use monochrome and color monitors, you need to indicate which is to be active. MODE does this for you. You will want to use one of the following:

MODE MONO	Use the monochrome adapter.
MODE BW40	Use the color adapter, show black and white 40 characters.

MODE BW80 Use the color adapter, show black and white 80 characters.
MODE CO40 Use the color adapter, show color 40 characters.
MODE CO80 Use the color adapter, show color 80 characters.

You can also use the MODE command to align your color monitor if the line pattern is off to the right or the left:

MODE 80,R Shifts line two characters to the right (80-column screen).
MODE 80,L Shifts line two characters to the left (80-column screen).
MODE 40,R Shifts line one character to the right (40-column screen).
MODE 40,L Shifts line one character to the left (40-column screen).

If you place another parameter, "T", on the end, you get a test pattern to help you judge the alignment. Each time you execute the command, the screen line is shifted again. Once you know how many characters to shift, you can place MODE in your AUTOEXEC.BAT file to align your monitor automatically each time you boot.

Printers and Plotters

7.06 Tip: *If you have activated the GRAPHICS.COM program from DOS, you can then use SHIFT-PRTSC to print a graphics screen to the printer from some software.*

There are some qualifications for following this Tip:

- The ASCII values used for text graphics (ASCII 176-233) will work on IBM printers, but not on Epson printers.
- It does not work unless you have a color/graphics adapter. If you have a monochrome adapter only, there is no effect at all.
- It prints your screen portrait (normal) in 320×200 mode. It prints your screen landscape (sideways) in 640×200 mode.
- It does not work with some programs, such as Lotus 1-2-3.

7.07 Trap: *When you print with a serial printer, you should use the MODE command to assign LPT1 to the proper serial*

port with the proper protocol parameters, or DOS and some applications programs won't be able to address the printer properly.

Some programs, like Lotus 1-2-3, let you control to which parallel or serial port they direct their standard printer output. The MODE command is unnecessary here as far as standard printer output from that program is concerned. However, if you want to use a program that does not give you this control, or if you want to be able to use SHIFT-PRTSC to print the contents of your screen, or the TYPE or PRINT command to print files, then use the MODE command both to set protocol parameters for your printer and to assign LPT1 or LPT2 to it as soon as you boot your PC.

It is convenient to have the two MODE command lines in the file named AUTOEXEC.BAT. Here is a typical example:

```
MODE COM1:12,N,8,1,P
MODE LPT1:=COM1
```

(If you sometimes want to use a parallel printer, you could place a PAUSE command in your batch file before these two mode commands. A CTRL-BREAK would then terminate the batch file.)

You can see that assigning LPT1 to a COM port is a two-step process:

- First, initialize the asynchronous communications adapter to identify the printer protocol parameters.
- Second, redirect the parallel printer output to the COM serial port.

The operations manual for your printer tells you what protocol parameters are used by your serial printer. The first MODE statement initializes COM1 as follows:

12	Baud transmission rate is 1200 bits per second.
N	Parity is None for error checking.
8	Eight bits per character.
1	1 stop bit per character.
P	Means "This is a printer. Don't time out the port if you have errors. Keep on trying." This enables you to insert new paper into the printer. If you really do have a problem, like a printer breakdown, you can override the "P" by pressing the CTRL-BREAK key combination.

7.08 Trap: *DOS 2.X requires you to have a colon after the device identifier in the MODE command.*

In DOS 2.X, you must have a colon (:) after the device identifier (LPT#: or COM#:), which is right after the MODE. In DOS 3.X, the post-device-name colon is optional.

7.09 Tip: *If you sometimes use a parallel printer and sometimes a serial printer, you can use the MODE command to assign each in turn to LPT1 or LPT2.*

As described in the previous Tip, you can redirect LPT1, LPT2, or LPT3 to a serial port COM1 or COM2. You can also reset LPT1 and LPT2 back to their standard status by issuing another MODE command to reset them.

To restore LPT# to a normal parallel printer status, enter

```
MODE LPT# n,m      (Where "n" is 80 or 132 characters per line,
                   and "m" is 6 or 8 lines per inch.)

MODE LPT# 80,6     (Restores LPT# to parallel 80 characters per line
                   and 6 lines per inch.)
```

You can therefore set and reset your parallel and serial printers to meet any combination of needs by setting and resetting the status with the MODE commands. If you assigned LPT# to one of the COM ports to use a serial printer, you can reassign it to a parallel port by simply issuing the command MODE LPT1 or MODE LPT2 without additional arguments.

7.10 Tip: *Option 1 of the MODE command lets you change the lines per inch and/or characters per line on a printer.*

If your word processor is limited, or if you issue the DOS commands COPY, PRINT, and TYPE to do some printing, you can use MODE to alter the vertical spacing on your printer.

```
MODE LPT# n,m      (Where "n" is 80 or 132 characters per line,
                   and "m" is 6 or 8 lines per inch.)

MODE LPT1 132,8    (Set LPT1 to 132 characters per line
                   and 8 lines per inch.)
```

You use MODE again to restore the printer to the way you want it.

7.11 Trap: *MODE does not tell you if you enter an invalid parameter.*

The MODE command recognizes only "80" and "132" for characters per line, and only "6" or "8" for lines per inch. If you enter any other values, MODE simply ignores them, does not change the existing parameter settings, and does not give any warning or error messages.

7.12 Tip: *You can use the PRINT command to send a queue of files to a printer while you work on other tasks.*

When you use TYPE or COPY to print a file, you have to wait while the file is printing. This can be inconvenient if you are printing long files. You can avoid the long wait by using the PRINT command, which can operate in the background as you continue to work on whatever you want—editing, spreadsheet work, and so on. DOS 2.X allows up to ten files queued at any given time. DOS 3.X allows up to 32 files and also enables you to fine-tune the PRINT command for better performance.

7.13 Tip: *You can send any printer control (including the escape character) from DOS to your printer with the ALT key and the numeric keypad.*

To enter an ASCII character into the computer directly, hold down the ALT key and then enter the ASCII code for the character you want. A complete list of ASCII codes is in the appendix of your BASIC manual. Some examples of ASCII codes are

Decimal Value	ASCII Code Equivalent
013	carriage return
027	escape
048-057	0 through 9
065-090	A through Z

This can be extremely helpful in creating batch files to control such printer features as double-strike and compressed mode. An often needed code is the escape code <Esc>, which has an internal value of ASCII 27. Whenever DOS receives an escape from the keyboard, it simply stops accepting data and resets for a new command entry. You can, however, trick the keyboard into accepting a disguised escape code by adding 128 to the 27 for a new value of ASCII 155. This simply changes the high-order eighth bit in the character from a 0 to a 1. PC-DOS no longer recognizes it as the escape character. However, your printer ignores the eighth bit, because it uses only the first seven bits of each character, and your printer interprets both ASCII 27 and ASCII 155 as the escape character.

Note that in the following examples, the Epson printer control codes are used. If your printer is not an Epson, consult your printer manual for the corresponding printer control codes. (See Tip 7.02 on entering special codes.)

1. Create the batch file DBLSTRIK.BAT to direct the printer to print a file in double-strike mode, then resume normal print mode:

```
COPY CON DBLSTRIK.BAT
ECHO <Alt>155G > LPT1     (ESC-G turns on double-strike mode.)
COPY %1          LPT1     (%1 is the file to be printed.)
ECHO <Alt>155H > LPT1     (ESC-H turns off double-strike mode.)
<F6>                      (Function key F6)
```

2. To print the file GOODMEN.TXT in double-strike mode, enter

```
DBLSTRIK GOODMEN.TXT
```

The printer prints the file in double-strike, then returns to single-strike mode:

```
Now is the time for all good men
to come to the aid of their party.
```

3. To create a batch file that directs your printer to print a file in compressed mode, then resume normal print mode:

```
COPY CON COMPRESS.BAT
ECHO <Alt>015   > LPT1    (ASCII 015 turns on compressed mode.)
COPY %1           LPT1    (%1 is the file to be printed.)
ECHO <Alt>146   > LPT1    (ASCII 146 turns off compressed mode.)
<F6>                      (Function key F6)
```

4. To print the file GOODMEN.TXT in compressed (132) character mode, enter

```
COMPRESS GOODMEN.TXT
```

The printer will print the file in compressed mode, then return to normal mode:

```
Now is the time for all good men
to come to the aid of their party.
```

Now that you have seen how it works, you can create batch files to initiate any number of printer controls and features.

7.14 Tip: *Construct a system of menu-driven batch files that let you send any escape code to a printer with one keystroke.*

You can expand on the previous Tip (using individual batch files for a single printer feature). By creating a file for each feature and using numbers for batch filenames, you can initialize any combination of features your printer will accept.

You can use the COPY CON FILENAME approach, the EDLIN editor, or another text editor to create the following batch files for your printer feature menu:

```
COPY CON PRINFEAT.BAT
REM ************************************************************************
REM ***************                PRINFEAT.BAT                ***************
REM ***************  SPECIFY PRINTER FEATURE CODES   ***************
REM ************************************************************************
REM *                                                                      *
REM * This batch file displays a menu of printer features                  *
REM * which can be invoked by executing other batch files.                 *
REM *                                                                      *
REM * You first select the feature or features you want to                 *
REM * use while printing your file from the menu.                          *
REM * You then print your file using any program or the                    *
REM * DOS commands PRINT, COPY, or TYPE.                                   *
REM *                                                                      *
REM *                                                                      *
REM ************************************************************************
REM
PAUSE **** PRESS ANY KEY WHEN READY TO SELECT PRINTER FEATURES *****
CLS
REM ************************************************************************
REM ******************     PRINTER OPTIONS MENU    ***********************
REM ************************************************************************
REM *                                                                      *
REM *    1. Compressed Print          11. Turn off Compressed Print        *
REM *                                                                      *
REM *    2. Double Strike             12. Turn off Double Strike           *
REM *                                                                      *
REM *    3. Emphasized                13. Turn off Emphasized              *
REM *                                                                      *
REM *    4. Double Width (Enter 4                                          *
REM *          followed by up to 9 words.)                                 *
REM *    5. 6 Lines Per Inch                                               *
REM *                                                                      *
REM *    6. 8 Lines Per Inch                                               *
REM *                                                                      *
REM *    7. Lines Per Page  (Enter  " 7  #lines"  )                        *
REM *                                                                      *
REM *   10. Turn off Features 1 through 3                                  *
REM *                                                                      *
REM *                                                                      *
REM ************************************************************************
```

For example, to print filename.txt both emphasized and double-strike, at 50 lines per page, take the following steps: To turn on double-strike mode, enter **2**. The following message will appear on your screen:

```
** Double Strike Print On **
```

To turn on emphasized mode in addition to double-strike, enter **3**. To change to only 50 lines per page, enter **7 50**. To print a file emphasized and double-strike, enter **PRINT GOODMEN.TXT**. The output will look like this:

```
Now is the time for all good men
to come to the aid of their party.
```

To turn off double-strike and turn off emphasized mode, enter **10**.
To return to 66 lines per page, enter **7 66**.

The following section creates the batch files for the print options menu.

Create a batch file to turn on compressed mode:

```
COPY CON 1.BAT
ECHO OFF
ECHO <Alt>015   > LPT1          (ASCII 015 turns on compressed mode.)
ECHO ** Compressed On    **
<F6>                            (Function key F6)
```

Create a batch file to turn off compressed mode:

```
COPY CON 11.BAT
ECHO OFF
ECHO <Alt>146   > LPT1          (ASCII 146 turns off compressed mode.)
ECHO ** Compressed Off   **
<F6>                            (Function key F6)
```

Catch a batch file to turn on double-strike mode:

```
COPY CON 2.BAT
ECHO OFF
ECHO <Alt>155G > LPT1           (ESC-G turns on double-strike mode.)
ECHO ** Double Strike On **
<F6>                            (Function key F6)
```

Create a batch file to turn off double-strike mode:

```
COPY CON 12.BAT
ECHO OFF
ECHO <Alt>155H > LPT1           (ESC-H turns off double-strike mode.)
ECHO ** Double Strike Off **
<F6>                            (Function key F6)
```

Create a batch file to turn on emphasized mode:

```
COPY CON 3.BAT
ECHO OFF
ECHO <Alt>155E  > LPT1          (ESC-E turns on emphasized mode.)
ECHO ** Emphasized On     **
<F6>                            (Function key F6)
```

Create a batch file to turn off emphasized mode:

```
COPY CON 13.BAT
ECHO OFF
ECHO <Alt>155F  > LPT1          (ESC-F turns off emphasized mode.)
ECHO ** Emphasized Off    **
<F6>                            (Function key F6)
```

Create a batch file to print one line in double-width mode. The double-width feature automatically turns off after one line. This batch file prints up to nine words, which are shown as %1 through %9 in the batch file.

```
COPY CON 4.BAT
ECHO OFF
ECHO <Alt>142 %1 %2 %3 %4 %5 %6 %7 %8 %9 > LPT1  ASCII 142 turns on double-
                                                 width mode. %1 through
                                                 %9 are the nine words
                                                 which can be printed.
ECHO *** Double Width Line Printed ***
<F6>                                             (Function key F6)
```

Create a batch file to turn on six-lines-per-inch mode.

```
COPY CON 5.BAT
ECHO OFF
ECHO <Alt>1552  > LPT1          (ESC-2 turns on 6 lines-per-inch.)
ECHO ** 6 Lines Per Inch  **
<F6>                            (Function key F6)
```

Create a batch file to turn on eight lines-per-inch mode.

```
COPY CON 6.BAT
ECHO OFF
ECHO <Alt>1550  > LPT1          (ESC-0 turns on 8 lines per inch.)
ECHO ** 8 Lines Per Inch  **
<F6>                            (Function key F6)
```

Create the Epson control code file to reset lines/page, ESC-Cn, in which "n" is the number of lines. When you execute 7.BAT, you need to enter the lines per page.

```
COPY CON 7.BAT
ECHO OFF
IF !==%1! GOTO NOLINES
```

(Make sure line number is entered. If the number of lines per page is not entered, then there will be only a "!" on both sides of the equal sign.)

```
ECHO <Alt>155C%1 > LPT1
```

(ESC-Cn sets lines per page in which "n" is the number of lines and is the %1 parameter.)

```
ECHO ** %1 Lines Per Page  **
GOTO END7BAT
:NOLINES
ECHO ****** LINES PER PAGE WERE NOT ENTERED AFTER 7  *******
PAUSE
:END7BAT
<F6>
```

(Function key F6)

Create a batch file to turn off features one through three:

```
COPY CON 10.BAT
ECHO OFF
ECHO <Alt>146    > LPT1
ECHO <Alt>155H   > LPT1
ECHO <Alt>155F   > LPT1
** Comp/Str/Emph Off **
<F6>
```

(ASCII 146 turns off compressed mode.)
(ESC-H turns off double-strike mode.)
(ESC-F turns off emphasized mode.)

(Function key F6)

7.15 Tip: *You can quickly send escape sequences to your printer by using the PROMPT command.*

The PROMPT command allows you to send escape characters to the display. A typical requirement, however, is to be able to send the escape character to the printer to enable such special modes as compressed print or bold print. Since you cannot simply copy from the keyboard to the printer with a COPY CON: PRN: command (DOS doesn't allow the escape character to pass through), you must use another method. A requirement for this Tip to work is that you do not have the device driver ANSI.SYS loaded by your CONFIG.SYS file. If you don't know whether you use ANSI.SYS, just issue a TYPE CONFIG.SYS command, to see if a DEVICE=ANSI.SYS command is in the CONFIG.SYS file. If there is, then this method will not work.

You will get the printer to "see" the escape sequence you want by using the DOS control-print screen (CTRL-PRTSC) function. You will set up your prompt to be the escape character, cause DOS to echo the characters to the printer, and add any necessary control codes. You then can turn off the print screen function and restore the prompt to the usual one. The key to getting this method to work is doing things in *exactly* the correct order and knowing what control codes you must send out.

The example we will use is for an Epson printer. We will set it to compressed print. The command sequence required is

```
prompt $e <Ctrl><PrtSc><cr>
G<Ctrl><PrtSc><backspace><cr>
prompt<cr>
```

In this example, the <cr> means press the ENTER key and <backspace> means press the BACKSPACE key. In general, you would type as many backspaces as you have control characters, so that when you press ENTER at the end, DOS won't think that they are a command name and try to execute it. If you want to turn off the compressed print on an Epson printer, you repeat the procedure, substituting a capital H for the capital G. Other printers may use different control codes; see your printer manual.

7.16 Trap: *Copy to the printer doesn't expand tabs.*

If you want to print a file that contains tabs and try to use the COPY command to copy the file to the printer, you may find that the tabs are not automatically expanded to an appropriate number of spaces. Some printers will expand the tabs for you, but if your printer does not, then the COPY *filename* PRN: method will not produce the results you desire. However, there are two ways around this problem. The first is to use the PRINT command supplied with the DOS 2.X system. This print-spooling command allows you to print files that contain tabs and expand them automatically. A second method is to turn on the screen echo to printer condition by holding down CTRL-PRTSC and then issuing a TYPE *filename* command. This method will also expand the tabs. If you have a file that contains graphics data, however, you are better off using the COPY *filename* PRN: method to print the contents of your unmodified file.

7.17 Tip: *If you have two parallel printers, you can swap their LPT1 and LPT2 identities.*

Some programs will print only to LPT1. You can have a printer on LPT2 and direct the LPT1 data to it with the following PRINTSW program. Each time you execute PRINTSW.COM, the addresses for LPT1 and LPT2 are swapped. The program can only be used with DOS 2.X and higher. Create the program with

DEBUG in the following way:

```
DEBUG
-NPRINTSW.COM
-A
XXXX:0100 MOV         AX,0040
XXXX:0103 MOV         DS,AX
XXXX:0105 MOV         SI,0008
XXXX:0108 MOV         AX,[SI]
XXXX:010A XCHG        [SI+02],AX
XXXX:010D MOV         [SI,AX]
XXXX:010F MOV         AX,004C
XXXX:0112 INT         21
XXXX:0114
-RCX
CX 0000    0014
-W
Writing 14 bytes
-Q
```

You want to test print a document on the dot matrix parallel printer before you print it on the letter-quality parallel printer. LPT1 is normally your dot matrix printer. Print the document on the dot matrix printer, which is now assigned to LPT1:

```
TYPE GOODMEN.TXT    > LPT1
```

Swap the dot matrix printer to LPT2 and assign the parallel printer to LPT1:

```
PRINTSW
```

Print the document on the parallel printer, which is now assigned to LPT1:

```
TYPE GOODMEN.TXT    > LPT1
```

Return the dot matrix printer to LPT1 and assign the parallel printer back to LPT2:

```
PRINTSW
```

Modems and Communications Ports

7.18 Trap: *Difficulties in addressing communications ports are often due to hardware problems.*

DOS allows for two logical serial ports, COM1 and COM2. You can have either one or two physical serial ports on one or more expansion cards inserted into your computer. A communications program can then address either or both. Many memory expansion cards allow for one or two serial ports on the same card. You may also have an expansion card that contains nothing but a serial port (or perhaps a serial port and a parallel port). Whatever the physical configuration, DOS must have a way of mapping the one or two physical serial ports to the logical COM1, COM2, or both. A jumper setting on the expansion card determines the mapping. A common problem is having one serial port in your machine that is physically set to COM1 and then adding an expansion card that contains a second serial port, also set to COM1. In such a case, DOS will get confused and may not be able to address either port properly. To solve the problem, find out from the expansion card manual how to set the jumper on that card to point to COM2 and then set that jumper accordingly.

Several other common hardware problems can cause a communications program to have trouble addressing a serial port. The most common relate to the cable connecting the modem with the serial port. This cable must be firmly attached at both the PC end and the modem end. We advise you to tighten the screws at both ends to make sure you have a good connection. Even if the connection is good at both ends, you could have a faulty cable; if you suspect this, try substituting another cable.

The other hardware problems concern the modem and the phone line. Of course, the modem must be plugged in and turned on. If you suspect that the modem or the phone line may be faulty, try substituting another modem or changing lines.

A simple way to check a COM port from DOS if you have an external modem (as opposed to one on an inboard expansion card) is to copy some characters to it. Then watch for the Send Data light, if you have one, to see whether the modem is receiving the data you're pumping out the serial port. You can do this in the following way:

```
COPY CON: COM1
XXXXXXXX
```

After these two commands, press F6 to close the file you're copying and press ENTER, while watching the modem closely. If you see the Send Data light blink, then chances are you have no problems with the serial port, the cable, and the modem. If you do not see the light blink, you have a problem with your serial port or cable. In such a case, you are likely to see the following message:

```
Write fault error writing device COMn
Abort, Retry, Ignore?
```

The simplest serial port problem occurs when you have your cable connected to COM1 instead of COM2, or vice versa. It's also possible, as we saw earlier, that you have two physical serial ports assigned to the same logical COM port.

7.19 Trap: *A modem problem could be caused by an inadequate phone connection.*

If you have a problem with a new modem hookup, check the phone line connectors you are using. Some modems require four wires, but some modular phone extension cords and Y-connectors have only two wires.

7.20 Tip: *The MODE command lets you set the protocol parameters for COM1 or COM2 from DOS.*

The typical use for the MODE command is specifying printer parameters and redirecting parallel printer output to a serial port. When you use a COM port for communications, you need to tell the PC asynchronous communications adapter the communications protocol. You will do this only when you are initially setting up your communications with the remote computer. You will use a commercially packaged communications program to do your transmissions. Most of these programs enable you to set the protocol parameters from within the program. The remote site and your modem manual tell you what protocol parameters are needed.

If, however, you find that you need to do this directly, you then put the MODE command into your AUTOEXEC.BAT file to automatically initialize the COM port. If you communicate with more than one remote computer, you may need to prepare an additional batch file for the MODE protocol parameters for each remote computer. The MODE format is

MODE COM#:*baud,parity,databits,stopbits.*

In this example, the MODE statement specifies the protocol parameters to COM1 for transmission of data over phone lines using a modem:

```
MODE COM1:12,N,8,1
```

12 Baud transmission rate is 1200 bits per second.

N Parity is None for error checking.

8 Eight bits per character.

1 1 stop bit per character.

Note that the optional parameter "P" for "printer" is not entered. The "P" parameter is used only if the COM1 or COM2 port is being used for a serial printer.

Refer to your user's manual for the features available on the modem. Refer also to the specifications of the computer that will be on the other end of the line for verification that the correct protocol parameters are specified.

8

Advanced Topics

The Tips and Traps in this chapter for the most part involve techniques that will be of interest only to the advanced user. Included are tips on using RAM disks, the ANSI.SYS device driver, and the DEBUG utility program. The final section in this chapter presents some information related specifically to version 3.X of DOS. This version is standard for the IBM PC/AT and is different in several respects from DOS 2.X, which has been the major focus of this book.

RAM Disks, Hard Disks, and Buffers

8.01 Tip: *RAM disks can speed up your programs.*

A *RAM disk* is software that can be added to DOS to allow it to use portions of memory as a simulated disk drive. There are many such RAM disk software packages, and often one will come with the purchase of a multifunction board or memory board for your computer. The purpose of a RAM disk is to fool DOS into treating a portion of the computer's memory as if it were a disk. DOS can therefore store information on the RAM disk and read it back much faster than it

can with a real disk. Since much of the time required for a program to execute can be consumed by moving data to and from a disk, a RAM disk can really speed things up.

When you set up a RAM disk, you must decide what parts of your programs and data files should be on it. The best guideline to follow is to put on the RAM disk the programs and data that are used most often when you run your application. These could include programs that use "overlays" or program fragments that are brought into the computer's memory when needed, and data files that access larger files or store frequently accessed or changed data. (But see the following Trap.)

Because it is not unusual for programs doing many RAM disk operations to be speeded up by a factor of two or three, buying a bit more memory and going to the trouble of installing the RAM disk may be worthwhile.

8.02 Trap: *Data stored on a RAM disk is lost if the computer is turned off.*

One of the problems with using a RAM disk is that the information is stored in memory and not on some kind of physical disk. Therefore, you must be extremely careful to save any needed information to a physical disk before turning off the computer, or the information will be lost. A good way to handle this is to use a batch file to move your program and data files to the RAM disk, change the default drive to the RAM disk, invoke the program, and save the data files back to the physical disk after the program completes. By automating this procedure, you are less likely to lose data.

8.03 Tip: *DOS provides you with a free RAM disk.*

Tucked away in the section dealing with device drivers of your DOS 2.X manual is the program source code for a RAM disk called VDISK. This program is written in assembly language and can be put together by the IBM Macro Assembler or even with DEBUG. Although implementing this RAM disk may be too technical for you, it is an interesting exercise and will give you a RAM disk if you don't already have one. A program called VDISK.SYS comes with the DOS 3.0 package, so if you have that version of DOS, you also have a RAM disk ready to use.

8.04 Tip: *You can operate your computer from a modem or another terminal.*

The CTTY or change console command in DOS allows you to have your computer controlled either by another terminal or a modem connection to another location. Although limited in some ways, this command allows the DOS system to take commands from one of its COM ports, COM1 or COM2. Perhaps the most interesting use of CTTY is to connect your computer to a modem and have it controlled from another remote site. To do this, first set up your COM port with the MODE command, giving your modem the commands it needs to set itself into autoanswer mode (the COPY CON: COM1: trick would probably be the easiest), and then type **CTTY COM1**. A good setup command for the Hayes modem is AT S0=1 E0 Q1. Your computer will then answer a call on your modem line and respond to commands by the remote terminal. When you are ready to turn control over to the local PC again, all you need to do is type the command **CTTY CON** on the remote terminal. This transfers control back to the console.

8.05 Trap: *Many programs cannot be used remotely using CTTY.*

Programs that make heavy use of the PC's screen for drawing graphics on it or directly accessing the screen memory to update the screen will not work properly if the CTTY command has changed the control console to a terminal other than the normal PC display. The programs that most often will not work are word-processing programs and those displaying graphics. Programs written to run on MS-DOS are more likely to operate correctly than programs that run only on PC-DOS. However, most normal command line commands such as CHDIR, MKDIR, DIR, DATE, ERASE, PATH, RENAME, TIME, RMDIR, VER, VERIFY, and VOL should work fine.

8.06 Tip: *The BUFFERS statement in CONFIG.SYS can speed up your application.*

DOS supports a special file called CONFIG.SYS, which it always reads when the system is booted. This file tells DOS what device drivers you want to use and what

space allocators you want for reading and writing files. One of these space allocators is the BUFFERS statement. An example of a BUFFERS statement in your CONFIG.SYS file is

```
BUFFERS=4
```

If this statement is found by DOS in your CONFIG.SYS file when the system is booted, DOS will allocate four buffers for reading and writing files. These buffers are simply scratch pad areas of memory that DOS uses to save bits of files during the transfer of data from the disk to your program. Because these buffers each take up 528 bytes of memory, you shouldn't have more buffers than you need. However, since DOS passes your data through these buffers, your system could slow down if you don't have enough. How many buffers are the right number? The answer to this tough question depends on the circumstances.

Each application program uses the resources of the computer differently. Some may do a lot of arithmetic, some may access the disk a great deal, and some may wring every drop of capability out of the display. Even two programs that are both "disk-intensive" can access the disk in very different ways. Like database management systems, random-access programs that tend to use the same records more than once generally benefit most from buffers. Since no two programs use the computer exactly alike, the best number of DOS buffers for one may not be the best number for another.

If you have the time and inclination, you can easily optimize the number of buffers for your system. Select a computer application you do frequently. When you have set up one that can be run repeatedly, create a file called CONFIG.SYS on your boot disk and put a BUFFERS statement in it like the one just shown. Then reboot the system and time your application. Repeat this process, each time increasing the number of buffers by, say, two or four until you have reached the maximum amount you think your system will allow. When you have all the numbers, pick the lowest execution time and go with that.

You may discover that varying the number of buffers defined for your system produces a wide difference in results. The effects are extremely system- and application-dependent, and can vary insignificantly or substantially. Therefore, you should try this Tip for yourself, on your system, with your specific application.

8.07 Trap: *DOS treats the CONTROL-Z character as the end of a file, even when it is not.*

In general, the DOS system treats the character CONTROL-Z as the end of a file.

This is usually safe, since CONTROL-Z is an unusual character to find in a file unless it was put there to indicate the end. If a CONTROL-Z character should creep into your file somewhere other than at the end, you have a problem. DOS will insist that the CONTROL-Z is the end of the file and ignore any data after it. To avoid this unpleasant situation, take advantage of a little-known feature of the EDLIN editor that comes with your DOS system. Type **EDLIN filename /B**, and the EDLIN editor will read all of your file including data after the CONTROL-Z character. You are then free to edit out the CONTROL-Zs that are in the wrong places and save the file back to the disk.

Consider the following illustration. Suppose that we create a 5-line test file with a CONTROL-Z in line 4. If we type it or use EDLIN without the /B option, we see only the first three lines:

```
C:\>type test
This is line one.
This is line two.
Line four will be a cntl Z.

C:\>edlin test
End of input file
*1
     1:*This is line one.
     2: This is line two.
     3: Line four will be a cntl Z.
*q
Abort edit (Y/N)? y
```

But if we use EDLIN with /B, we can see and edit the CONTROL-Z.

```
C:\>edlin test /b
End of input file
*1
     1:*This is line one.
     2: This is line two.
     3: Line four will be a cntl Z.
     4: ^Z
     5. This is line five (after the cntl Z).
```

8.08 ***Trap:*** *You cannot install a hard disk in an IBM PC 1.*

The original IBM PC, Model 1, was not designed for installing hard disks. The ROM BIOS that resides on this machine does not support them. So, if you have an old IBM PC and want to install a hard disk, you could be in for a bit of trouble. Fortunately, IBM has taken pity on its early customers and provided a ROM BIOS Upgrade Kit that allows you to replace the ROM storage in the machine with a new version that will support hard disks.

ANSI.SYS, Keyboard Macros, and Color

With the introduction of DOS 2.0, a new capability was added to the operating system: *installable device drivers,* which made it possible to add "pieces" to the operating system to support add-on peripherals like hard disks. One such device driver, ANSI.SYS, is provided on the DOS disk. This file contains the program code that can be added to DOS to make the console and keyboard act like an ANSI standard terminal device. This may sound mysterious, but the results can be very useful indeed.

To make use of ANSI.SYS, you must add a DEVICE=ANSI.SYS command line to a file called CONFIG.SYS on your boot disk. If CONFIG.SYS does not yet exist, you must create it, using the COPY CON: CONFIG.SYS technique, any text editor, or any word processor. This file is like a batch file, instructing the operating system how to configure itself as it boots. If any device drivers are specified in the CONFIG.SYS file, they are loaded into memory along with the rest of DOS.

Many programs have you create a CONFIG.SYS file or provide you with one. If you have one, just add the DEVICE=ANSI.SYS line on the end. If you don't, create one by entering

```
COPY CON: CONFIG.SYS
DEVICE=ANSI.SYS
<ctrl> Z <Enter>
```

You will then need to have ANSI.SYS on your boot disk and do a reboot. After that, you can take advantage of the Tips that use ANSI.SYS.

8.09 Tip: *You can program your function keys with your own commands.*

The popularity of such keyboard macro programs as PROKEY lies in their ability to redefine the function of your keys, allowing you to type one key rather than a whole series to accomplish a task. ANSI.SYS provides you with a "poor man's" keyboard macro capability that comes free with DOS.

The clue to using this capability is the PROMPT command. PROMPT has two features that you will use extensively. The first allows you to issue escape characters, and the second sends a programmable series of characters to the console device, which, in this case, is ANSI.SYS.

The ANSI.SYS driver substitutes any arbitrary series of characters for a character pressed on the keyboard. In essence, ANSI.SYS stands between the keyboard and DOS. It takes what the keyboard gives and uses a table to decide what to pass on to DOS. You may rewrite parts of this table and tell ANSI.SYS to pass on your own particular series of characters to DOS when you press, say, a function key. You tell ANSI.SYS to do this by giving it escape commands using PROMPT. These escape commands are simply commands telling ANSI.SYS to do particular things. All begin with the escape character and the left bracket character ([). This way, it is unlikely that ANSI.SYS would accidentally get a command when arbitrary characters are being typed and displayed on the screen. ESCAPE-[is a very unlikely combination of characters, and DOS makes it *very* difficult to send an escape character to the screen by accident.

To redefine your function keys, you need to know what the "magic" PROMPT commands look like and how to tell ANSI.SYS which key you want to redefine. The keyboard issues scan codes, which are arbitrary numbers corresponding to the key that was pressed. ANSI.SYS wants you to tell it about these scan codes, because it gets them from the keyboard and translates them for DOS. Many keys can be redefined, but we will limit this discussion to the function keys.

The basic PROMPT command is PROMPT $e[0;#;"anything"p, in which $e is the magic that PROMPT translates into the escape character, and the # is the scan code for the function key you want to redefine. The quoted string, "anything", is what you want substituted. Whatever is between the quotes will be sent to DOS when the key is pressed. The p tells ANSI.SYS to leave the cursor where it was found. You can substitute a ;13p for p if your function wants an ENTER at the end of it. The 13 in this prompt string is the ASCII value for the carriage return character.

To put it all together, you need a list of the scan codes for the function keys, which are as follows:

Keyboard Function Key Scan Codes

Key	Normal	Shift	Control	Alt
F1	59	84	94	104
F2	60	85	95	105
F3	61	86	96	106
F4	62	87	97	107
F5	63	88	98	108
F6	64	89	99	109
F7	65	90	100	110
F8	66	91	101	111
F9	67	92	102	112
F10	68	93	103	113

With this information you can create a batch file that will set up your function keys to the commands you want stored in them. You might type

```
COPY CON: REDEF.BAT
PROMPT $e[0;94;"DIR A:";13p
PROMPT $e[0;95;"DIR B:";13p
PROMPT $e[0;96;"DIR C:";13p
PROMPT $e[0;104;"DIR A: ¦ SORT";13p
PROMPT $e[0;105;"DIR B: ¦ SORT";13p
PROMPT $e[0;106;"DIR C: ¦ SORT";13p
PROMPT $e[0;84;"BASICA "p
PROMPT $e[0;85;"EDLIN "p
PROMPT
<ctrl> Z
```

This would create the batch file REDEF.BAT, which you could invoke any time you want your function keys set to these definitions. For example, PROMPT $e[0;94;"DIR A:";13p resets CTRL-F1 to the string "DIR A:" followed by an ENTER. When you press CTRL and F1, DOS receives the characters DIR A: ENTER, and you will see the directory listing displayed. For this all to work, of course, you must have a CONFIG.SYS file with DEVICE=ANSI.SYS in it.

The lines that include a call to the sort program to produce a sorted directory list demonstrate that several things can be done with one keystroke. In general, anything that DOS will allow you to type on a single command line can be stored into a function key.

In the EDLIN PROMPT command you set SHIFT-F2 to generate EDLIN with no following ENTER. This allows you to type SHIFT-F2 and then the filename you wish to edit. It is clear that this capability allows for many interesting uses of your computer system.

8.10 Trap: *Some programs that use function keys will reset any definitions that are programmed with ANSI.SYS.*

Most programs, like BASIC, use the function keys for their own purposes and define the keys as they want. If the function keys (or any other keys for that matter) are defined using the ANSI.SYS commands, they will be overwritten by the program and be lost. The solution to this is to put the offending program in a batch file with REDEF as the last line of the file. This will reset the function keys to the configuration you want them in when the program terminates.

8.11 Tip: *You can set a color display to various color combinations.*

Among the many capabilities of the ANSI.SYS driver is setting a color display to any of its combination of background and foreground colors. For you to use this capability, you must again issue the PROMPT command. You must also install the ANSI.SYS driver using your CONFIG.SYS file as previously explained. The basic command format for setting color attributes is PROMPT $e[#;.....;#m, in which # corresponds to one of the following codes:

Code	Function
0	Reset to normal black and white
1	High intensity on
4	Underscore on (monochrome only)
5	Blink on
7	Inverse video
8	Don't display characters
*0	Black
*1	Red
*2	Green
*3	Yellow
*4	Blue
*5	Magenta
*6	Cyan
*7	White

The * indicates foreground or background. When * is a 3, it is for foreground (e.g., 33 is yellow foreground) and when * is a 4, it is background (e.g., 44 is blue background).

It is possible to combine functions into a single PROMPT command. If you want to set the display to bold yellow letters on a blue background, which is a very appealing combination, you would use the following PROMPT command:

```
PROMPT $e[1;33;44m
```

Multiple attributes in the command are separated by semicolons, and the one trailing "m" indicates that you are setting up a display. You can try several combinations of foreground letters on various backgrounds to find one that pleases you.

Just like our function key definitions, if one of your programs resets the display, then you will have to put it back when the program is done. Batch files

are very handy for such things.

8.12 Tip: *You can control the position of the cursor on the screen with ANSI.SYS.*

Although it is beyond the scope of this book to detail all the ANSI.SYS device driver features, describing what it can do may be helpful. For fuller discussions on the driver, you should consult various computer magazines and DOS technical literature.

ANSI.SYS allows you to control the position of the cursor on the screen and its movement up, down, left, and right. It will report where the cursor is if you ask it with the proper command, and save and restore the cursor under program control.

Getting Started With DEBUG

The Tips in this section require you to use a little-known DOS utility program called DEBUG.COM, which is found on the Supplemental Programs disk that was supplied with your DOS system. DEBUG was designed for computer programmers to find and fix problems in their programs. DEBUG works like an editor, allowing you to change programs that have already been converted into the computer's internal or binary format. DEBUG has commands that permit a programmer to look at the computer's memory, show what information might be there, and put stops or breakpoints in a program so that its progress can be watched closely during execution. Most of the functions of DEBUG are not very useful to someone who does not program, but it has a few capabilities that can be quite handy. In the following Tips, we will use DEBUG either to make small changes to already existing programs or to create small programs that do useful things.

Although DEBUG is a powerful tool, you should use it with caution. Since DEBUG was designed for programmers, it provides you with little or no protection. You must be very sure of everything you do and check things twice. DEBUG modifies the computer's memory and nothing is protected. Items in RAM disk or critical parts of DOS can be changed with DEBUG as easily as intended areas can be changed. Therefore, you should make changes only on backups of your system disks and not have valuable data on your disk when you work with DEBUG.

For a better understanding of all the DEBUG commands, you should read the section in the DOS manual that covers them. For our discussion here, we will do

most of the commands by rote and treat them as though they were magic. You do not need to understand anything at all about computer programming or how the programs are held in the computer's memory. If you can follow a recipe, you can succeed with our Tips.

To use DEBUG, you must first copy the DEBUG.COM program off the Supplemental Programs disk and onto your DOS system disk. You may also want to create a special working version of your DOS disk for experimentation. This way, if you should make a mistake, nothing is harmed.

Once you have DEBUG.COM on a DOS system disk, you can begin to try out some of the Tips. When you type DEBUG at the command prompt, DEBUG comes up with a dash (—) prompt to tell you that it is ready for input. You then can enter the information required and proceed. For convention's sake, since the DEBUG command language is obscure at best, we will indicate characters that DEBUG types on the screen in uppercase and those that you should type in lowercase. We will also show some special keys. The <cr> symbol indicates that you should press ENTER. The <sp> symbol indicates that you should type a space. The XXXX symbol means that DEBUG will type a number that could be of any value.

8.13 ***Tip:*** *You can shorten the time it takes your system to boot.*

When DOS boots, it tests all of the memory it knows about to see if it is okay. If your system has a lot of memory, DOS takes more time to boot. The memory is also tested as it is used, so you may think that you can skip the initial test to save time on each boot. If you hold this view, then read on.

This Tip could really hurt your system disk if you make any mistakes, so you should begin by making a copy of your DOS system disk that you can work with. Once you have made a copy of your system disk and then copied DEBUG.COM from the Supplemental Programs disk onto it, you are ready to proceed.

The trick you will play on DOS is to add a bit of code that is read into memory first when DOS is booted. This code, known as the *boot record,* is on the very first sector of your DOS disk. After it is read into memory, it causes the rest of DOS to be read in and run.

If you are now afraid of modifying something you don't understand, take heart. Very few of us can understand the dark, inner workings of the DOS boot, and even fewer want to. What you will try to achieve here is to save time on something that you do frequently.

You are going to use DEBUG to change the way the DOS boot record finds out how much memory is in the system. Normally, it just looks at your system

switches and uses what it finds. But you will make it a little smarter and let it look directly. Once the boot record learns how much memory you have, it saves the memory inside the system. Thus, DOS and programs running under DOS can find out easily when they need to know. Once the boot record is changed to disregard the switches, you can set your switches, say, to 64K, and shorten the memory test.

This Tip shows you how to make DEBUG less formidable. You will take advantage of the DOS redirection function discussed in Chapter 6 to create a file of input for DEBUG. By creating a disk file with your instructions to DEBUG in it, you have a chance to review the information before DEBUG acts on it. This will make you feel comfortable about giving a fairly complex series of commands to a program you don't know much about.

So, starting with your working disk, you bring up the EDLIN editor and create a text file. (You can use any other editor if it can create ordinary ASCII files.) The file to type in with EDLIN follows:

```
A>edlin patch
New File
*1i
     1:*l cs:100 0 0 1
     2:*a 100
     3:*jmp 282
     4:*
     5:*a 282
     6:*xor ax,ax
     7:*out a0,al
     8:*mov bx,1000
     9:*mov es,bx
     10:*xor di,di
     11:*mov cx,8000
     12:*repz
     13:*stosw
     14:*es:
     15:*cmp [0000],ax
     16:*jnz 2a3
     17:*add bx,1000
     18:*cmp bx,a000
     19:*jnz 289
     20:*mov cl,06
     21:*shr bx,cl
     22:*mov ds,ax
     23:*mov [413],bx
     24:*mov al,80
     25:*out a0,al
     26:*jmp 12e
     27:*
     28:*w cs:100 0 0 1
     29:*q
     30:*^Z
*e

A>
```

Once you have created the patch file and checked it very carefully, you can proceed to have DEBUG execute your commands. This is done by typing the command DEBUG < PATCH:

```
A>debug < patch
-1 cs:100 0 0 1
-a 100
0E21:0100 jmp 282
0E21:0103
-a 282
0E21:0282 xor ax,ax
0E21:0284 out a0, a1
0E21:0286 mov bx,1000
0E21:0289 mov es,bx
0E21:028B xor di,di
0E21:028D mov cx,8000
0E21:0290 repz
0E21:0291 stows
0E21:0292 es:
0E21:0293 cmp [0000],ax
0E21:0297 jnz 2a3
0E21:0299 add bx,1000
0E21:029D cmp bx,a000
0E21:02A1 jnz 289
0E21:02A3 mov c1,06
0E21:02A5 shr bx,c1
0E21:02A7 mov ds,ax
0E21:02A9 mov [413],bx
0E21:02AD mov a1,80
0E21:02AF out a0,a1
0E21:02B1 jmp 12e
0E21:02B4
-w cs:100 0 0 1
-q

A>
```

You can take this approach whenever you need to provide DEBUG with a large amount of input.

Once your DOS boot has been modified, you can change the switches in your computer to indicate that the system has only 64K of memory. Then reboot the system with your new system disk and note how fast the boot is.

Some caution is advised, however. Not every program combination will work with this new boot. Some system utilities, like RAM disks, and programs that modify the amount of memory that DOS thinks it has, may not work. Try out your favorites before you place your new system disk into general use.

8.14 Tip: *You can change the border color on your color display.*

Earlier in this chapter, we discussed using ANSI.SYS and the PROMPT command to set the foreground and background colors on your color display. This technique works fine, but it does nothing for the border around the display. The border area will remain black unless you tell the color display board to paint it in another color. Since ANSI.SYS doesn't do borders, you will need another approach. It is possible to create a small program using DEBUG that will set the border to whatever color you choose. Such a program takes the color value you want (in our example we chose blue) and outputs it to the proper port on the color board. You don't have to understand this, thank goodness, to get the program to work.

Set up your DOS disk with DEBUG as we have described and then type the following:

```
C:\>debug
-n frame.com
-e100
149F:0100  FF.b0  BC.03  10.ba  00.d9  10.03  4C.ee  CD.cd  21.20
-rcx
CX 0000
:8
-w
Writing 0008 bytes
-q
C:\>
```

The program FRAME.COM will now exist on your disk and can be run to paint the border blue. If you don't like blue, you can change the first XX.03 in the example to XX.0*n* in which *n* is the color code discussed in the PROMPT example in Tip 8.11.

This Tip demonstrates the power of DEBUG to create short assembly language programs that allow you to take full advantage of your computer system.

8.15 Tip: *You can make menus with batch files.*

In Chapter 5 we investigated menus using batch files. One approach that might work is using the IF ERRORLEVEL construct supported in the batch file language. ERRORLEVEL is a DOS variable that lets programs report back to a calling batch file. It is often used to return an error code value, but any integer number is allowable. Unfortunately, DOS provides no easy method for setting the ERRORLEVEL based on something the user types. This Tip will help you correct that omission.

Using DEBUG again, you will create a small program that reads the keyboard for a single character and puts that character into the ERRORLEVEL location in DOS. Once ERRORLEVEL is set to the character value, you can check it in a batch file. Begin by creating a little program called GETKEY.COM with DEBUG:

```
C:\>debug getkey.com
File not found
-a100
149F:0100 mov ah,00
149F:0102 int 16
149F:0104 mov ah,4c
149F:0106 int 21
149F:0108
-rcx
CX 0000
:8
-w
Writing 0008 bytes
-q

C:\>
```

Once you have GETKEY.COM on your disk, you can write a batch file to use it. In the example presented here, we paint a menu with echo commands, execute GETKEY to pick up whatever you type, and compare the value to the range of legal character values. In the case of numbers, the values will range from 49 for a 1, to 57 for a 9. These are the ASCII values associated with the numbers. In constructing such batch file menus, it is important to get the GOTOs and labels correct, so note them in particular. Your batch file then is

```
ECHO OFF
REM BEGIN BY CLEARING SCREEN AND DISPLAYING MENU
CLS
REM START IS WHERE WE RETURN TO REDISPLAY THE MENU
:START
REM DISPLAY THE MENU CHOICES
ECHO            1  FIRST CHOICE
ECHO            2  SECOND CHOICE
ECHO            3  THIRD CHOICE
ECHO            4  EXIT
REM HERE IS WHERE WE COME IF THE CHOICE WAS WRONG
:CONTINUE
ECHO    TYPE THE NUMBER OF THE CHOICE YOU WANT.
REM GET THE ASCII VALUE IN ERRORLEVEL USING GETKEY
GETKEY
REM THE IF ERRORLEVEL TESTS FOR GREATER THAN
REM SO WE HAVE TO ELIMINATE VALUES GREATER THAN 4
REM AND TEST BACKWARD.
IF ERRORLEVEL 53 GOTO CONTINUE
REM NOW TRY FOR LESS THAN 5
```

```
IF ERRORLEVEL 52 GOTO EXIT
REM NOW LESS THAN 4
IF ERRORLEVEL 51 GOTO THIRD
REM NOW LESS THAN 3
IF ERRORLEVEL 50 GOTO SECOND
REM NOW LESS THAN 2
IF ERRORLEVEL 49 GOTO FIRST
REM ALL THE REST ARE IN ERROR
GOTO CONTINUE
REM EXECUTE CHOICE 1 PROGRAM
:FIRST
ECHO DO FIRST HERE
REM AND REDISPLAY THE MENU
GOTO START
REM EXECUTE CHOICE 2 PROGRAM
:SECOND
ECHO DO SECOND HERE
REM AND REDISPLAY THE MENU
GOTO START
REM EXECUTE CHOICE 3 PROGRAM
:THIRD
ECHO DO THIRD HERE
REM AND REDISPLAY THE MENU
GOTO START
REM THE CHOICE WAS EXIT
:EXIT
REM CLEAR THE SCREEN AND RETURN TO DOS
CLS
```

The concept behind this Tip is that you can pick up a character from the keyboard and test it inside a batch file. With the test results, you can execute one or more programs that correspond to the menu lines. You can modify this batch file to execute any series of necessary program choices and extend it to more than four choices. It would be ideal for providing an easy-to-use interface to a few applications that are regularly used by people who are not computer experts.

The application of batch file menus is quite broad. They can help new users in accessing the computer's power, provide a selected series of options for users, and generally make the computer less complex for people who are not experts.

DOS 3.X

IBM introduced DOS 3.0 primarily to support the IBM PC AT's hardware—particularly the 1.2-MB floppy disk drive and the 20-MB hard disk. DOS 3.1 was introduced primarily to support the IBM PC Network. In addition to these important areas of support, IBM added many new features, commands, and enhancements to existing commands. The Tips and Traps in this section will alert 2.X and first-time DOS users to these changes.

8.16 Tip: *Save your old versions of DOS if you convert to DOS 3.X.*

You may prefer older versions of some of the DOS utilities. The SORT command, for instance, functions differently under DOS 3.X (see Trap 8.22). There is no guarantee that older versions of the utilities will work under a new version of DOS, but in some cases you may find that an older version works fine.

8.17 Tip: *Get the DOS 3.X Technical Reference Manual.*

There are so many references to this technical manual in the DOS 3.X manual that you will feel uneasy if you don't have it. The Technical Reference Manual explains some features more completely than the DOS 3.X manual does.

8.18 Trap: *DOS 3.X takes up 20K more RAM than DOS 2.1.*

Applications that took up most of the available RAM under DOS 2.X may run out of memory under DOS 3.X, since DOS itself uses up about 20K more RAM before the application even gets started. So be alert to the fact that you may need to add more RAM to your system before switching from 2.X to 3.X.

8.19 Tip: *DOS 3.X has a number of new commands and sub-commands.*

See your DOS 3.X manual for detailed descriptions. Here is a summary:

ATTRIB	Mark a file for read-only status.
BASIC 3.X	SHELL statement enables a BASIC program to execute DOS commands and run DOS programs.
CONFIG.SYS	
COUNTRY	Format for date, time, currency symbol.
FCBS	Specify number of file control blocks that can be open at one time.

LASTDRIVE	Set maximum number of drives (A through Z).
JOIN	Connect a drive to a directory on another drive.
LABEL	Change a disk volume label.
SUBST	Substitute a one-letter drive name for a path.
VDISK	Create a RAM disk. (This capability was provided under DOS 2.X, but the VDISK program itself was not provided on disk.)

8.20 Tip: *DOS 3.X contains several enhancements to 2.X commands and subcommands. See the DOS 3.X manual for details.*

PRINT	You can specify how many files you want to print and some other fine-tuning parameters.
TREE	This has been improved, but for a better version, see Chapter 6, "Redirection, Piping, and Filters."

8.21 Trap: *DOS 3.X-enhanced hard disk file management works only on an AT.*

DOS 2.X has file management overhead of approximately 2K for each hard disk file. DOS 3.X has only 1K overhead if you have AT hardware; with XT hardware, DOS 3.X will still have 2K overhead per file.

8.22 Trap: *The DOS 3.X SORT filter sorts in a different sequence than does DOS 2.X.*

DOS 2.X sorts in pure binary sequence. Numbers are followed by uppercase letters, which are followed by lowercase letters. DOS 3.X, however, ignores uppercase and lowercase letters. If you currently rely on the DOS 2.X sequence, you may want to save the 2.X SORT.EXE program with a new name, like SORT2._EXE.

Original File	DOS 2.X Sorted	DOS 3.X Sorted
A	1	1
b	2	2
1	A	A
2	B	a
a	a	b
B	b	B

8.23 Tip: *DOS 3.1 accommodates programs that do not work with paths.*

The SUBST, or substitute, command allows you to specify a one-letter drive name in the place of another drive or a complete path name. To access a letter text file C:\WP\LETTERS\INVOICE.TXT by specifying just D:INVOICE.TXT, you would enter the following command statement:

```
SUBST D: C:\WP\LETTERS
```

Note: The DOS 3.1 default provides drives A through E. If you want to use drives F through Z, you need to specify your last drive letter in the CONFIG.SYS subcommand LASTDRIVE.

8.24 Tip: *You can change the volume label on a disk.*

DOS 3.X allows you to change the volume label on a floppy, hard, or RAM disk with the LABEL command. Under DOS 2.X, you could supply a label only while formatting a disk and could not change it later without reformatting.

8.25 Tip: *You do not need to reformat your hard disk to change from DOS 2.X to DOS 3.X.*

Boot 3.X on your floppy disk; then use the SYS command to transfer DOS 3.X to the hard disk. COPY the COMMAND.COM and other DOS commands to the hard disk.

8.26 Trap: *Dot comments in a batch file are not accepted by DOS 3.X.*

You could use a period instead of REM in a batch file in DOS 1.X and 2.X. This resulted in somewhat prettier remarks when the batch file displayed them on the screen. DOS 3.X, however, returns the error message "Bad command or filename."

8.27 Trap: *Some batch files that worked with DOS 2.X may not work with DOS 3.X if they use redirection of input from response files.*

Some DOS 3.X commands, like FORMAT, require the user to press ENTER to continue instead of "any key," which DOS-2 requests. Chapters 4 and 6 contain several Tips that show how input redirection from a response file can be used to provide automatic responses to DOS commands. To make such response files work under DOS 3.X, you may need to replace whatever key you have placed in the response file with an ENTER (i.e., a carriage return character).

8.28 Trap: *You may not be able to format a low-capacity, double-sided diskette reliably on an AT high-capacity, floppy disk drive under DOS 3.X.*

The standard floppy disk drive on the IBM AT is a high-capacity drive that normally expects to read from and write to a special high-capacity diskette (holding 1.2 megabytes of data) instead of the "low-capacity," double-sided diskette (holding 360K bytes of data) that is now standard on the IBM PC and XT models.

The AT high-capacity drive will read from and write to low-capacity diskettes that were formatted in a low-capacity drive on a PC, XT, or AT. This is important because it allows you to exchange disks only between an AT with a high-capacity drive and a PC or XT. The DOS 3.X FORMAT command option 4 theoretically allows you to format a low-capacity diskette on the AT's high-capacity drive. However, as the DOS 3.0 manual states, a disk formatted in this way on the high-capacity drive cannot necessarily be used reliably in a low-capacity drive. If you want to exchange diskettes between an AT and an XT or PC, format these diskettes on the XT or PC.

A

Behind the Scenes

Chapters 1 and 2 introduced the DOS commands you need to know to start using your computer. We covered how to boot the system, copy files and disks, create and list the contents of subdirectories, install and use applications software, and back up your files. As you use the DOS commands to perform these tasks, DOS itself is doing a great deal behind the scenes that is invisible to you. It is useful to know some of the details of these invisible processes, primarily because knowledge of them will help you handle unexpected situations. So, this appendix takes a "behind the scenes" look at the operating system and how it coordinates the hardware and software components. We will present this information in nontechnical terms to give you a practical working idea of what's going on inside your machine.

DOS: Your Computer's Autonomic Nervous System

The importance of the operating system can be illustrated by comparing its functions to those of your body's central nervous system. Without a nervous system, you would be doomed to a senseless, motionless, thoughtless, vegetative existence. Nerve cells transmit messages that move muscles, give meaning to this printed

page, or regulate hundreds of such automatic activities as your heartbeat, which is essential to life. Sensory nerves carry impulses to the central nervous system, and motor nerves carry impulses away from it to effect action. For example, sensory nerves detect that your finger is resting on a hot stove; motor nerves direct muscles to retract your finger, thus removing it from danger. The nervous system is also responsible for monitoring your external environment. Every movement of your body involves a coordinated effort between your brain and peripheral nervous system, with countless messages being sent back and forth and interpreted again and again. All of these detailed interactions occur automatically without your conscious awareness. The autonomic nervous system takes care of these details, thereby making it possible for your conscious mind to do its job every day without being overwhelmed.

The message is quite clear: You cannot function without your central nervous system. Similarly, you also cannot function without your computer's operating system, which also handles numerous essential, boring details. Without DOS, you would have to be responsible for performing these tasks yourself.

Similarly, DOS handles the low-level details that let word processing, spreadsheet analysis, and database management programs help you with your business tasks. DOS is the link between you and your computer hardware. To understand this link, we must first take a quick look at the various components of this hardware. As with a component stereo system, various configurations are possible.

Basic Personal Computer Hardware Components

The mechanical and electronic parts of your computer are called hardware.

The main component of your computer is the *system unit.* It houses the following parts:

- *System Board:* This is the principal board; it lies flat on the bottom of the system unit and holds chips and expansion boards.
- A *chip* contains many miniaturized electronic circuits. There are many different types of chips, each performing a specific function. The computer's brain is the *microprocessor chip,* also known as the *central processing unit (CPU).* It contains the logic that interprets programs and follows instructions. The development of microprocessors made personal computers possible. There are many microprocessor chips, made by different manufacturers. IBM PCs use Intel chips in the 8086 family. Other chips, which contain the computer's "memory," are known as RAM and ROM chips.

- *RAM (random access memory)* is memory that can be read from or written to. Programs and data are stored in RAM. Unless saved to disk, the contents of most types of RAM are lost when the computer is turned off.

- *ROM (read-only memory)* contains permanent data or program code, and cannot easily be accessed or altered by the user. A special part of ROM called ROM BIOS contains instructions that allow the operating system to communicate with the computer's components and devices. You will learn more about ROM BIOS later in this appendix.

- *Expansion boards* make adding memory and optional equipment to your system possible. Printers, modems, and additional disk drive units are some of the many products available to you from various vendors. The system board is proprietary to IBM, while expansion boards can be designed by other companies, if IBM specifications are followed.

- *Disk drives:* One of the most important components of your computer, the disk drive makes storing and retrieving information possible. A floppy disk drive uses floppy diskettes as the storage medium, with a maximum storage capacity of 300,000 to 1,000,000 *bytes* (characters). Hard disks (fixed disks) are also available, offering 30 times the capacity of a floppy diskette. All programs, including PC-DOS, are distributed on floppy diskettes; some can be copied onto a hard disk.

- *Miscellaneous parts:* The system unit also contains a *speaker,* which you will occasionally hear beeping at you, usually when DOS or some applications software decides you have committed an error you need to be warned about. The power supply and miscellaneous wires and connectors account for the remainder of the internal system parts.

- The *keyboard* is the component with which you are primarily concerned. Referred to as an input device, the keyboard makes it possible for you to communicate with your computer. It translates the particular key you press into an electronic signal. The operating system detects these signals and further translates them into a different code that the applications program can use.

  One of the functions of the operating system is to translate your keystrokes in much the same way an interpreter translates a foreign language. The IBM PC keyboard is divided into three sections: special function keys, standard typewriter keys, and numeric keys arranged in "10-key" order. Function keys are defined by individual software programs.

- The *display screen* rounds out the basic system. It is the computer's visual output, which lets you monitor some of what is going on internally. Two

types of displays are available from IBM: monochrome and color. The monochrome display is popular because of its high resolution, with easy-to-read green or amber characters. It will not, however, display graphics without a special adapter board. Color monitors are becoming more popular as software relies more heavily on graphic representation of data.

- Although a *printer* is optional, few computers are used without one. The printer converts what you see on the display screen into printed images. While hundreds of different printers are available, there are basically three different categories: dot matrix, daisywheel, and laser. Dot matrix printers generate characters from dots and most have graphics capability. Daisywheel printers are designed like a typewriter, with shaped character images (letters, numbers, symbols) attached to the ends of spokes on a wheel. They produce letter-quality print, as if produced on a typewriter, but cannot ordinarily print graphics. The laser printer represents the latest technology and employs photocopying techniques. The most expensive laser printers can handle high-volume operations and offer both letter-quality print and graphics images. They produce an entire page at a time.

- Another common hardware addition is a *modem*. With this device you can communicate with other computers through standard telephone lines. Modems are classified by the rate at which they transmit characters: 30 characters per second (300 baud), 120 characters per second (1200 baud), and more recently, 240 characters per second (2400 baud). The rate of transmission is usually known as the *baud rate*.

How DOS Handles The Dirty Details

Hardware makes it possible, but software makes it happen. No matter how powerful the hardware, a computer can't do anything without software. As we suggested at the beginning of this appendix, your computer's operating system is like your autonomic nervous system. In this section, we take this analogy a bit further.

From one standpoint, your body without the nervous system is just a collection of unintelligent hardware, like an assistant who can't do anything right unless you provide explicit instructions. Your assistant may be loyal, conscientious, and hardworking, but incapable of independent thought or initiative. Your computer hardware is a simpleton just like your assistant. It needs software to provide explicit instructions on what to do and when.

An operating system makes it possible for you to use a computer without

having to be concerned with the multitude of details that are involved in its operation. To get a better idea of all the dirty details the operating system handles for you, let's examine what it does when you use the COPY command to copy a file from one diskette to another. This is one of the DOS procedures you used in Chapter 2. DOS asks itself and then answers the following questions every time you issue the COPY command:

- Does the source diskette have a single- or double-sided format?
- Does the target diskette have a single- or double-sided format?
- Does a file with the specified name exist on the source diskette?
- Does a file by that same name already exist on the target diskette?
- Does a specified subdirectory path exist?
- Does the target diskette have enough room for the file?
- Is the user trying to copy the file to itself (not allowed)?
- Do the source and target drives exist, or is a "fake" drive designation being used?
- Is the file allocation table for the source diskette loaded into memory?
- Is the file allocation table for the target diskette loaded into memory?
- Is the file size reported in the directory equal to the allocated file size?
- Is there enough room in memory to buffer the file copy?
- Is the diskette drive operating?
- Is the read/write head located on the right track?
- How many sectors of data will be read from or written to on this track?
- Is the disk drive ready to accept a command?
- Is it necessary to display an error message?
- Was the copy successful?

This is actually just a partial list. If you had to concern yourself with this list of technical details every time you wanted to copy a file, you probably wouldn't bother to use a computer.

In what follows, we will take a quick tour of PC-DOS, always with this question in mind: "What does this mean to you, the user?" PC-DOS's modular design has essentially six different parts. When you boot your system, each of these parts in turn comes into play and then "hands the ball off" to the next part.

ROM BIOS

Primary Purpose

ROM BIOS is the software "burned" into ROM that directly communicates with and controls the PC's peripherals. A computer cannot use a peripheral device without a set of instructions about how to communicate with this device and control its actions. The ROM BIOS supplies this information, including fundamental routines that control the keyboard, display screen, disk drive, and other peripherals. Without ROM BIOS, your computer is unable to coordinate the efforts of its own components.

BIOS is short for Basic Input/Output System, and ROM stands for read-only memory. The ROM BIOS is resident in the read-only memory that comes with your hardware.

Description

Since the ROM BIOS resides on a ROM chip, it can only be changed by replacing that chip, thus modifying the hardware. When program code is part of ROM, it is referred to as *firmware*. It is still software, but can't be modified like software on a disk.

ROM BIOS is technically not part of DOS. It serves as the basic interface with the hardware for any operating system. This includes not only PC-DOS but also CP/M-86 and any other operating system that runs on the IBM PC. (While DOS is the most popular operating system for the IBM PC, it isn't the only one available.)

Responsibilities

1. When the power switch is turned on, ROM BIOS initiates the first program, a self-test routine of the hardware and the memory. If your system is configured with maximum RAM memory, this process will take longer to complete.

2. The next step is to load the operating system. This program is called the "start-up boot loader" because the operating system needs to pull itself up by its bootstraps to get started. (The start-up boot loader is a tiny program that loads the boot record off the disk. The boot record contains a larger loader program that in turn loads the rest of the operating system off the disk and into memory.)

3. ROM BIOS then checks to see if a disk drive is installed, and if so, reads the boot record from the diskette. Control is then passed to the BOOT RECORD program, so that the remainder of the operating system can be read into memory.

Visible to You

During the self-test routine, the cursor is flashing in the upper-left corner of your screen. When the boot record is read, the red light on your disk drive will glow.

What This Means to You

With a non-hard disk system, if ROM BIOS doesn't find a diskette in the drive, then it can't read the boot record, so it defaults to the cassette BASIC system. (This is the BASIC programming language, but without the ability to access the disk drive system.)

One of the primary functions of the self-test routine that ROM BIOS initiates is to check the RAM memory in your system. Up to 256K of RAM memory can be installed on the system board. To extend beyond this limit, a memory expansion board is needed. If ROM BIOS detects a problem with the system board memory, it displays this error message on your screen:

```
PARITY CHECK 1
```

Should there be a problem with the memory installed on your memory expansion board, you will see this message:

```
PARITY CHECK 2
```

There have been two versions of the IBM PC. The first, known as PC1, has a maximum user-addressable memory of 544K. The second version, PC2, allows software to address up to 640K directly. When IBM made this change, the ROM BIOS had to be replaced. (Remember, ROM BIOS is firmware and, by definition, not flexible, since its programs have been "burned" into a chip permanently.)

If you own a PC1 version of the IBM PC, you can take your system unit to an authorized dealer and have the ROM chip replaced, so that you can reach the current maximum addressable memory level of 640K.

Boot Record

Primary Purpose

The boot record is actually a small program. Its primary task is to decide whether the diskette in the drive is a "system" diskette (DOS). A system diskette includes two files, IBMBIO and IBMDOS, which contain the information for starting up the system and communicating with the peripheral devices. If the boot record finds these two files, they are loaded into memory, which starts DOS.

Description

The boot record is located at track 0, sector 1, of a system diskette containing DOS and is the first program read into memory. It is a very small program, taking up just one sector (approximately 512 bytes). This relatively stable part of the operating system needs to be modified only if the size or location of the system files changes. (Such was the case when IBM changed from single-sided to double-sided diskette drives.)

Responsibilities

The boot record has but one responsibility—to load the IBMBIO and IBMDOS files. This job has been simplified by the fact that these two files are always placed at a predefined location on the system diskette.

Visible to You

The red light on your disk drive will glow while the boot record is reading the system files.

What This Means to You

If the boot record encounters either a non-DOS diskette or a DOS diskette that is physically damaged, the following error message is displayed on the screen:

```
Non-System disk or disk error
Replace and strike any key when ready
```

This message tells you that the boot record is unable to read the system files or can't find them in the expected location on the disk.

Now you can see why a system-formatted diskette differs from a diskette that does not include system files. It isn't possible to convert a nonsystem diskette into a system diskette unless the former has been formatted with the B or S option of the FORMAT command, which leaves space so that the system files can be added later. This is because files are saved in the same sectors that system files are placed in so that the boot record can find them.

IBMBIO

Primary Purpose

The program named IBMBIO extends the functions of the ROM BIOS to deal with all the tedious details of operating the peripheral devices, such as the keyboard, printer, and disk drives.

Description

IBMBIO is the first DOS program to be read and loaded into memory when DOS is started up. It is one of two "hidden" files on the DOS diskette. You cannot access a "hidden" file.

Responsibilities

IBMBIO is a flexible software program, not permanent firmware like the ROM BIOS. Any operating system can access the ROM BIOS, but it is the IBMBIO program that is tailored to the specific needs of a particular operating system.

ROM BIOS is carefully tested because, as firmware, it is essentially unchangeable once it is embedded in ROM. However, IBMBIO, being software distributed on a disk, can be easily changed. It is therefore possible even for a new version of IBMBIO to fix certain errors discovered in the ROM BIOS.

The job of IBMBIO is to handle new peripheral devices that become available. Support programming can be added to the IBMBIO program without having to replace ROM-BIOS chips.

In the case of DOS-2 and subsequent releases, the very first task performed by IBMBIO is to check for a "configuration" file (CONFIG.SYS) on the diskette. If one is found, it is read into memory. A configuration file can contain pointers to

device drivers, which are special programs used to control such peripheral devices as hard disk drives. When you buy a peripheral device, you generally receive a diskette with the device driver program and instructions on how to construct a configuration file that points to it.

Visible to You

There is nothing visible on your screen at this time.

What This Means to You

IBMBIO, together with ROM BIOS, makes it possible for you to use your computer and available peripheral devices without having to be concerned with the extremely technical aspects of operating input/output devices.

With the earliest version of DOS (DOS 1.0), a user had great difficulty adding new peripheral devices. The ability to add components to a system is very important if you wish to customize a computer to meet a specialized need. DOS-2 and subsequent versions make this more practical by looking for and reading any configuration file that you may add.

The flexibility of DOS-2 extends its life span and makes it possible for it to respond to future trends in the computing industry. If this were not the case, then you would have to discard all of the experience you had acquired and start over with a new operating system each time technology took another step forward.

IBMDOS

Primary Purpose

Provides the core DOS services; creates and manages disk files.

While the ROM BIOS and IBMBIO make the physical connection between your system and its peripherals, IBMDOS controls the information that is transferred between them. It gives directions to IBMBIO, which communicates directly with the device with the help of the ROM BIOS at the lowest level. IBMDOS is often regarded as the "logical" component of the input/output system, handling the flow of information between devices.

Description

This is the second DOS file to be read and loaded into memory. IBMDOS is the other "hidden" file on the DOS diskette. This file contains the major part of the operating system, and two different filing systems.

Responsibilities

The IBMDOS program uses two different methods for communicating with peripherals: one for disk drives and another for all other peripherals. Every peripheral communicates to the system unit either one character at a time or in groups of characters called *blocks*. The keyboard, printer, and screen display are character-oriented devices, while a disk drive is block-oriented. The two different types of devices require separate management routines to handle them.

When DOS communicates with a character-oriented device, IBMDOS performs the following tasks:

- Retrieves a character from a device
- Transmits a character to a device
- Determines whether a device is ready
- Retrieves a string of characters from the keyboard
- Transmits a string of characters to the display screen.

In the case of block-oriented devices, IBMDOS employs the disk filing system to do any or all of the following:

- Create a file
- Locate a file
- Save information to file
- Read information in the file
- Update a file
- Close a file
- Identify a file by name

- Update the file directory
- Erase a file
- Report the amount of available disk space
- Keep track of which disk drives are being used
- Load and run applications programs
- Maintain the hierarchical directory system.

These filing system routines are the "control center" of the operating system.

Data is stored in files on the diskette, and IBMDOS is responsible for keeping track of these files. Every file must be given a name when it is saved, and IBM-DOS creates and maintains a file directory on each disk that includes filenames, sizes, and exact locations on the disk. A file directory serves in a capacity similar to that of a table of contents for a book.

Before a file can be saved on the diskette, IBMDOS checks the file allocation table (FAT). This table occupies the two sectors that follow the boot record on all diskettes. The file allocation table is used to keep track of disk storage, indicating which sectors are being used and which are not.

Whenever a new file is added to a diskette, IBMDOS performs the following tasks:

1. Checks the file directory to make sure that no file with the same name is already on the diskette.
2. Checks the file directory to make sure there is room for the new file.
3. Checks the file allocation table for an available location for the new file.
4. Updates the file directory with information about the new file.

Visible to You

Nothing is visible on your screen at this time.

What This Means to You

IBMDOS is responsible for numerous tasks that make it possible for you to save and retrieve information on diskettes. Trying to operate a computer without this capability would be like trying to run an office without a filing system.

The IBMDOS program interprets your keystrokes so that you can communicate with your computer in a syntax close to that of spoken English. If this translation didn't take place, you would be forced to speak the computer's binary language of 1's and 0's.

COMMAND.COM

Primary Purpose

The COMMAND.COM program, commonly referred to as the Command Processor, reads each DOS command you enter from the keyboard and executes the subprogram that performs that particular task.

Description

COMMAND.COM is a large program that if loaded entirely would take up a large amount of RAM memory. For this reason, COMMAND.COM is segmented into three parts. The first part is loaded into memory and becomes integrated with the resident portion of DOS. The second part is activated when DOS is loaded; it checks for and executes an AUTOEXEC.BAT batch file and then is discarded. The third part is "semiresident" in memory. It contains the program that interprets your commands, the programs to execute internal commands, and a routine to load and execute external commands. This part of COMMAND.COM is very complex and takes up a lot of memory when loaded. DOS allows other programs to overwrite this portion of COMMAND.COM when they need memory. When this happens, the next time you exit to DOS, the resident portion detects that this transient portion of COMMAND.COM has been overwritten and reloads it from the disk.

The COMMAND.COM file could have been combined with IBMBIO and IBMDOS, but IBM elected to keep them separate so that custom versions could easily be created and integrated with DOS. This separation makes modifying existing DOS commands or creating new ones possible. Each new version of PC-DOS has taken advantage of this capability, and as a result, continues to respond to our changing needs.

Responsibilities

Responding to our commands, COMMAND.COM finds a program file, loads it into memory, and then passes control to the program.

COMMAND.COM maintains a table of command names to recognize the commands you enter on the keyboard.

The command interpreter portion of COMMAND.COM looks for and executes batch files.

The COMMAND.COM file contains programs for DOS utilities that are used frequently. Because they are part of the command processor, these functions are available to you whenever DOS is loaded into memory. The following is a list of these DOS programs, known as internal commands:

DIR displays information about the files on a disk or directory.

COPY copies one or more files from one disk or directory to another.

DATE sets or displays the system date.

TIME sets or displays the system time.

TYPE displays the contents of a file (text files only).

RENAME changes the name of a file.

ERASE eliminates disk files.

When any of these commands are entered, they are immediately executable, because they are loaded as part of COMMAND.COM.

Visible to You

After COMMAND.COM is successfully loaded into memory, you will see the system prompt on the screen, A> or C>, indicating that the system is waiting for you to issue a command.

What This Means to You

Without COMMAND.COM, you would not have access to DOS functions or applications programs. This limitation would render your computer useless for running PC-DOS and its family of applications programs.

If the resident portion of COMMAND.COM has been overwritten, the following error message will be displayed:

```
Invalid COMMAND.COM
Insert COMMAND.COM disk in drive A
and strike any key when ready
```

It is possible to use COMMAND.COM to restart your system. If you enter

```
COMMAND
```

the COMMAND.COM program will be reloaded, and your system will be restarted. This type of rebooting does not include loading the IBMBIO and IBMDOS files. The command interpreter is reloaded, thus making it possible to reexecute an AUTOEXEC batch file.

Some applications programs include a custom COMMMAND.COM program that replaces the standard version. Software developers can customize the internal commands or modify the command interpreter to address a specific need.

A disk drive cannot successfully read a disk if the door is open. The COMMAND.COM program will report the problem by displaying the following message:

```
Disk error reading drive A
Retry, Ignore, Abort
```

External Commands

The DOS commands that are not part of the COMMAND.COM file are called external commands and reside on the diskette. If all of the DOS commands were resident in memory, there might not be enough available RAM to load an applications program. External commands represent additional utilities that are used in the same way as internal commands, except that they are not part of resident DOS. When you issue one of these commands, the command interpreter follows the same procedure for loading internal commands, except that it needs to access the disk to locate the program file.

If you enter one of these external commands, and the command interpreter cannot find the file, DOS will display this error message:

```
Bad command or filename
```

How DOS Stores And Manages Files

Now that you understand how the operating system works and are aware of the responsibilities of each component, we will turn our attention to the first word in

"Disk Operating System." Your computer's random access memory is *volatile:* Your work is lost when you turn off the system. The only way to save your work permanently is to store information in a file on a disk.

Let's look at a floppy diskette. DOS handles a diskette by dividing it into smaller units. This process is initiated when you issue the FORMAT command. First, concentric circles called *tracks* are created. Forty tracks are marked off. These tracks are then divided into sectors. DOS-1 divides the tracks into eight sectors, and DOS-2 normally into nine sectors. Organizing diskettes by tracks and sectors creates locations on the diskette that are assigned addresses. DOS uses these addresses to reference a file when saving or retrieving it.

Under DOS-2, disk drives are capable of storing 360K (360,000) bytes of information on a floppy diskette. If we examine the formatting procedure numerically, you will understand this equation:

40 tracks * 9 sectors * 2 sides = 720 sectors

Two sides are involved, since DOS stores information on both sides of the diskette. A total of 512 bytes can be stored in a single sector:

720 sectors * 512 bytes per sector = 368,640 bytes

368,640 bytes / 1,024 bytes per K (1,000) = 360K

After a diskette is divided into sectors, DOS performs three important tasks:

- The boot record is placed in the first sector of the first track.
- Two copies of the file allocation table are placed on the diskette.
- The directory is created. Here DOS stores the name of each disk file, the date and time a file was created or last modified, the file attributes, the starting cluster entry in the file allocation table, and the size of the file.

The file allocation table (FAT) is very important to DOS, so two copies of it are found on every diskette. The FAT tells DOS where to find your file on the diskette. (If DOS can't find your file, your work is lost.) Before saving a file, DOS checks the FAT to determine which sectors are available.

Using disk storage is analogous to using a filing cabinet. Information is organized into files, and each file contains data related to a specific application. This might be text such as a document or letter, a spreadsheet analysis, or accounting records. When you need to access this information, you issue the appropriate command to retrieve the file, open it, read or write the information, and then close the file.

Electronic filing systems, like paper filing systems, can be organized according to different conceptual plans. While alphabetical ordering is the most typical plan, files can also be organized by subject or time. The important point is that the person in control of the filing system decides how the records should be arranged. Without a plan, both electronic and paper filing systems become totally disorganized and inefficient and can even disappear.

DOS handles most of the tasks involved in creating and retrieving file information. Your primary responsibility is to name your files. A filename has two parts: an eight-character name that identifies the contents of the file and an optional three-character extension that identifies the file type. Most applications programs automatically assign a unique extension, like .WKS, for all worksheet files created with the LOTUS 1-2-3 program. This extension is an abbreviation for worksheet. A typical filename is BUDGET85.WKS.

The following rules are always in effect when you name your files:

- Filename of one to eight characters
- Optional extension of one to three characters
- A period between the filename and the extension
- All alphabetical characters can be used (A to Z)
- The numbers 0 to 9 can be used
- Spaces and the following characters are *not* allowed:
  ^ + = / [ ] " : ; , ? *
- Filenames must be unique.

Trademarks

Bernouli Box®	Iomega Corporation
dBASE III™	Ashton-Tate
Hercules™	Hercules Computer Technology
IBM®	International Business Machines, Inc.
IOMEGA®	Iomega Corporation
Lotus®	Lotus Development Corporation
Mylar®	E.I. du Pont de Nemours & Co., Inc.
Norton™	Peter Norton Computing, Inc.
1-2-3®	Lotus Development Corporation
PC AT™	International Business Machines, Inc.
PC XT™	International Business Machines, Inc.
Symphony®	Lotus Development Corporation
Timeline™	Breakthrough™ Software Corporation

Index

PC-DOS TIPS & TRAPS

With this book of practical shortcuts and helpful hints you can solve immediate problems at the keyboard and master challenging business tasks on your IBM® PC or PC-compatible computer.

PC-DOS Tips & Traps is for everyone using PC-DOS 2.1, DOS 3.0, or DOS 3.1. In these pages you'll find an array of tips and easy solutions to frequently encountered traps to help you with a wide range of computer operations, from initializing your system and formatting disks, to controlling peripherals and managing the DOS environment.

PC-DOS Tips & Traps offers detailed coverage of

- DOS batch files
- DEBUG for programming
- Filters, piping, and redirection
- Hard disks and hierarchical directories

You'll save computing time and minimize the chance for error with these techniques and insights into both the PC-DOS and MS-DOS operating systems.

Dick Andersen is the author of Osborne/McGraw-Hill's series of **Tips & Traps** books for Jazz,™ dBASE,® and AppleWorks.™ He is also the coauthor of the best-selling **1-2-3™ Tips, Tricks, and Traps** (Que). With more than 17 years of experience in the computer industry, Andersen is director of the consulting firm Net 1 and has served as a consultant to several Fortune 1000 companies.

Janice M. Gessin is a microcomputer consultant who also teaches classes on Lotus™ 1-2-3,™ Symphony,™ and PC-DOS.

Fred Warren is a software developer and systems integrator.

Jack Rodgers is a PC consultant and trainer who offers seminars on PC-DOS and micro-to-mainframe communications.

McGraw Hill

06-DTA-885

ISBN 0-07-881194-5