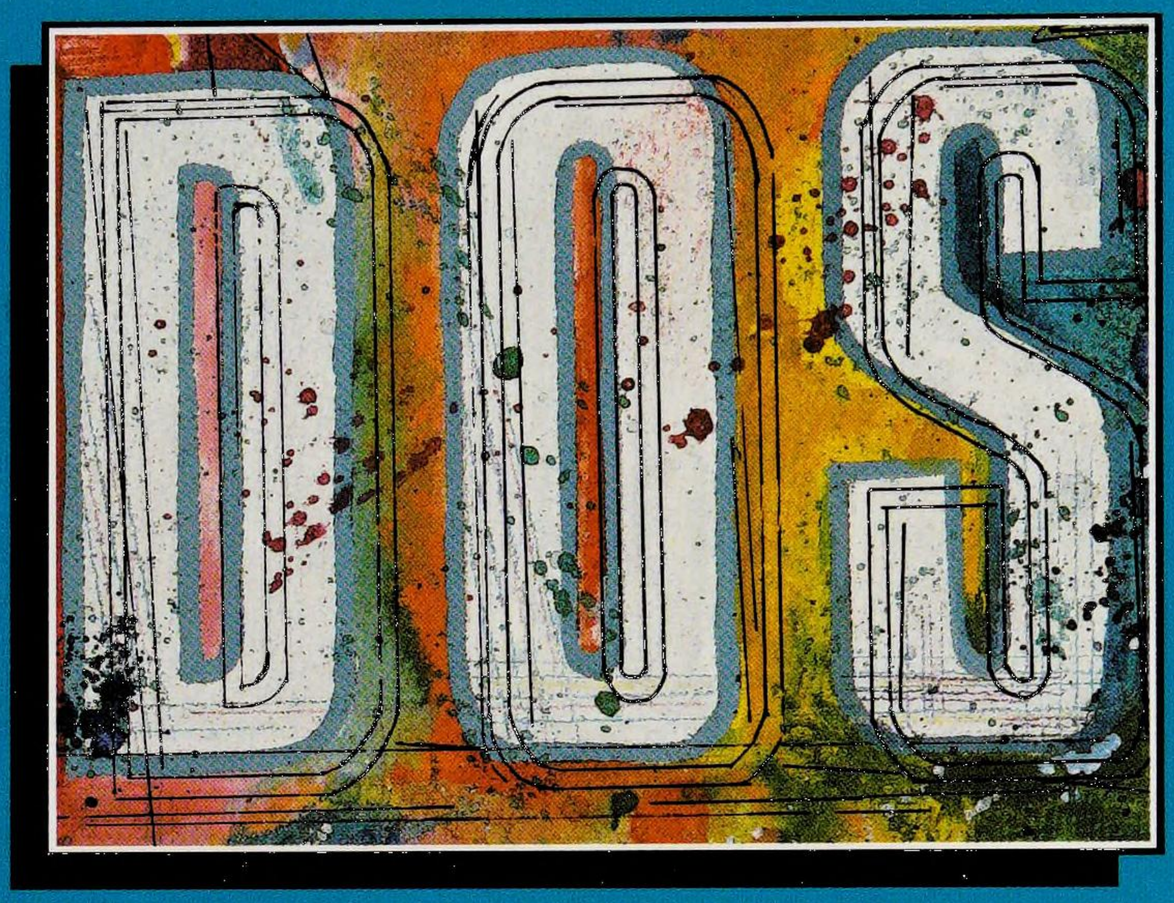


Complete & Simplified

ROD B. SOUTHWORTH

Titles available from boyd & fraser

BASIC Programming

Applesoft BASIC Fundamentals and Style
BASIC Fundamentals and Style
Complete BASIC for the Short Course
Structured BASIC Fundamentals and Style for the IBM® PC and Compatibles
Structured Microsoft BASIC: Essentials for Business
Structuring Programs in Microsoft BASIC

COBOL Programming

Advanced Structured COBOL: Batch and Interactive
COBOL: Structured Programming Techniques for Solving Problems
Comprehensive Structured COBOL
Fundamentals of Structured COBOL

Database

A Guide to SQL
Database Systems: Management and Design

Computer Information Systems

Applications Software Programming with Fourth-Generation Languages
Business Data Communications and Networks
Expert Systems for Business: Concepts and Applications
Fundamentals of Systems Analysis with Application Design
Investment Management: Decision Support and Expert Systems
Learning Computer Programming: Structured Logic Algorithms, and Flowcharting
Office Automation: An Information Systems Approach

Microcomputer Applications

An Introduction to Desktop Publishing
dBASE III PLUS® Programming
DOS: Complete and Simplified
Introduction to Computers and Microcomputer Applications
Macintosh Productivity Tools
Mastering and Using Lotus 1-2-3®, Release 3
Mastering and Using Lotus 1-2-3®, Version 2.2
Mastering and Using WordPerfect® 5.0 and 5.1
Mastering Lotus 1-2-3®
Microcomputer Applications: A Practical Approach
Microcomputer Applications: Using Small Systems Software, Second Edition
Microcomputer Database Management Using dBASE III PLUS®
Microcomputer Database Management Using dBASE IV®
Microcomputer Database Management Using R:BASE System V®
Microcomputer Productivity Tools
Microcomputer Systems Management and Applications
PC-DOS®/MS-DOS® Simplified, Second Edition
Using Enable®: An Introduction to Integrated Software

Shelly and Cashman Titles

Computer Concepts with Microcomputer Applications (Lotus 1-2-3® and VP-Planner Plus® versions)
Computer Concepts
Essential Computer Concepts
Learning to Use WordPerfect®, Lotus 1-2-3®, and dBASE III PLUS®
Learning to Use WordPerfect®, VP-Planner Plus®, and dBASE III PLUS®
Learning to Use WordPerfect®
Learning to Use Lotus 1-2-3®
Learning to Use VP-Planner Plus®
Learning to Use dBASE III PLUS®
Computer Fundamentals with Application Software
Learning to Use SuperCalc®3, dBASE III®, and WordStar® 3.3: An Introduction
Learning to Use SuperCalc®3: An Introduction
Learning to Use dBASE III®: An Introduction
Learning to Use WordStar® 3.3: An Introduction
BASIC Programming for the IBM Personal Computer
Structured COBOL: Pseudocode Edition
Structured COBOL: Flowchart Edition
RPG II, RPG III, & RPG/400

DOS

Complete & Simplified

ROD B. SOUTHWORTH

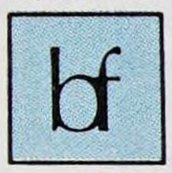
Laramie County Community College

boyd & fraser publishing company

CREDITS:

Publisher: Tom Walker
Acquisitions Editor: Jim Edwards
Editor: Pat Donegan
Director of Production: Becky Herrington
Manufacturing Director: Dean Sherman
Composition: Huntington & Black Typography
Cover Illustration & Design/Book Design: Becky Herrington

Boyd & Fraser



© 1990 by boyd & fraser publishing company
A Division of South-Western Publishing Company
Boston, MA 02116

All rights reserved. No part of this work may be reproduced or used in any form or by any means—graphic, electronic, or mechanical, including photocopying, recording, taping, or information and retrieval systems—without written permission from the publisher.

Manufactured in the United States of America

DOS is a registered trademark of IBM Corporation.
MS-DOS is a registered trademark of Microsoft Corporation.
PC-DOS is a registered trademark of IBM Corporation.

Library of Congress Cataloging-in-Publication Data

Southworth, Rod B., 1941-
DOS : complete & simplified / Rod B. Southworth.
p. cm.
ISBN 0-87835-492-1
1. PC DOS (Computer operating system) 2. MS-DOS (Computer operating system) I. Title.
QA76.76.063S659 1990
005.4'46--dc20 90-1359
CIP

4 5 6 7 8 9 10 K 4 3 2 1

CONTENTS

Preface ix
Objectives of this Book ix
Distinguishing Features ix
Acknowledgments x

Chapter 1 **INTRODUCTION TO MICROCOMPUTERS** **1**

HARDWARE 4
The Central Processing Unit (CPU) 4
Bits, Bytes, and Words 5
Input/Output Devices 6
Secondary Storage 14
SOFTWARE 16
Application Software 17
System Software 17
Review Questions 19
Chapter 1 Lab Exercise 20

Chapter 2 **INTRODUCTION TO DOS** **21**

BASIC DOS FUNCTIONS 24
Control of Input/Output Operations 25
Interpret and Execute Commands 25
File Management 25
SAVING FILES WITH DOS 26
BOOTING DOS 27
FUNDAMENTAL DOS COMMAND CONCEPTS 30
Default Disk Drive 31
Standard Device Names 31
File Naming Conventions 32
Wildcard Characters 34
The DOS Directory Listing 35
Internal vs. External Commands 38
DOS Versions 38
FORMATTING DISKS WITH DOS 39
Review Questions 41
Chapter 2 Lab Exercises 42

Chapter 3 **INTERNAL FLOPPY DISK COMMANDS** **49**

DOS INTERNAL COMMANDS 53
BREAK (Control Break) Command 54
CLS (Clear Screen) Command 54
COPY Command 54
DATE Command 57
DEL (Delete) Command 57

DIR (Directory) Command 58
PROMPT Command 60
RENAME Command 61
TIME Command 62
TYPE Command 63
VER (Version) Command 63
VERIFY Command 64
VOL (Display Volume Label) Command 64
Review Questions 65
Chapter 3 Lab Exercises 66

Chapter 4 EXTERNAL FLOPPY DISK COMMANDS 69

DOS EXTERNAL COMMANDS 71
ATTRIB (Attribute) Command 72
CHKDSK (Check Disk) Command 73
COMP (Compare Files) Command 75
DISKCOMP Command 76
DISKCOPY Command 76
FIND Command 77
FORMAT Command 78
LABEL (Volume Label) Command 79
RECOVER Command 79
SYS (System) Command 80
Review Questions 81
Chapter 4 Lab Exercises 82

Chapter 5 THREE IMPORTANT CONCEPTS 87

BATCH FILES 89
REDIRECTION 95
PIPING (FILTERS) 95
Review Questions 98
Chapter 5 Lab Exercises 99

Chapter 6 DOS EDITING CAPABILITIES 105

EDLIN 107
DOS EDITING KEYS 110
Review Questions 111
Chapter 6 Lab Exercises 112

Chapter 7 HARD DISK CONCEPTS AND COMMANDS 117

DIRECTORIES AND SUBDIRECTORIES 120
HARD DISK COMMANDS 124
FDISK Command 124
CD (Change Directory) Command 125

MD (Make Directory) Command 126
RD (Remove Directory) Command 127
PATH (Set Search Directory) Command 127
TREE Command 128
BACKUP Command 129
RESTORE Command 130
Review Questions 131
Chapter 7 Lab Exercises 132

Chapter 8 **MANAGING HARD DISK SYSTEMS 137**

HARD DISK MANAGEMENT TECHNIQUES 139
MORE HARD DISK COMMANDS 141
APPEND Command 141
ASSIGN Command 142
XCOPY Command 143
Review Questions 144
Chapter 8 Lab Exercises 145

Chapter 9 **CUSTOMIZING DOS 149**

CONFIG.SYS FILE 151
ANSI.SYS 154
VDISK.SYS (RAMDRIVE.SYS) 157
MODE COMMAND 158
Review Questions 160
Chapter 9 Lab Exercises 161

Chapter 10 **ADVANCED DOS COMMANDS 163**

ADVANCED DOS COMMANDS 166
FASTOPEN Command 166
PRINT Command 167
REPLACE Command 169
SUBST Command 170
USING DEBUG 171
Review Questions 173
Chapter 10 Lab Exercises 174

Chapter 11 **ADVANCED DOS BATCH FILES 179**

ADVANCED BATCH FILE COMMANDS 181
IF 182
FOR 185
CALL 187
CREATING EFFECTIVE BATCH FILES 188
Review Questions 194
Chapter 11 Lab Exercises 195

Chapter 12 UTILITY SUPPORT PROGRAMS 201

EXTENDING DOS 204
Improvements to DOS 204
Additions to DOS 205
DOS SHELLS, THE FRIENDLY APPROACH 205
Terminate and Stay Resident Programs 207
The Benefits of Using Windows 207
COMMERCIAL UTILITY PROGRAMS 209
The Norton Utilities Advanced Edition 209
SideKick Plus 211
Treeview 212
PC Tools Deluxe 214
Other Commercial Utility Programs 215
NONCOMMERCIAL PROGRAMS 218
Freeware, Public-Domain, and Shareware Programs 218
Sampling of Utility Support Programs 219
CONSIDERATIONS, CAUTIONS, AND CONCLUSIONS 222
Review Questions 223

Chapter 13 MENU-DRIVEN SYSTEMS 225

INTRODUCTION TO MENUS 227
USING AUTOMENU 231
The Elements of Automenu 231
Format of a Menu Display 233
Building and Modifying Menus 236
Technical Considerations 239
Review Questions 240
Chapter 13 Lab Exercises 241

Appendix A SUMMARY OF DOS COMMANDS 245

Appendix B DOS 4.0 AND OS/2 HIGHLIGHTS 249

Appendix C COMMON DOS ERROR MESSAGES 252

Appendix D ASCII CODES 256

Index 261

PREFACE

DOS: Complete and Simplified is ideally suited for use in any formal educational or training environment, or for self-study. It was developed with a two-credit PC-DOS/MS-DOS course in mind, but is appropriate for use as a supplementary text in any microcomputer course. This text is an excellent reference book for anyone using a microcomputer, be it in the classroom, in the office, or at home. Even though no previous experience with computers is required to use this book, students who typically gain the most from it are those who have already experienced frustrations when trying to use DOS effectively.

OBJECTIVES OF THIS BOOK

☐

The objectives of this book are as follows:

- To provide readers with a fundamental overview of the components of microcomputer systems.
- To introduce readers to the concepts of using an operating system.
- To simplify the use of high-frequency DOS commands and associated options.
- To improve readers' overall ability to use microcomputers effectively through minimized keystrokes, improved disk management, and customized execution of computer processes.
- To acquaint readers with the vast number of utility support programs available to supplement DOS and give more power and control to the user.

DISTINGUISHING FEATURES

☐

Simplifies Using DOS

To accommodate the different backgrounds and levels of expertise of students using this book, topics in this text are developed in a logical step-by-step manner. By building on students' prior experience and carefully constructed examples of DOS in action, this simplified approach helps readers become self-sufficient microcomputer users.

Focus on High-Frequency DOS Commands

This textbook features step-by-step instruction in using the DOS 3.3 commands and associated options that are most frequently required by microcomputer users. It is designed to help readers gain better understanding and control of microcomputers through efficient use of DOS.

Free Automenu and Treeview Software

Free copies of Automenu, the de facto standard for menu systems; and Treeview, a file management utility program for hard disk systems, are included in the Instructor's Manual. This software may be copied by the instructor for use by students in any course where *DOS: Complete and Simplified* is the adopted text. Any adopter choosing to register their copies of Automenu and Treeview are eligible for software support and actual documentations from Magee Enterprises, Inc., manufacturers of the software.

Floppy Disk vs. Hard Disk Environments

In keeping with the current trend in microcomputer instruction, this book consistently addresses the use of DOS in both floppy and hard disk environments. Exercises in the text are designed for floppy disk environments, but are modified for hard disk environments in the Instructor's Manual.

Creation of Effective Batch Files and Customizing DOS

The text extensively covers the creation of effective batch files and the customization of DOS, thus making students more efficient and advanced DOS users.

Class-Tested Exercises

Each chapter includes a substantial set of student exercises that have been class-tested over the last three years. These exercises build on material learned from previous chapters, as well as reinforcing the new material contained in each chapter. These exercises are specifically designed for floppy disk systems, but are modified for hard disk systems in the Instructor's Manual.

Actual Screen Illustrations

Each DOS instruction is fully supported with screen "dumps" that accurately reflect what users' screens will look like as they execute each target command. The screen illustrations provide users with visual verification, which highlights the impact of each operation performed.

Proven Material

This text is based on many semesters of teaching this course and on the collective experience of the instructors and students who have shared their comments and suggestions. Every attempt has been made to preserve the integrity of those elements that proved effective and to improve on those that did not.

Instructor's Support Material

An Instructor's Manual featuring additional student exercises, helpful teaching suggestions, answers to chapter review questions, a selection of class-tested, multiple choice test questions, and transparency masters is available at no charge for adopters of this text. Instructors may contact South-Western Publishing Company to obtain this supplementary material.

ACKNOWLEDGMENTS

This book would not have been possible without the guidance, help, and advice of many supportive individuals. I would like to offer my sincere thanks to the students and instructors at Laramie County Community College who had faith in my material and never failed to make valuable comments about whatever they did and did not like.

I would also like to offer my gratitude to the following individuals who provided valuable suggestions during the development of this text: Bill Gardner, Southern Technical College; Jean E. Gutman, University of Southern Maine; Michael Michaelson, Palomar College; and Paul Ross, Millersville University.

The entire staff at boyd & fraser, especially Pat Donegan and Jim Edwards, did a remarkable job of editing and producing this text. To all of these people, I remain indebted for their efforts on my behalf. A special thanks goes to my wife Patrice, whose constant encouragement and proofreading assistance was a tremendous help.

Rod B. Southworth
Cheyenne, Wyoming

January 3, 1990
CompuServe 71210, 1546

Chapter

1

INTRODUCTION TO MICROCOMPUTERS

HARDWARE

- ☐ The Central Processing Unit (CPU)
  Bits, Bytes, and Words
  Input/Output Devices
  Secondary Storage

SOFTWARE

- ☐ Application Software
  System Software

Chapter

1

INTRODUCTION TO MICROCOMPUTERS

After reading this chapter, you should understand the major components of microcomputer systems. Because students using this text undoubtedly have varying degrees of computer experience and knowledge, this chapter provides a common framework of microcomputer concepts and terminology.

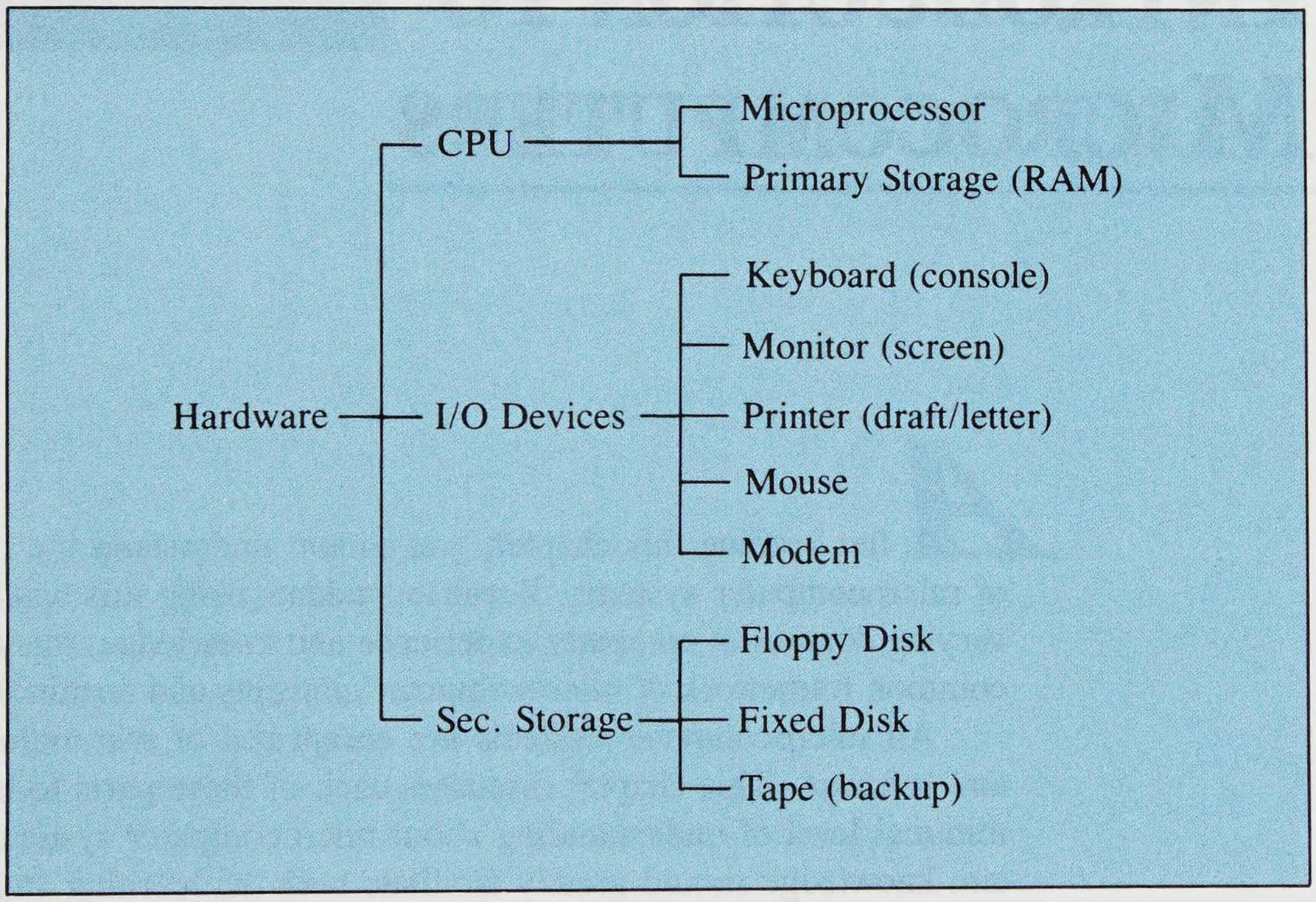
All microcomputer systems are comprised of two major parts: hardware and software. This chapter discusses each of these parts to bring students to a minimal level of understanding about microcomputer systems. This basic system knowledge should greatly facilitate both the learning and understanding of either PC-DOS or MS-DOS.

When you purchase a microcomputer system, you may need to make choices about the power of the CPU and the types of input, output, and storage devices you will want to attach. The level of technical knowledge presented in this chapter will assist you in making the right choices.

HARDWARE

Typically, discussion of **hardware** involves three categories: the central processing unit (CPU), the various input/output (I/O) devices, and storage devices. Figure 1-1 summarizes the various hardware parts and categories of microcomputer systems discussed in this chapter.

Figure 1-1
Microcomputer Hardware



The Central Processing Unit (CPU)

The **central processing unit**, or **CPU**, has often been described as the "heart" of a computer system. It is comprised of a **microprocessor** and a varying amount of temporary storage locations referred to as **primary storage**. The CPU is typically organized around one of four microprocessor chips designed by Intel Corporation: the 8088, 8086, 80286, and 80386. Each of these microprocessor chips has different capabilities, related primarily to speed and overall processing power. Soon, microcomputers will utilize the most powerful microprocessor chip, the Intel 80486.

Bits, Bytes, and Words

All computer circuits, including microprocessors, function in one of two states: on or off. Symbolically, we represent the on condition by the value 1 and the off condition by the value 0. These two values are binary digits, or **bits**. A grouping of bits can be combined to represent characters that we need to store data on a computer. Eight bits are typically grouped together to represent characters, where a character is a number (0–9), alphabetic letter (A–Z), or special symbol such as an asterisk, dollar sign, decimal point, and so on. For example, the bit pattern 10000001 might represent the letter A.

When each group of 8 bits is individually addressable it is called a **byte**. Most of the earlier computers were "byte machines." However, it is generally more efficient to access and work with more than one character at a time. When bytes are grouped (always in multiples of 2), the addressable groups are called **words**. A 16-bit word represents 2 characters, a 32-bit word represents 4 characters, and so on. Word machines access and transfer characters faster than byte machines.

The 8088 microprocessor is the most common of the four chips. It has a 16-bit internal word structure with an 8-bit path for transfer of input and output data. The 8086 is a similar chip, but with a 16-bit path for input and output, which allows for faster transfer of data between the CPU and I/O devices.

All microprocessor chips use a **clock rate** that determines the frequency of the internal operations and keeps everything in proper synchronization. The faster the clock runs, the faster the computer can process data and instructions. Clock rates are measured in units called **megahertz**, a term for one million cycles per second. The internal clock speed of both the 8088 and 8086 chips is 4.77 megahertz (MHz).

The 80286 microprocessor has a 32-bit internal word length with a 16-bit I/O path. Its internal clock is rated at 8–12 MHz, making it at least twice as powerful as the older, more common chips previously mentioned. The 80386 microprocessor is at least twice as fast as the 80286 with an internal clock speed of 16–33 MHz. The processing power of the 80386 is often required for the high-powered graphics used in desktop publishing or computer-aided design (CAD) applications. But don't get discouraged if you don't have the latest chip. The more common chips used today, while not as fast and powerful as the 80386, are more than adequate for most applications. As my dad once told me, "When the speed limit is only 55 MPH, a VW bug can be as effective as a fancy sports car."

Primary storage, the second major part of a CPU, is a temporary holding location for both programmed instructions (software) and data to be processed.

The number of primary storage locations on microcomputers typically ranges from 256KB to 768KB, where KB is roughly equivalent to a **kilobyte**, or 1000 characters. Actually, one KB of storage is 1024 bytes, but it is a lot simpler to work in units of 1000. The microprocessor is responsible for executing software instructions that tell the computer how to process the data in primary storage. These instructions also tell the microprocessor when and where to send data to an output device, such as a printer, as well as when and where to get additional data to be processed.

Primary storage is generally referred to as **RAM (Random Access Memory)** because the storing of data causes the affected storage locations to be changed and allows for any storage location to be accessed at any time. Characters are stored in a given storage location and remain there until new characters have replaced them, or until the electricity has been turned off. Because most RAM chips lose their "memory" when the power is discontinued, primary storage is considered temporary. If you want to permanently save data, you must save it to a secondary storage device such as magnetic tape or disk.

It is important for microcomputer users to realize the potential damage that static electricity can do to the sensitive electronic circuits in the CPU. The amount of static electricity that you sometimes feel when you touch a doorknob or another person is hundreds of times greater than the static electricity needed to permanently damage a microprocessor or RAM chip. For this reason, take precautions to minimize the potential for static electricity around your microcomputer. Protect the computer or computer area by the following techniques:

- Place the computer system in a noncarpeted area. Keep the computer area at a relative humidity of about 45% or more.
- Use a static spray on fabrics.
- Use a static mat and good grounding to discharge static electricity.

Input/Output Devices

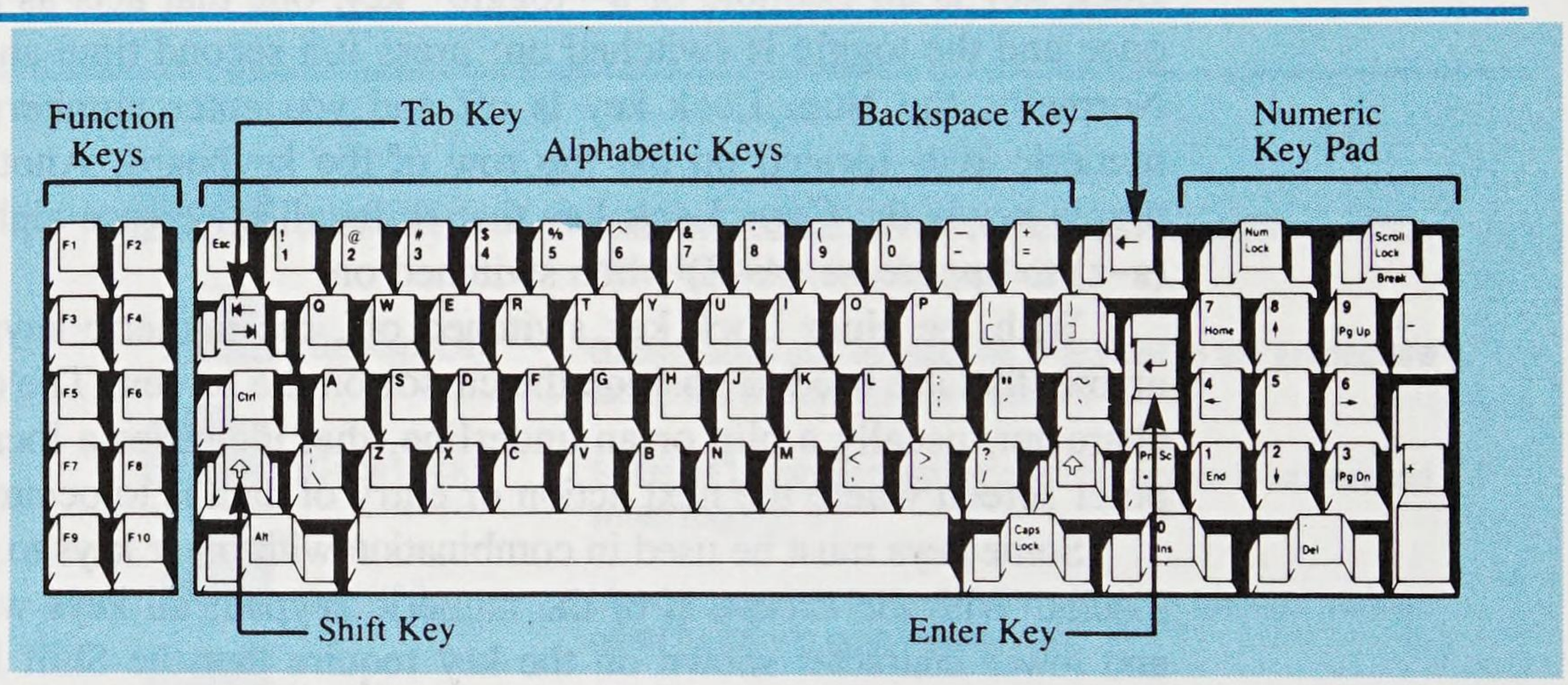
Input/output devices are the means by which you enter data into the computer (input) or view data you have previously saved (output). This chapter discusses the most common devices: keyboard, monitor, printer, mouse, and modem.

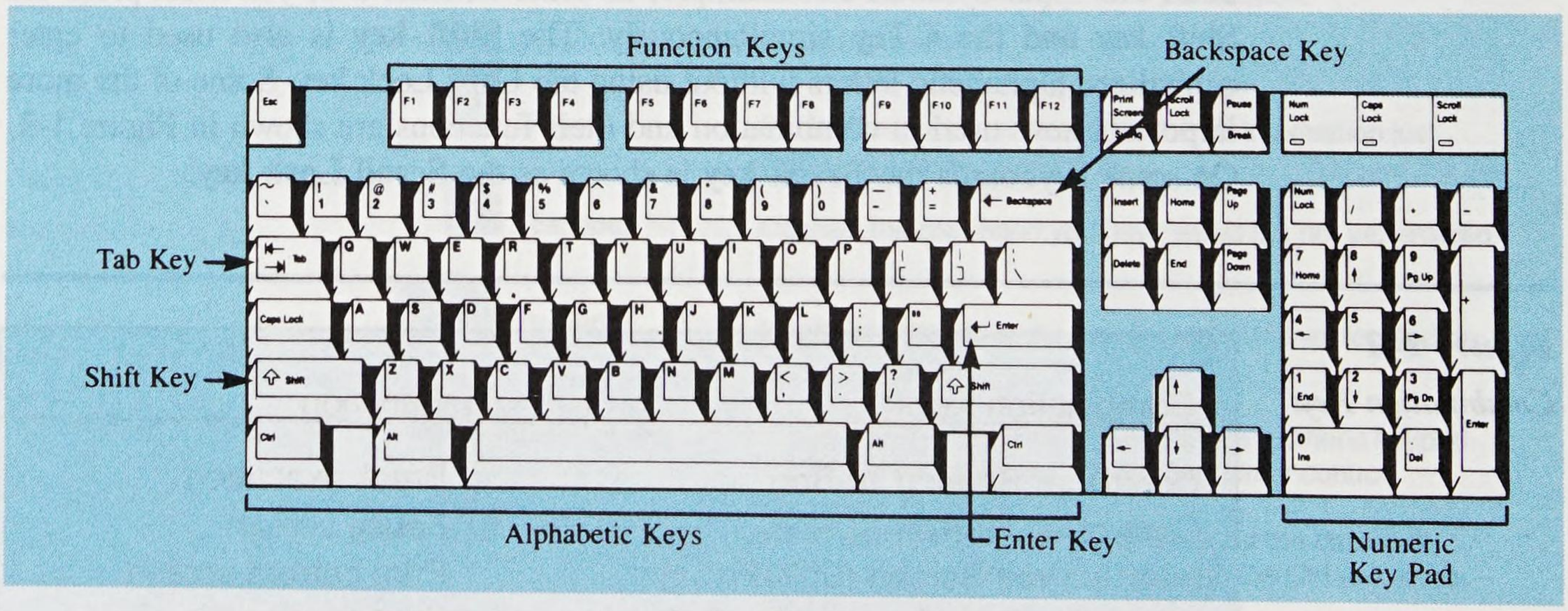
The Keyboard

The **keyboard** on a personal computer is an input device similar to a typewriter keyboard, except it has additional special keys. A good understanding of the keyboard is essential when working with DOS. The IBM-PC and IBM-compatible keyboards typically have between 83 and 101 keys (see Figure 1-2) and vary somewhat among manufacturers. Most of the variation occurs in the number keys and in the placement of keys. The cost of a typical keyboard is about $100.

Figure 1-2

Typical Keyboards





A set of 10 keys, usually on the left side (or additional top row) and labeled F1 through F10, are called **function keys**. These "programmable" keys serve different functions depending on how the software has been programmed to use them. When a function key is referred to in this text or in the DOS Manual, it will generally be called by the key name, such as the F1 key or the F6 key. When you are instructed to enter the F6 function key in the text exercises, it will be shown as <F6>.

The **numeric keypad** is a group of 10 keys, usually located near the right side of the keyboard, containing keys numbered 0–9. The keypad can be used to enter numbers only when the Num Lock key has been activated. The **Num Lock key** is an example of a "toggle" key, one that acts as a switch. Press it once and the toggle is switched on; press it a second time and it is turned off. Normally, the Num Lock key is off and you enter numbers by pressing the numeric keys located on the top row of the keyboard. Another example of a toggle key is the **Caps Lock key** that shifts all lowercase alphabetic characters (a–z) to uppercase (A–Z) when switched on.

With the Num Lock key switched off, the numeric keypad keys become arrows that are used to position the cursor on the screen. The cursor is a special character, usually a blip or an underline, that identifies a location on the computer screen where the next action or entry of data is to occur.

Some keys must be used in combination with other keys to obtain the desired results. With the exception of the numeric keypad, all keys with both an upper and lower character shown on the key require that the Shift key be pressed to enter the upper symbol. For example, in order to enter a $, you must press the Shift key and the 4 key simultaneously. The **Shift key** is also used to enter capitalized alphabetic letters without using the Caps Lock key. Some of the more important keys used in combination and their functions are shown in Figure 1-3. On some keyboards the **Break key** is shown as the **Scroll Lock key**.

Figure 1-3

Combination Keys

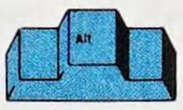
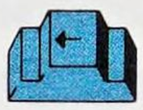
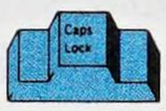
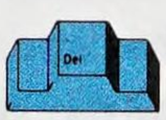
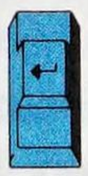
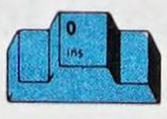
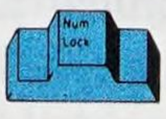
Combination Keys	Function
Control + Break (Ctrl-Brk)	Break execution
Control + S (Ctrl-S)	Pause screen
Shift + Print Screen (Shift-PrtSc)	Print current screen
Control + Print Screen (Ctrl-PrtSc)	Print continuous screen
Control + Alt + Del (Ctrl-Alt-Del)	System reset (warm boot)

As you enter commands from a keyboard, you can use the **Backspace key**, normally shown as a large left-facing arrow (←), to backspace and erase unwanted characters. The Backspace key erases one character at a time each time it is pressed.

Once a command has been keyed completely, it must be sent to the computer, or entered, by pressing the Enter key. The Enter key, sometimes shown as the Return key, is normally located near the right side of the keyboard, just to the left of the numeric keypad. On most keyboards it is shown as a bent left-facing arrow (↵). We refer to this key as the **Enter key** in the rest of the text. Figure 1-4 summarizes the keyboard keys.

Figure 1-4

Summary of Keyboard Keys

 (Alternate)	Used in combination with other keys to perform various other functions (similar to Ctrl).
 (Backspace)	Backspaces and erases the character to the immediate left of the current cursor position.
 (Caps Lock)	Shifts all lowercase alphabetic characters to uppercase when toggled on.
(Control)	Used in combination with other keys to perform various other functions (see Figure 1-3).
 (Delete)	Deletes the character at the cursor location.
 (Enter)	Sends the current command or keyed information just typed to DOS for processing.
(Escape)	Cancels the command you have typed but not yet entered.
 (Insert)	Inserts characters at the cursor location when toggled on.
 (Num Lock)	Toggle key that determines whether the numeric keypad will be used to enter numbers or for cursor control.
(Print Screen)	When used with the Shift key, prints out the data that is displayed on the screen. Used with the Ctrl key, it can print screen displays continuously.
(Scroll Lock)	Used in combination with the Ctrl key to break the execution of a command. Also called the Break key.
(Shift)	Used like the shift key on a typewriter to enter uppercase alphabetic characters.

Since modern keyboards are electronic and not mechanical, a very light touch is all that is required to activate the keys. If you continue to hold down a key, the keyboard will repeat that keystroke until you release the key. This is known as the repeating key concept. It may cause problems for you until you get used to it.

In computer terminology, a **buffer** is a holding area that can temporarily store a limited number of computer characters. The keyboard has a buffer that allows it to retain keystrokes until the program you are using has "caught up" with you. If you are an extremely fast typist or are using a program that processes keystrokes slower than you are entering them, the buffer can save them until they are needed. As you type, some keystrokes will not be automatically "echoed" on the screen, since they may still be in the buffer.

The Monitor

In addition to requiring the use of a keyboard, personal computers need a **monitor** (screen) to communicate with the user. Keystrokes entered at the keyboard are displayed on the screen to provide visual verification. When the computer needs to communicate with you, it normally displays data, error messages, and system prompts on the screen. Most monitors can display 80 characters of text on a line and 25 lines on the screen.

The sharpness and clarity of images on the screen are directly related to the screen resolution. The higher the resolution, the sharper the image. Screen resolution is usually rated by the number of picture elements (pixels) on a screen that can be lit to form images. Low-resolution monitors display about 320 times 200 pixels on the screen, 320 pixels horizontally and 200 pixels vertically. Medium-resolution monitors, like most IBM-PCs, typically have 640 times 200 pixels. The highest resolution monitors, like those used by engineers for complex computer graphics, often go as high as 1024 times 1024 pixels.

There are two major categories of monitors: monochrome monitors and color monitors. Monochrome monitors are limited to a single color, usually amber, green, or white, and are suitable for many applications. If an application requires multiple colors or graphics, a color monitor is required.

A control unit allows the monitor to communicate with the CPU. These control units are called **video display adapters**. Two common video display adapters are the monochrome adapter and the Color Graphics Adapter (CGA). A CGA monitor is limited to 4 colors in the graphics mode of operation. For this reason, CGA cards are being replaced with more powerful adapters, such

as the EGA or VGA cards. The EGA (Enhanced Graphics Adapter) card has been the standard in high-resolution color graphics. All EGA cards can generate up to 64 colors and offer resolutions of 640 times 350 pixels. The VGA (Video Graphics Array) display adapter is the new standard in high-resolution graphics. VGA cards offer resolutions of 640 times 480 pixels and up to 256 colors. The cost of a high-resolution color monitor with an EGA card is about $500, whereas the cost of a high-resolution monitor with a VGA card is about $800.

The Printer

Printers for microcomputers are generally classified as one of two types: letter quality or draft quality (dot matrix). **Letter quality printers**, which print documents that look as if they were created by a typewriter, typically create each character by striking a fully formed image of a character against an inked ribbon and paper. **Draft quality printers**, which produce images of lesser quality but are sufficient for most applications, use a dot matrix technique that creates a pattern of dots to represent each character or image. Many dot matrix printers can produce near letter quality output by reprinting each character, adding additional dots to fill in the image with a denser pattern of dots. The printing of additional dots tends to cut printing speeds by half or more. As a general rule, dot matrix printers are faster and less expensive than letter quality printers. In addition, dot matrix printers are more flexible in that they can print a wide variety of patterns, including graphics.

Printers for microcomputers typically print 80 characters per line, but by printing in condensed mode they can print 132 characters on a line. Prices for printers vary greatly due to the wide range of capabilities, but a fairly typical printer sells for about $300 and prints 120 characters per second.

The latest type of dot matrix printer is the **laser printer**, which uses a toner process similar to the common copy machine. Laser printers have extremely high quality print, handle graphics well, are quite fast, and cost about $2000. These printers are becoming very popular for business and desktop publishing applications using microcomputers.

The Mouse

Most microcomputers allow a mouse to be attached as an input device. The **mouse** is a pointing device. As you move the mouse across a flat surface, it relays information to the computer that moves the cursor in the direction you move the mouse. Once the curser is positioned, you can press a button on the mouse to inform the computer you have selected the desired spot on the screen. A mouse allows you to select options on the screen quickly and easily. However, you need software, called a mouse driver, that supports a mouse before you can use it to control the cursor. A typical mouse with its supporting software costs about $100.

The Modem

To communicate with other computers via telephone lines, two items are required: a **modem** (at each end) and compatible communications software. A modem is used to convert the computer's digital signals to the analog signals used by telephones, and vice versa. Modems can be internal (located inside the computer) or external. The speed with which the modem can send and receive data is known as its **baud rate**. The higher the baud rate, the less time it takes to transmit and receive data. Common baud rates are 1200 and 2400, but data is transmitted as high as 9600 baud. If you divide the baud rate by 10, you can approximate the data transmission rate in more relevant terms, characters per second.

Communications software is readily available to let you use your modem effectively. The major purpose of communications software is to guide you through the various communications parameters required to let your computer talk to another computer. In addition to the baud rate, communication parameters include the specific bit patterns for transmission and the methods of error detection and correction. A 1200-baud modem with supporting software generally costs between $100 and $200.

The I/O Interface

Any input/output device attached to a CPU must have some type of control unit to allow it to interact with the CPU and make the appropriate translations between each I/O device and the CPU. These control units are often referred to as **interface boards** or **cards**. They consist of a sturdy card containing electronic chips. Occasionally, static electricity or some other power-related problem causes these chips to fail and they may have to be replaced.

These cards are relatively easy to remove because they plug directly into slots on the CPU's system board. The system board is the large printed circuit board that holds most of the computer chips that make up the CPU. The chips on the system board include the microprocessor, RAM chips, and ROM chips. The number of slots available (sometimes called expansion slots or option slots) depends on the system board. Most system boards have slots for one keyboard, a monitor (display screen adapter), two serial devices (like a mouse or a modem), a parallel device (like a printer), and two or more disk drives. These slots are all connected to a common communications channel called a bus. The bigger the bus, the faster data can be transmitted inside the CPU. The bus on most microcomputers is either an 8-bit or a 16-bit path.

Slots for a variety of serial and parallel devices are referred to as ports. Data is transmitted through a serial port via a single wire one bit at a time. The most common serial port is based on an industry standard called the RS-232 serial interface. It is designed for a wide variety of uses. Because each character requires eight bits of data, serial transmission is not very fast. Considerably faster transmission speeds are obtained with a parallel port. These ports use eight parallel wires to send data eight bits at a time.

Because the number of slots is limited, manufacturers have developed multifunction boards, which combine two or more tasks. Some of the functions typically combined on multifunction boards include additional primary storage, a clock and battery to automatically set the computer's clock, a parallel port to connect a printer, several RS-232 ports to connect serial devices like a mouse or modem, and a game adapter port to connect a joystick.

Secondary Storage

When you create data on a computer or write programs, you cannot save them permanently in RAM, the computer's primary storage area. RAM is not large enough to store even a modest number of files containing data and/or programs. Secondary storage facilities are required to store a potentially unlimited amount of data on a permanent basis. The costs per bit stored on secondary storage are considerably less than for storage in primary storage. Microcomputers usually support three types of secondary storage devices: floppy disks, hard (or fixed) disks, and magnetic tape.

Floppy Disks

Floppy disks are the most common medium for secondary storage and come in various sizes. The most common size for the IBM-PC family of personal computers is 5 1/4 inch disks, each of which can contain up to 360KB (or 360,000 characters) of data. These disks allow data to be written on 40 tracks per side, using both sides of the disk for storage. Some 5 1/4 inch disks, called high-density disks, can hold up to 1,200,000 bytes (1.2 megabytes) of data. (A **megabyte**, **MB**, is equivalent to about one million characters.)

A relative newcomer, the 3 1/2 inch microfloppy disk, is gaining rapid acceptance by microcomputer users. Not only is it better protected and smaller in physical size, but it holds more data than the 5 1/4 inch floppy disk. The low-density microfloppy typically stores 720KB, while the high-density model stores 1.44MB. DOS Version 3.2 supports these newer disks, but the older versions do not.

When you insert a floppy disk into a floppy disk drive and close the latch, the disk is secured by two clamps that close on the large center hole of the disk. When data is to be read from or written to the disk, a motor connected to the clamps spins the disk inside its protective jacket at 300 RPM, about 10 times faster than a 33 1/3 album spins on a record turntable. A small red light on the disk drive indicates when the drive is spinning, so you will know not to attempt to remove the disk.

Floppy disks are not completely reliable because they can be damaged. However, if you take proper care of them, floppy disks can serve you faithfully for a long time. Figure 1-5 contains some helpful and important tips for the "care and feeding" of floppies.

Figure 1-5
Proper Care of Floppy Disks

1. Store floppies in their protective covers when not in use.
2. Store them vertically to minimize chances of warping.
3. Shade them from direct sunlight or intense heat.
4. Write on any attached labels with felt tip pens only.
5. Never touch the recording surface of a floppy disk.
6. Never bend, fold, or otherwise mutilate a floppy disk.
7. Keep the disk clean from contaminants, including smoke.
8. Do not attempt to clean the surface of a disk.
9. Never place a disk near a magnetic field or magnets.

Hard Disks

Hard disks, referred to as **fixed disks** by IBM, are not removable. Because hard disks are "fixed," they have a significant advantage over floppy disks in that they are roughly 15 to 20 times faster to use. When the hard disk drive is operating, it spins at 3600 RPM, or 12 times faster than a floppy disk. In addition, you do not have to be continually swapping disks, because hard disks will typically hold 20, 40, or even 191 megabytes of data. Hard disks can store data using a much higher density than floppy disks because they use rigid metal platters in a sealed environment that protects them from contaminants. Because of the vast number of files that can be stored on most hard disk systems, it is important to establish appropriate file naming conventions, a topic covered in Chapter 2. A 20MB hard disk can store the equivalent of 56 360KB floppy disks.

It is not practical for personal computers to have only a hard disk drive installed, however. Floppy disks are used extensively for transferring data and programs from one computer to another and for backup, so it is ideal to have at least one hard disk drive and one floppy drive.

Magnetic Tape

Another medium of secondary storage is **magnetic tape**. Since this medium only allows for sequential retrieval of data, it is not appropriate for most applications. Magnetic tape is significantly less expensive than disks (per unit stored), so it makes an ideal medium for storing historical or backup data. Large businesses typically use one of their expansion slots on the system board to connect a high-speed tape drive to be used solely for backup. A fast, high-capacity tape drive that greatly facilitates backup costs about $400.

SOFTWARE

In addition to hardware, a computer system must have software to control and operate the hardware. Figure 1-6 outlines the various classifications of software required.

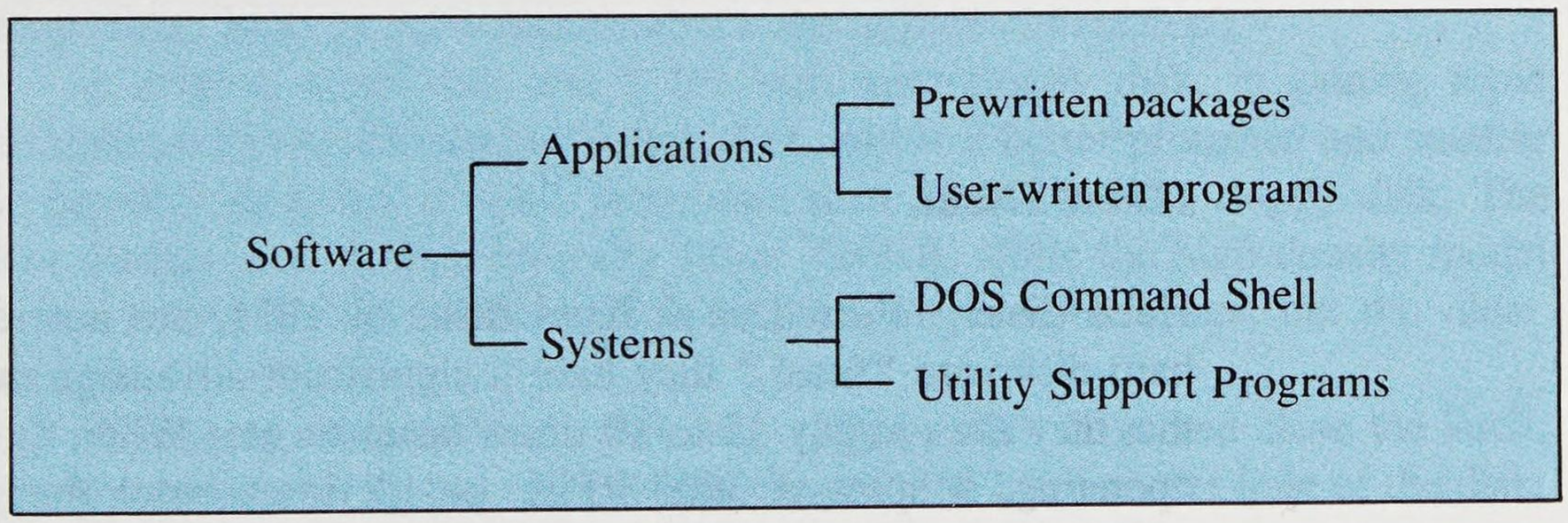


Figure 1-6
Microcomputer Software

Application Software

Application software is written for a specific purpose, such as inventory, word processing, or payroll. It can be created by the user at great expense and time or it can be purchased outright. Novice users should not attempt to write their own application software; it takes a great deal of skill to be able to write reasonably productive software. Furthermore, the cost of prewritten programs is relatively low.

Individual application programs are more helpful if they are **integrated** (provide several applications in one package). Integration provides two major benefits. Learning how to operate the programs is easier, since commands are usually similar in integrated packages. In addition, it is easier to transfer data between a variety of integrated applications. For example, you can easily transfer data from a spreadsheet to your word processor.

System Software

Your computer system needs **system software** to act as an interface between the application programs and the hardware. The two major functions of a computer's operating system are to control the computer's operations and to manage files. The operating system allows you to use the computer in either an interactive mode or a batch mode. In normal **interactive processing** the user enters a command, the computer executes that command, and then the user issues another command when the computer has finished the previous one. **Batch processing** occurs after a user has stored a file of commands. The user can direct DOS to automatically execute the commands in the file one after another. This text is devoted to the operating system used by the IBM and IBM-compatible family of microcomputers (PC-DOS and MS-DOS) and covers both methods of operation.

System software is generally thought of as being both a **command shell** and a set of supporting **utility programs**. The command shell for the IBM-PC family is PC-DOS. Even with all its power and capabilities, many users have found the need for additional software, primarily to assist them in making DOS friendlier (easier to use) and more powerful, so they turn to utility programs like Automenu, PC-Tools, or Treeview. Most utility support programs are devoted to improving upon DOS, such as Microsoft Windows, Norton Utilities, and Fastback, to name just a few. Chapter 12 discusses utility support programs and provides detailed examples.

The last topic that we should mention before delving into DOS is a special kind of software called firmware. **Firmware** is defined as software contained in the form of hardware. This hardware, used to store preprogrammed instructions, is called **ROM** (**Read-Only Memory**). Special-purpose programs are built into ROM chips during manufacturing. Because programs stored in ROM are permanently embedded into computer memory, they are available to be executed without having to be specifically loaded into RAM. ROM is a form of nonvolatile memory in that it does not lose its instructions when the power is turned off like RAM. Therefore, ROM is often used to hold operating system startup programs and language translators such as BASIC.

When you turn on your computer system, a ROM chip automatically tests the hardware for problems and then loads a portion of DOS from the default disk drive. If you have an IBM-PC (not one of the many IBM clones or IBM-compatibles), the instructions for interpreting BASIC programs are contained on a proprietary ROM chip. In the future, more software will be available in the form of inexpensive ROM chips. Currently, laptop and other portable microcomputers contain both the operating system and several common application programs in ROM.

I bought the latest computer;
It came completely loaded.
It was guaranteed 90 days,
But in 30 was outmoded.

–Anonymous

Review Questions

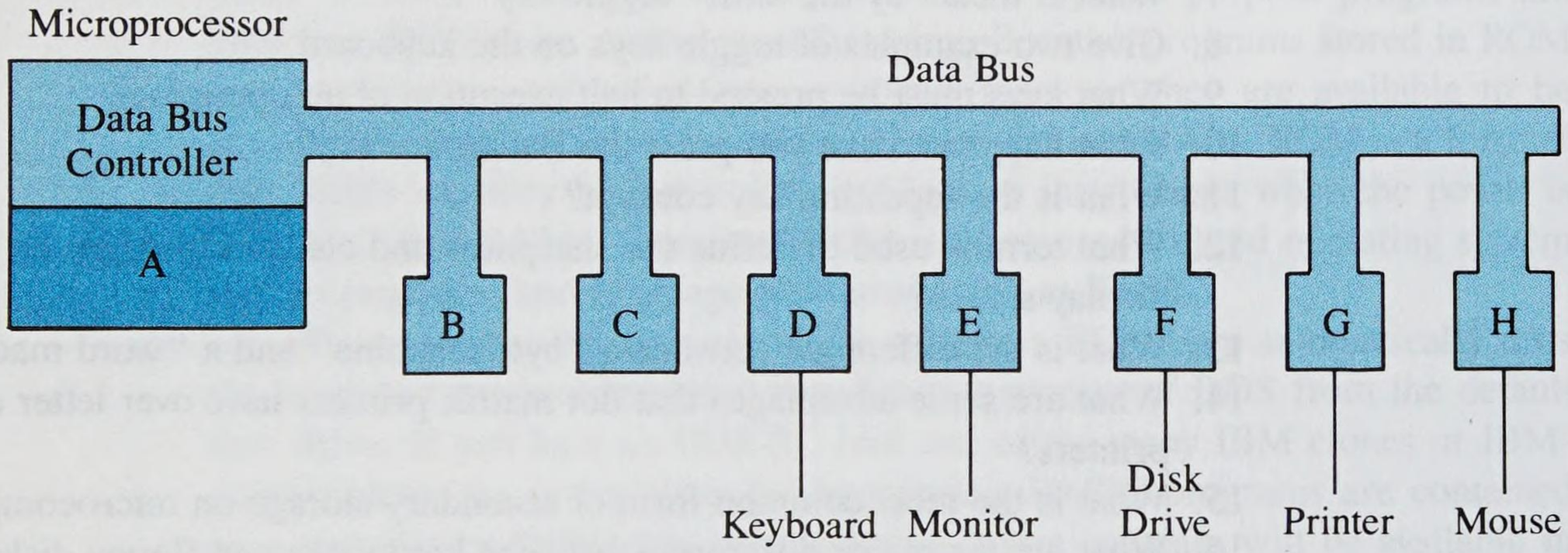
1

1. What are the two major parts of a microcomputer?
2. What is the major difference between primary storage and secondary storage?
3. What are the major differences between RAM and ROM?
4. What is the purpose of a clock rate in a microprocessor?
5. What is the potential danger of static electricity?
6. What is the purpose of function keys?
7. What is meant by the term "toggle key"?
8. Give two examples of toggle keys on the keyboard.
9. What keys must be pressed to halt execution of an operation?
10. What happens when you press the Backspace key?
11. What is the repeating key concept?
12. What term is used to define the sharpness and clarity of images on the display screen?
13. What is the difference between a "byte machine" and a "word machine"?
14. What are some advantages that dot matrix printers have over letter quality printers?
15. What is the most common form of secondary storage on microcomputers?
16. What are the major differences between hard disks and floppy disks?
17. What is the difference between application and system software?
18. What are the major benefits of utility support programs?
19. What is the difference between batch and interactive processing?
20. What are the benefits of using integrated software packages?

Chapter 1 Lab Exercise

A data bus is an electronic connection that links all the major components of a microcomputer system such as RAM, ROM, and the I/O devices. A data bus controller in the microprocessor uses the data bus to transfer data between the components. The data bus path is normally 8, 16, or 32 bits. The bigger the data path, the faster the transfer rate.

Using what you learned in this chapter, match the components labeled (A to H) in the following diagram to the component names provided.



_____ Parallel port
_____ Serial port
_____ Internal clock
_____ Display adapter
_____ Random Access Memory (RAM)
_____ Read-Only Memory (ROM)
_____ Keyboard interface
_____ Disk controller

Where would expansion slots have to be added to this diagram? What would you have to add to connect a modem? What size data bus path would you expect to find if the microprocessor were an Intel 80386 chip? Why?

Chapter

2

INTRODUCTION TO DOS

BASIC DOS FUNCTIONS

- ☐ Control of Input/Output Operations
  Interpret and Execute Commands
  File Management

SAVING FILES WITH DOS

- ☐

BOOTING DOS

- ☐

FUNDAMENTAL DOS COMMAND CONCEPTS

- ☐ Default Disk Drive
  Standard Device Names
  File Naming Conventions
  Wildcard Characters
  The DOS Directory Listing
  Internal vs. External Commands
  DOS Versions

FORMATTING DISKS WITH DOS

- ☐

Chapter

2

INTRODUCTION TO DOS

The primary objective of Chapter 2 is to teach you the basic functions of DOS related to file management and command processing. By the end of this chapter, you will understand such file management techniques as how disks are formatted and how files are saved on disk. In addition to how to boot DOS, Chapter 2 also teaches you fundamental DOS command concepts including the default disk drive, standard device names, file naming conventions, wildcard characters, and internal vs. external commands. This chapter introduces you to four introductory DOS commands: DATE, TIME, DIR, and FORMAT.

An **operating system** is an integral part of all computer systems. It allows users like yourself to conveniently use the computer as a tool. For example, suppose you wanted to use your computer to create a term paper with a new word processing program you recently acquired. You would have to know how to load and execute the correct application program. In addition, that application program has to know how to save and retrieve disk files on your computer. This is all provided for you by your computer's operating system.

The operating system is a necessary translator between the hardware and either application programs or users, coordinating and controlling all the activities of the computer (see Figure 2-1). The operating system contains a group of commands and programs that allow users to interact directly with the computer. For example, it provides an easy way to copy data from one disk to another, allowing you to conveniently make backup copies of important data.

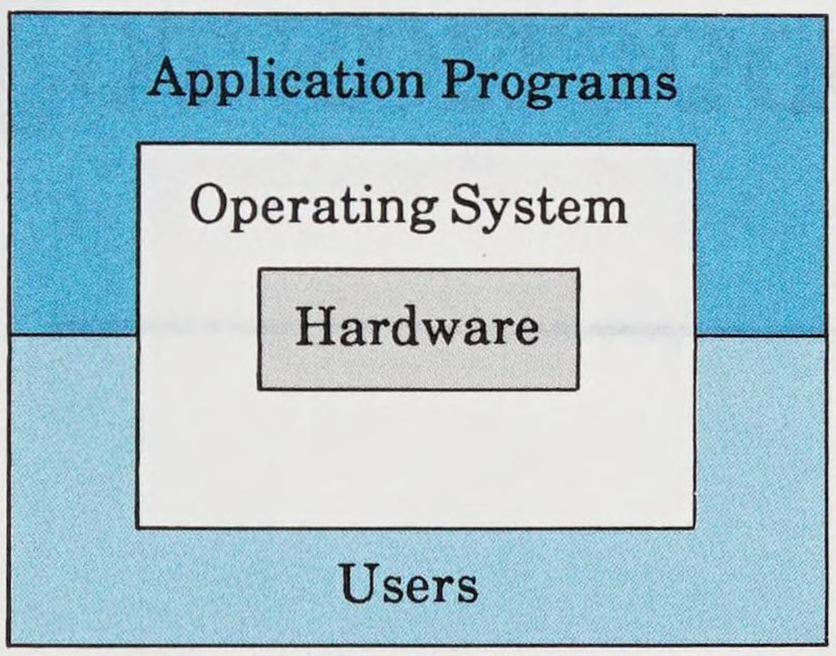


Figure 2-1
The Role of an Operating System

This chapter introduces you to a specific type of disk operating system (DOS) called PC-DOS (or MS-DOS) that is used on both IBM and IBM-compatible microcomputers. Many users are confused by the terms PC-DOS and MS-DOS. Essentially, the two operating systems are identical. **PC-DOS** is the IBM implementation of DOS and **MS-DOS** is Microsoft's version used on all IBM-compatible microcomputers. Because of the similarities of PC-DOS and MS-DOS, this text refers to this operating system simply as DOS. When a specific reference is needed, the appropriate term will be used.

BASIC DOS FUNCTIONS

DOS has three major functions. The first controls the input and output operations of your computer. The second function interprets and executes commands that you enter from your keyboard or other input devices. The third major function deals with file management, which allows you to permanently record files on disks and manage them effectively.

Control of Input/Output Operations

All application programs share the same input and output problems. They all have to accept data from the keyboard, display data on the monitor, store data temporarily in main memory, store data permanently on disk, and retrieve data from disks. It requires a great number of instructions to coordinate and control all these activities on a microcomputer. Without an operating system, each application program would have to duplicate these instructions. On the PC-DOS disk, two "hidden files," **IBMDOS.COM** and **IBMBIO.COM**, provide the input/output instructions required by DOS and by application programs. They are called hidden files because they do not appear on the directory of files when you attempt to list the files contained on your DOS disk. IBMDOS.COM provides some common services like copying files, deleting files, searching directories, and accepting keystrokes from the keyboard. IBMBIO.COM contains additions and corrections to the basic I/O system routines that are built into your system on ROM chips. In MS-DOS, the hidden files are named **MSDOS.SYS** and **IO.SYS**.

Interpret and Execute Commands

The command processor part of DOS interprets and executes the commands that you enter and interprets and executes commands from application programs as well. Without an operating system, you would have no effective way to communicate with the hardware and direct its activities.

File Management

As a user, you are heavily involved with the file management role of DOS. For example, before files can be saved on a disk, you must prepare the disk to record files. Then you can save, rename, copy, or delete disk files. DOS provides a series of complex commands to allow both the user and application programs to manage the multitude of disk files that you create over a period of time. Subsequent chapters of this book are primarily devoted to the file management commands provided by DOS.

SAVING FILES WITH DOS

A file is a group of related records, where each **record** consists of a string of characters that can be entered and saved as part of that file. Records consist of either program instructions or data. Thus, files are categorized as either **program files** or **data files**.

Saving files on disk is a requirement of all operating systems. The files you create and work with in main memory are temporary. They only become permanent when you save them on disk. When you turn off the computer or otherwise lose power to main memory, all programs and data stored in main memory are destroyed. However, you can easily save files on disk and recall them when needed. DOS saves files when directed to by application programs. You can also save files by using the COPY command, which is covered in detail in Chapter 3.

Although it appears a fairly simple process, DOS goes through a number of important steps to save files. The following overview will help you understand this process.

(1) IBM and IBM-compatible floppy disk drives are double-sided, meaning data can be stored on both sides of the disk. The area on formatted disks used to record data is comprised of clusters that are in turn made up of **sectors** and **tracks**. A **cluster** is the smallest addressable location that can save data on a disk. The cluster size on a 360KB floppy is 1024 bytes, since a sector contains 512 bytes and two sectors (one on each side of a disk) make a cluster. Although the sector size remains constant, the size of the cluster may vary depending on the disk drive used and the number of read/write heads per track. On stacked hard disks, cluster sizes typically involve 4 sectors or 2048 bytes. Large cluster sizes improve performance of disk access when files are large, but may waste space when files are smaller than a cluster. By recording both sides of a disk before moving to another track, recording head movement is minimized and disk access speed is increased. When you issue a save command from an application program or directly from DOS, the operating system reads the directory portion of the disk to determine a suitable location to save the file.

(2) If the file being saved is a new file, DOS tries to locate enough unallocated space (clusters) to hold the file. If the file was previously saved on that disk, DOS replaces the old file with the contents of the file currently in main memory. Because DOS uses whatever clusters it can find to save the data, the clusters may not be adjoining. Files written in noncontiguous clusters are called **fragmented files**. The additional head movement caused by fragmentation may slow down the reading and writing of these files considerably.

(3) When a file has been saved, DOS updates the **File Allocation Table (FAT)** on the disk with the clusters required to permanently store the file. The FAT, covered later in this chapter, is used by DOS to keep track of which clusters contain data and which are available to be allocated to a new file. Then DOS updates the directory on the disk with the size of the file in bytes and the date/time the file was saved. For this reason, it is important to boot the operating system with the correct date and time. Depending on the type of disk, the directory allows up to 512 entries, including filenames and subdirectory names. Since subdirectories are normally applicable to hard disk systems, we discuss them in Chapter 7. The file information stored on a disk directory is as follows:

- filename and filename extension
- file attribute (hidden file, read-only file, and backup status)
- date and time of creation or last update
- starting cluster number in the FAT
- file size in bytes

BOOTING DOS

Before you can use an operating system, you must load it from a secondary storage device such as a disk to the main memory of the computer. Because not all of the operating system can be loaded at once, a process known as **booting** the system is used to load the controlling portion of the operating system.

There are two general methods of booting any operating system. One way is called a "**cold boot**" because the computer is turned off prior to booting the operating system. The second method is termed a "**warm boot**" because the computer is already warmed up and was previously in use, but needs to be rebooted. In both cases, DOS executes the same sets of commands in the boot process.

The major difference between the two ways of booting the system is in how the boot process is initiated. For a cold boot, turn on the computer to initiate the booting process. With the warm boot, you must enter a sequence of three keys simultaneously: the Control key, the Alternate key, and the Delete key (**Ctrl-Alt-Del**). It is advisable to use a warm boot whenever you can to minimize the possibility of damaging the electronic chips when the power is turned on.

To do a cold boot of DOS from a floppy disk, follow these steps:

(1) Place the disk containing DOS in disk drive A with the latch securely closed. Drive A is usually either the top drive or leftmost drive in microcomputers with two drives. Then turn on the power to both the monitor and the CPU.

(2) The microcomputer begins executing a small startup program stored in ROM that instructs it to run some predetermined diagnostic tests. These tests include checking the computer's RAM and keyboard interface to make sure they are functional. If there are any problems, the program displays an appropriate error message on your screen.

(3) If the computer passes the diagnostic checks, the program in ROM loads the two DOS hidden files contained on the DOS disk in Drive A. It also loads the file called COMMAND.COM into RAM from Drive A. The **COMMAND.COM** file contains many of the DOS commands you will use. It is the primary command interpreter, processor, and loader. Occasionally, another program requires some of the memory occupied by the COMMAND.COM file. When that program is finished, you will be prompted to insert the disk containing the COMMAND.COM file in the default drive so that it can be reloaded.

(4) At this point, DOS looks for an optional **CONFIG.SYS** file that can be used to specify some of the different ways your system can be configured, or customized. In addition, DOS accommodates more advanced users by looking for a file on Drive A called **AUTOEXEC.BAT**. If DOS finds this file, it executes it immediately. Sophisticated DOS users utilize an AUTOEXEC.BAT file to simplify the booting process by executing a common set of boot-related programs. The uses of these two files are explained in later chapters.

(5) If there is no AUTOEXEC.BAT file, the system asks you to enter the correct date and correct time, so that it can keep track of the time with its own internal clock. Once you have entered the date and time, DOS displays the version number of the operating system and prompts you to enter a command. The **default prompt** is A>, where the A represents the **default disk drive**, the one containing DOS. Figure 2-2 shows you how the screen might look if you entered a date of July 4, 1990 and a time of 1:45 p.m. The A> is the system prompt, which requests the next command; the underline character following it is the cursor. For illustrative purposes, the data you entered is shown in boldface.

Figure 2-2

Screen Display after Booting PC-DOS

```
Current date is Wed  7-04-1990
Enter new date (mm-dd-yy): 07-04-90
Current time is  13:45:12.68
Enter new time: 13:45


Microsoft(R) MS-DOS(R)  Version 3.30 Plus
             (C)Copyright Microsoft Corp 1981-1987
             (C)Copyright Zenith Data Systems 1987,1988

A>_
```

The system date is entered using month, day, and year in the form of mm-dd-yy or mm/dd/yy. You can use either a slash (/), a hyphen (-), or a period (.) to separate the date entries. The system time is entered using hours, minutes, seconds, and hundredths of seconds in the form of hh:mm:ss.xx. The use of seconds (ss) and hundredths of seconds (xx) is optional. Hours are always entered using the 24-hour system and range from 0 to 23. You can only use the colon (:) or a period (.) to separate time entries.

As with most operating systems, when you are finished entering a command or some data, like date or time, you need to press the Enter key to let the system know you are finished keying.

If at a later time you wanted to change either the system date or time, all you would need to do is enter a simple command. To change the date, enter **DATE** and press the Enter key. DOS displays the current date for you and asks you to enter a new date. The format of the new date is the same as with the boot process (mm-dd-yy). When entering the date, you do not have to enter leading zeros. For example, the month of February can be entered as 02 or just 2. Also, you do not have to enter the century. For instance, the year 1990 is simply entered as 90. If you change your mind and decide not to change the date, just press the Enter key.

To change the time, enter **TIME** and change it using the same format as the boot process. For all practical purposes, you only need to enter the hours and minutes (hh:mm). When entering hours, remember that hours are entered using the 24-hour system. For example, 8:15 p.m. is entered as 20:15.

As long as DOS keeps running, it keeps track of the time, automatically changing both the time and the date. Many systems today have a small battery and additional software to keep track of the date and time when the computer is turned off. When the system is booted, an AUTOEXEC.BAT file can contain the command to automatically set the system date and time from this battery-operated clock.

The boot process is identical for hard disk systems except for one major difference: DOS looks for the DOS boot files on the hard disk rather than a floppy disk drive.

FUNDAMENTAL DOS COMMAND CONCEPTS

Before learning specific DOS commands, you should understand the basic concepts common to the majority of DOS commands. Once learned, these concepts can be easily applied to all the individual commands. The concepts to be learned include the use of a default disk drive, standard device names used by DOS, the need for effective file naming conventions, the use of wildcard characters, and the basic types of DOS commands.

Default Disk Drive

Most computer systems have at least two disk drives. Applications on floppy disk systems generally require that the program disk be located in one drive and your data disk in another. If your computer has only one floppy disk, it will likely have a hard disk as well. A hard disk can easily contain both your application program(s) and your data. However, you would still need a floppy disk drive to allow you to load files from floppy disks to the hard disk or to make backup copies of your hard disk files on floppy disks.

When running DOS, the system needs to know what disk drive applies to the commands you enter. DOS employs the "default drive technique," a concept that is referred to often in this text. Fundamental to this concept is that you can specify which of the disk drives is to be the default. Whenever you enter a command that does not contain a disk drive specification, DOS substitutes the default drive for the missing one(s) in the command. By understanding the default drive concept when entering commands, you can save a significant number of keystrokes. Remember, you only need to designate a disk drive if it is other than the default drive.

To identify the different disk drives, DOS uses a coding scheme consisting of letters: A and B are used for floppy drives, C and D are used for hard disk drives. DOS establishes the initial default drive as the drive that was used to boot the system. You can easily change the default to another drive whenever you wish. When you booted DOS and saw the A> prompt, the A referred to the default disk drive. The > is the symbol used by DOS to identify the system prompt. This is the operating system's way of reminding you what drive is the current default drive.

To change the default drive, you need to enter a new disk drive letter followed by a colon. For example, the command B: entered at the A> prompt changes the default drive from A to B. Once entered, the default drive is the B drive (the second floppy drive) and the system prompt is displayed as B>.

Standard Device Names

As with most operating systems, DOS reserves some names to represent system devices. For example, when used in DOS commands, the reserved word **CON** represents the keyboard (console). The reserved words **LPT1** and **PRN** are used to designate the line printer. Printer designations are especially useful

when you wish to redirect output that would normally go to the screen to the printer for a hard copy listing. Because reserved words have specific meaning to DOS, you should never use them to name a file. Other reserved device names include AUX1 and COM1 (serial port 1 devices), COM2 (a serial port 2 device), LPT2 and LPT3 (parallel ports), and NUL (a nonexistent device).

File Naming Conventions

DOS uses the **full filename** to tell it where to search for a specified file. The full filename consists of four parts: the disk drive designator, the path, the filename, and the filename extension. Only the filename itself is required. The optional parts are shown in brackets as follows:

```
[d:][path]filename[.ext]
```

The first part of the full filename, **[d:]**, is used to specify the disk drive. To specify a drive, enter the drive letter followed by a colon. If you omit the **drive designator**, the default disk drive is substituted by DOS.

The next part, **[path]**, is the location of the subdirectory containing the file. Discussion of this parameter is postponed until Chapter 7, when we discuss in detail hard disk systems that typically use subdirectories.

DOS requires that a **filename** be at least one character in length, but it can be as long as eight characters. Filenames can be made up of numeric digits, alphabetic letters, and certain special characters. You should avoid using any of the special characters, except possibly the hyphen (-), which is used to make filenames more readable. But don't use spaces or punctuation such as periods, commas, colons, or semicolons.

The final part is an optional extension. The **extension** is comprised of the same set of characters that are valid for filenames, but is limited to three characters. If you enter an extension of more than 3 characters or a filename of more than 8 characters in length, DOS will ignore the extra characters. If an extension is used, it must be preceded by a period. Example: **A:JONES89.DOC**

The intended purpose of a filename extension is to indicate the category of each file. Using filename extensions is highly recommended to aid in keeping track of your files. For example, you should use .DOC, .WP, or .TXT for word

processing files and .WK1 or .WKS for worksheet files. Extensions can be very beneficial when used in conjunction with the wildcard characters discussed in the next section.

When creating filenames, you should always use meaningful titles to further classify the type of data contained in each file. You can code a great deal of information into your filenames. For example, a set of memos on the new bottling plant could be named BOTTLE1.DOC, BOTTLE2.DOC, and so on. Or, if the memo dates were critical, the set could be named BOTmmdd.DOC, where the mm represented the month created and the dd the day. Later you will see how wildcard characters can be used to select only the desired category of files. When sorted file directories are displayed, the memos appear listed together in chronological sequence. As the number of your saved files grows, the benefits of care and foresight in creating filenames should become more significant.

When you assign file extensions, you should abide by the standard extensions already established and commonly used. Some of these standard extensions are described in Figure 2-3.

Figure 2-3

Standard Filename Extensions

```
$$$ - scratch or temporary file
BAK - backup file
BAS - BASIC program file (needs compiling first)
BAT - executable batch file (file of DOS commands)
COM - executable command file (machine language), type in the
      filename and press the Return key
DAT - data file (listable via the TYPE command)
DBF - dBASE III, III PLUS, or IV file
DEF - program definition or setup file
DOC - documentation file, similar to DAT
EXE - executable file, similar to COM file, but must be allocated
      to a specific memory location
HLP - text file containing help instructions
MSG - message file, similar to DAT or DOC
OVL - overlay file (used by large programs)
PRN - printer file (can be modified prior to printing)
SYS - operating system file (like a device driver)
TXT - text file, similar to DAT, DOC and MSG
WKS - Lotus 1-2-3 worksheet file
```

The $$$ extension is normally used by various application programs to identify temporary work files that the program intends to delete prior to completion. If you see any of these files on a directory listing, it is an indication that a program terminated abnormally, which could be due to a power failure or a user-initiated break.

Wildcard Characters

Wildcard characters have been appropriately named because, like jokers in a card game, they can be used to represent different characters in DOS commands. The two wildcard characters used by DOS are the asterisk (*), representing a group of characters, and the question mark (?), representing only a single character.

Perhaps the best way to understand the use of wildcard characters is by example. If you wished to display a directory listing of all the files on Drive A that begin with the characters LTR and that have an extension of DOC, you could enter the following directory command: **DIR A:LTR*.DOC**. In this example, the asterisk substitutes for any group of characters, so that filenames LTRSMITH.DOC, LTR4.DOC, and LTRBILL3.DOC would all qualify to be displayed in the directory.

If you wished to display all the files on Drive A that only contain a single character following LTR and contain any filename extension, you could enter **DIR A:LTR?.***. In this example, the filenames LTR1.DOC, LTR2, and LTRX.TXT would be displayed in the directory.

Wildcard characters can also be used with optional filename extensions. For example:

```
DIR A:TEXT.*   (list all files on Drive A with a filename of TEXT)
DIR B:P*.T*    (list any filename on Drive B starting with P having an
               extension starting with T)
```

If you would like to try some examples using your DOS disk, enter the following three commands at the A> system prompt:

```
DIR            (to list all files on Drive A)
DIR A:*.EXE    (lists just those files with an EXE extension)
DIR A:S*.*     (lists just those files that begin with an S)
```

Figure 2-4 shows you what the display screen might look like after executing the last two commands.

Figure 2-4

Screen Display of Using Wildcards

```
A>DIR A:*.EXE

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

APPEND   EXE     5794   6-08-88   9:52a
ATTRIB   EXE    10656   4-07-88   7:42a
FIND     EXE     6483   6-07-88   3:15p
SORT     EXE     1946   4-07-88   7:42a
XCOPY    EXE    11216   4-07-88   7:44a
        5 File(s)    164864 bytes free

A>DIR A:S*.*

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

SORT     EXE     1946   4-07-88   7:42a
SYS      COM     5045   6-14-88   3:45p
        2 File(s)    164864 bytes free

A>
```

The DOS Directory Listing

When you get a **directory listing** on your screen, it displays more than just the filenames. The file size in bytes as well as a "date stamp" is also displayed for each file. The date stamp is the date and time that each file was last written to the disk. As the number of files saved becomes large, the correct date stamp becomes even more important. This is why it is important to enter the date and time correctly each time DOS is booted. Figure 2-5 on the next page shows you a sample directory listing of a DOS disk.

Figure 2-5
DOS Disk Directory

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

COMMAND  COM     23210    3-07-89   1:43p
ANSI     SYS      1651    3-07-89   1:43p
ASSIGN   COM      1509    3-07-89   1:43p
ATTRIB   EXE     15091    3-07-89   1:43p
BACKUP   COM      5577    3-07-89   1:43p
BASIC    COM     17792    3-07-89   1:43p
BASICA   COM     27520    3-07-89   1:43p
CHKDSK   COM      9435    3-07-89   1:43p
COMP     COM      3664    3-07-89   1:43p
DISKCOMP COM      4073    3-07-89   1:43p
DISKCOPY COM      4329    3-07-89   1:43p
EDLIN    COM      7261    3-07-89   1:43p
FDISK    COM      8173    3-07-89   1:43p
FIND     EXE      6403    3-07-89   1:43p
FORMAT   COM      9398    3-07-89   1:43p
GRAFTABL COM      1169    3-07-89   1:43p
GRAPHICS COM      3111    3-07-89   1:43p
JOIN     EXE     15971    3-07-89   1:43p
LABEL    COM      1826    3-07-89   1:43p
MODE     COM      5295    3-07-89   1:43p
PRINT    COM      8291    3-07-89   1:43p
RECOVER  COM      4050    3-07-89   1:43p
RESTORE  COM      5410    3-07-89   1:43p
SELECT   COM      2084    3-07-89   1:43p
SHARE    EXE      8304    3-07-89   1:43p
SORT     EXE      1664    3-07-89   1:43p
SUBST    EXE     16611    3-07-89   1:43p
TREE     COM      2831    3-07-89   1:43p
VDISK    SYS      3307    3-07-89   1:43p
            29 file(s)     77,420 bytes free
```

The directory can be displayed in three general ways. One way is to list all the files with all the information, one after another, until all files have been displayed. To produce this display, simply enter the **DIR** command at the system prompt. However, many times disks contain more files than can be displayed on the screen without scrolling. To get a directory listing of filenames one screen at a time, enter **DIR /P**. The slash (/) identifies a **command option** to DOS. The term "switch" is sometimes used instead of command option. With the DIR command, the P option "pauses" when the screen is full and waits for you to "Press any key to continue..".

The message "Press any key to continue" should not be taken too literally, however. Some keys, like the Shift, Alt, Caps Lock, Num Lock, Scroll Lock, Sys Reset, and PrtSc keys, are ignored by DOS and do not work.

Another way the directory can be displayed is with the /W (wide) option. This option is helpful if you do not need to have the file size and date stamp displayed. It displays only the filenames, listing them five to a line in "wide" format. To use this option, enter **DIR /W**. Figure 2-6 shows you what the wide format might look like.

Figure 2-6

Example of DIR with Wide (/W) Option

```
A>DIR /W

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

COMMAND  COM     ANSI     SYS     APPEND   EXE     ASSIGN   COM     ATTRIB   EXE
CHKDSK   COM     DEBUG    COM     DISKCOMP COM     EDLIN    COM     FIND     EXE
FORMAT   COM     LABEL    COM     MORE     COM     RECOVER  COM     SORT     EXE
SYS      COM     TREE     COM     XCOPY    EXE
       18 File(s)    164864 bytes free

A>
```

Internal vs. External Commands

DOS commands are either internal or external. **Internal commands** are included with the portion of DOS that is loaded into RAM when the system is booted. They are available when the boot process loads the COMMAND.COM file from the system disk. Because it is much faster to access commands from RAM, the designers of DOS have made the more commonly used commands internal.

Some of the more commonly used internal commands include DIR, TYPE, COPY, and DEL. The DIR command was discussed previously. The TYPE command is used to display the contents of a "listable" file on the screen. It is often used to display the contents of batch files. The COPY command is used to copy a file from one location to another, optionally giving the duplicate file a new name. It is also used to copy characters from the keyboard (CONsole) to some device like the printer or a disk file. The DEL command allows you to delete unwanted files from a disk. These internal commands and others are discussed in detail in Chapter 3.

Because the full set of DOS commands is too extensive to be completely loaded into RAM, a substantial number of commands, called **external commands**, reside on the DOS disk. This could be either a floppy disk or a hard disk. External commands include all the filenames shown in Figure 2-5 except for the COMMAND.COM file, which contains the internal commands. Later on you will see how you can create your own "external" commands, referred to as batch files.

DOS automatically translates all commands to uppercase characters, so you may enter DOS commands using either uppercase or lowercase letters. Furthermore, commands can be entered using a combination of both types. For example, the commands Dir, dir, and DIR are treated the same way by DOS. To help you remember this option, our text examples use both uppercase and lowercase letters.

DOS Versions

Figures in this chapter refer specifically to Version 3.x of DOS, where x refers to any one of the modification levels. Major releases of DOS generally accompany a major change or improvement in hardware design. The first version, DOS 1.x, was somewhat limited and is now considered obsolete. DOS 2.x

added hard disk capability, a necessity for most business applications. DOS 3.x added networking capability, better file management commands, and the ability to use the new 3 1/2 inch disks. Most colleges and universities use this version of DOS. The newest version, DOS 4.0, provides an easy-to-use file manager shell, direct support for hard disk exceeding 32 megabytes, and expanded memory support beyond 640KB. The DOS shell may prove a disadvantage to users because it occupies 256KB of RAM. By comparison, DOS 3.x occupies less than 40KB.

If you have version 2.x of DOS, it may be well worth a small fee and some restoring of files to upgrade to Version 3.3. The 3.x versions contain three useful additions:

ATTRIB — a command that allows you to protect a file by making it a read-only file (you cannot change a read-only file).

LABEL — a command that allows you to add, change, or delete a disk's volume label (its internal label).

VDISK.SYS — a device driver that enables you to use part of RAM as a very fast disk drive (called RAMDRIVE.SYS in MS-DOS).

FORMATTING DISKS WITH DOS

To save files on disk, the disk must be properly prepared to accept the data to be saved. When you purchase a new blank disk, it is a "generic" disk that can be used with many different microcomputers, including a variety of different operating systems. Therefore, each blank disk (floppy or hard disk) must be customized according to the requirements of DOS. This customizing process is called **formatting** and includes the operations described below.

(1) It creates addressable areas of the disk made up of clusters where data can be stored. Because each disk drive and operating system has its own addressing scheme, this activity is mandatory prior to saving files. With DOS 2.x and later versions, each of the 40 tracks on one side of a 5 1/4 inch floppy disk is divided into 9 sectors, for a total of 360 sectors. Each sector is designed to hold 512 bytes of data. A cluster is a corresponding track and sector on both sides of a disk.

(2) It then checks every recording spot on the new disk for damage and determines which clusters, if any, are not acceptable for storing data.

(3) Additionally, it creates a directory area and a File Allocation Table (FAT) on the disk that can be used by DOS to keep track of the files saved. The cluster numbers of any unacceptable areas of the disk are recorded in the FAT. As files are later saved on the disk, the occupied (allocated) cluster numbers are also noted in the FAT.

The **FORMAT** command is used to prepare a disk for use by DOS. When you format a disk, the system displays the status of your disk, including the number of any bad sectors it finds. Here is an example of the information displayed after the command FORMAT was entered and the disk being formatted had some unacceptable sectors:

```
362496 bytes total disk space
  8192 bytes in bad sectors
354304 bytes available on disk
```

A bad sector is a small portion of the disk that DOS determines is damaged. These sectors are noted on the disk's File Allocation Table. DOS does not allow data to be stored in these sectors. It is a good practice to replace disks containing a high percentage of bad sectors with good disks, as the cost of losing data is much higher than the cost of a new floppy disk.

The FORMAT command offers several helpful options. The system or /S option tells DOS to format the disk to include the DOS boot programs. When DOS formats a disk with the /S option, it actually writes three files on the disk. Two of these are hidden, which means you cannot see them when you display a directory of that disk. The third file is the COMMAND.COM file that contains all the internal DOS commands. To use the /S option to create bootable disks, enter **FORMAT /S**.

Another helpful option of the FORMAT command is the /V option, which allows you to name your disk with a volume label. The volume label acts as an internal file label. This option is especially helpful when working with numerous disks, because it helps you keep track of your disks. Volume labels can be up to 11 characters long and you can create a different volume label for each disk. The label can be viewed on the screen without having to remove the disk from the drive, a requirement with external labels. To use the /V option, enter **FORMAT /V**. To use it with the system option, enter **FORMAT /S/V** or **FORMAT /V/S**.

Beginning with Version 3.2 of DOS, the FORMAT command no longer assumes you want to format the default drive if you enter the command without any parameters. This precaution keeps you from attempting to reformat your DOS disk accidentally.

Review Questions

2

1. What are the main purposes of an operating system?
2. What is meant by the term "booting the system"?
3. What are the hidden files on a DOS disk?
4. Why is it advantageous to write on both sides of a floppy disk before moving to another track?
5. Why is it important to boot your system with the correct date and time?
6. What is the function of the File Allocation Table?
7. What is the difference between a warm boot and a cold boot?
8. How do you activate a warm boot?
9. What is the purpose of the COMMAND.COM file?
10. What is the purpose of the AUTOEXEC.BAT file?
11. How can you change the system date once the system is booted?
12. How would you enter a time of 2:35 p.m. when prompted to do so?
13. What does the command DIR B:/W do when executed?
14. What is the purpose of the FORMAT command?
15. What do the standard device names CON and PRN represent?
16. What is a filename extension and how is it identified by DOS?
17. What kind of file would likely have an extension of .$$$?
18. What are some common extensions for listable files?
19. What are some common extensions for executable files?
20. Why would you want to use the /S option with FORMAT?

Chapter 2 Lab Exercises

PLEASE NOTE:

This exercise assumes that the system is already turned on from a prior class demonstration. If not, boot the system according to what you learned in this chapter. These exercises were developed for computers with two floppy disk drives. Drive A will hold DOS and Drive B will hold your working data disk.

If you are working with a hard disk system with only one floppy disk, the following modification must be made to your exercises:

The DOS commands should already be loaded on your hard disk (C:), so you should turn on the power to the CPU without having any disk in the floppy drive. Change any reference to Drive A in the exercises to drive C. Now floppy disk Drive A is used to hold your working data disk. Therefore, you must also change any reference to Drive B in the exercises to Drive A.

1. Warm boot:
   - With the DOS disk in Drive A, use **Ctrl-Alt-Del** to boot the system.
   - Enter the date in mm-dd-yy format (i.e., 9-15-90).
   - Enter the time in hh:mm format (i.e., 13:07).
   - Don't forget to press the Enter key (or Return) to cause the computer to act on your data.

2. Format a data disk to contain the DOS system files:
   - Enter `FORMAT b:/s/v`
   - Insert a blank disk in Drive B and press Enter.
   - Enter your name (up to 11 characters with no special characters) when prompted to do so by the system.
   - Enter n when prompted to format another disk.

Figure 2-7 shows you what the screen should look like when you finish formatting your blank disk.

Figure 2-7

Screen Display of the FORMAT Command

```
A>FORMAT B:/S/V

               FORMAT version 3.30.05
Copyright(C) 1988, Zenith Data Systems Corporation

Insert new disk in drive B:
and press RETURN when ready

Format complete
System transferred

Volume label (11 characters, RETURN for none)? YOUR NAME


    362496 bytes total disk space
     81920 bytes used by system
    280576 bytes available on disk

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

3. Create a file from the keyboard:
   - Enter `COPY CON B:READ.ME` (or `copy con b:read.me`).
   - On the lines that follow, enter the following text, pressing the Enter key at the end of each line.

When entering DOS commands, the commands and parameters must be separated by delimiters. Delimiters are normally either a space, a comma, or a semicolon. They can be used interchangeably within any command (i.e., COPY A:oldfile,B:).

   - When done entering lines of text, press function key **F6** (to tell DOS you are done with the copy operation) and press Enter. When pressed, the F-6 key displays a Ctrl-Z (^Z) on the monitor. The text you just keyed will be stored on the disk in Drive B with the filename of READ.ME.

Figure 2-8 shows you the screen after you have completed the COPY CON command.

Figure 2-8

Screen Display of COPY CON

```
A>COPY CON B:READ.ME
When entering DOS commands, the commands and parameters
must be separated by delimiters.  Delimiters are normally
either a space, a comma, or a semicolon.  They can be used
interchangeably within any command (i.e., COPY A:oldfile,b:).
^Z
        1 File(s) copied

A>
```

4. Display a disk file on the monitor:
   - Enter `DIR B:` (to verify that your file was stored). This listing should include READ.ME plus the COMMAND.COM file placed there during formatting.
   - Now enter `TYPE B:READ.ME` to see a display of your text file on the monitor.

Figure 2-9 shows you what the screen should look like when you finish this portion of the exercise.

Figure 2-9

Screen Display of DIR and TYPE

```
A>DIR B:

 Volume in drive B is YOUR NAME
 Directory of  B:\

COMMAND  COM    25532  10-31-88   2:54p
READ     ME       240  11-15-89  11:24a
        2 File(s)    279552 bytes free

A>TYPE B:READ.ME
When entering DOS commands, the commands and parameters
must be separated by delimiters.  Delimiters are normally
either a space, a comma, or a semicolon.  They can be used
interchangeable within any command (i.e., Copy A:oldfile,b:).

A>
```

5. COPY a file to a disk:
   - Enter `COPY B:READ.ME B:READ.BAK`. This creates a duplicate copy of your file, but with a different filename (the extension is .BAK rather than .ME).
   - Now enter `DIR B:` (to get a directory listing of B). You should have three files listed on your data disk: COMMAND.COM, READ.ME, and READ.BAK.

Figure 2-10 shows the display screen at this point.

Figure 2-10

Screen Display of COPY and DIR

```
A>COPY B:READ.ME B:READ.BAK
        1 File(s) copied

A>DIR B:

 Volume in drive B is YOUR NAME
 Directory of  B:\

COMMAND  COM     25532  10-31-88   2:54p
READ     ME        240  11-15-89  11:24a
READ     BAK       240  11-15-89  11:24a
        3 File(s)    278528 bytes free

A>
```

6. DELete a file from a disk:
   - Enter `DEL B:READ.BAK`.
   - Now enter `DIR B:` (to verify that the file was deleted).

   Your screen should contain data similar to that shown in Figure 2-11.

Figure 2-11

Screen Display of DEL and DIR

```
A>DEL B:READ.BAK

A>DIR B:

 Volume in drive B is YOUR NAME
 Directory of  B:\

COMMAND  COM     25532  10-31-88   2:54p
READ     ME        240  11-15-89  11:24a
        2 File(s)    279552 bytes free

A>
```

7. Change the default disk drive:
   - Enter `B:` (to switch default to Drive B).
   - Enter `DIR` (to get a directory of the default disk, Drive B).
   - Enter `DIR A:/W` (to get a wide listing of Drive A).
   - Enter `A:` (to change the default drive back to Drive A).

Figure 2-12 shows the display screen when this exercise is over. Some of the data you may still have on your screen is shown in this figure. Don't be concerned if several of the top lines have scrolled off the top of your screen.

Figure 2-12

Screen Display of DIR

```
A>B:

B>DIR

 Volume in drive B is YOUR NAME
 Directory of  B:\

COMMAND  COM     25532  10-31-88   2:54p
READ     ME        240  11-15-89  11:24a
        2 File(s)    279552 bytes free

B>DIR A:/W

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

COMMAND  COM    ANSI     SYS    APPEND   EXE    ASSIGN   COM    ATTRIB   EXE
CHKDSK   COM    DEBUG    COM    DISKCOMP COM    DISKCOPY COM    EDLIN    COM
FIND     EXE    FORMAT   COM    LABEL    COM    MORE     COM    PRINT    COM
RECOVER  COM    SORT     EXE    SYS      COM    TREE     COM
       19 File(s)    158720 bytes free

B>A:

A>
```

8. Bonus exercise:
   - If the system date is not the correct current date, use the **DATE** command to enter the current date. Otherwise, change the system date to yesterday's date. Use the **TIME** command to make sure that the system time is correct.
   - Use the `COPY CON` command to create a new file called **B:EX2.TXT**. It should contain the following 5 lines of text, entered one line at a time. After entering the text, press function key **F6**.

The Lotus 1-2-3 software package, as the name implies, has three logical and integrated parts: spreadsheets, graphics, and data management. Integration means that you do not have to leave the spreadsheet portion, for instance, to get to the graphing or data management portions.

- Use the **TYPE** command to display this newly created file.
- Copy this file to a new file (on Drive B) with the same filename, but with an extension of **.DOC**. Use the **DIR** command to confirm the copy. The COPY command preserves the date stamp of the original file being copied.
- Notice what happens if you enter a command that uses wildcard characters incorrectly, such as `DIR B:*X*.*`. Modify this command to get a directory listing of all files on Drive B with an **X** in the second position of the filename.
- Using DOS wildcard characters, delete all files on Drive B with a filename of **EX2** and any extension. Before you enter this command, however, be sure it is correct, or you may delete more files than you intended. Two files should be deleted. Display a directory listing to check the status of files on Drive B. Does it look like Figure 2-11?

This concludes Chapter 2 Lab Exercises. When you are finished, remove your disks.

Chapter

3

Internal Floppy Disk Commands

DOS INTERNAL COMMANDS

- ☐ BREAK (Control Break) Command
- CLS (Clear Screen) Command
- COPY Command
- DATE Command
- DEL (Delete) Command
- DIR (Directory) Command
- PROMPT Command
- RENAME Command
- TIME Command
- TYPE Command
- VER (Version) Command
- VERIFY Command
- VOL (Display Volume Label) Command

Chapter

3

Internal Floppy Disk Commands

The DOS commands covered in this chapter are all internal commands. External floppy disk commands are covered in Chapter 4.

This chapter covers 13 internal DOS commands likely to be used with floppy disk systems. Many of these commands are also applicable to hard disk systems, but do not necessarily require hard disks. More advanced DOS internal commands, those used primarily with hard disk systems, are covered in later chapters.

Our discussion of each command includes the following items:

- The general format of the command.
- An explanation of the command parameters.
- Helpful options when using the command.
- Some examples of usage.

In addition to the command name itself, DOS commands can contain various **parameters** to indicate the target of the command, as well as **options** to indicate how the command is to be executed. Options are often referred to as switches. Command parameters and options that apply primarily to hard disk systems are bypassed for now.

If the format of a command has a parameter or option enclosed in brackets, that part of the command is not always required. Do not include the brackets as part of the command when entering it. Command options are always preceded by a **slash** (/). We have intentionally omitted command options that are seldom used.

A **delimiter** is a special character used to define the end of a word or specific portion of DOS commands. Commands and associated parameters must be separated by a delimiter. DOS uses a variety of delimiters, including a space, a comma, a semicolon, an equals sign, and the Tab key. The examples in this text normally use a space as a delimiter. As mentioned in Chapter 2, you can enter commands using any combination of uppercase or lowercase characters.

You can stop execution before normal completion in a variety of ways. You can abort DOS commands while they are running by entering **Ctrl-C** (or **Ctrl-Break**). When commands display a large amount of output on the screen, **Ctrl-S** temporarily suspends the display. Pressing any key continues the display process. When software instructions direct you to "Press any key" this normally refers to any alphabetic (A–Z) or numeric (0–9) key, or the space bar. It does not refer to any of the special or control keys.

A command you enter may be rejected by DOS, spurring an error message such as "**Bad Command or Filename**." You see this message if the command entered was not spelled correctly or not found, or if a filename entered was not on the disk specified. If this error message appears, simply retype the command correctly.

Because the space between the read/write heads and the surface of the disks is incredibly small, any movement of the disk drive when the disk is operating can be very destructive. If you try to read data from a defective disk, DOS displays an error message. If you attempt to read a disk and no disk is in the designated drive or the drive latch is open, you get a "not ready" error message. Likewise, if track zero on a disk is bad and cannot store the FAT and directory, or the type of disk is not correct (like a high-density floppy disk in a low-density drive), DOS displays an error message.

These error messages relate to read or write operations on a disk. Whenever you get an error message on a disk operation, DOS also displays the following response message: **Abort, Ignore, Retry?**

When you get this response message, and cannot correct the problem, you can enter **A** to end (abort) the command that requested the disk operation. It is possible to simply ignore the problem by entering an **I**, but this often causes a variety of additional problems and should be avoided. If you can correct the problem, for example a disk that is not ready or a write-protect error, fix the problem and enter **R** to retry.

Beginning with DOS 3.3, the previous message has been revised slightly. Now, the response message displays as: **Abort, Ignore, Retry, Fail?**

These responses mean the same as in earlier DOS versions. The response of **F** means to fail (or abort) the disk operation, but do not abort the command that requested the disk operation. Usually you will want to retry the disk operation. If the same error occurs again, then enter **A** to terminate the command.

Appendix C includes a listing of the more common DOS error messages with suggestions for corrective action. For a more complete description of DOS commands and error messages, consult the DOS manual.

DOS INTERNAL COMMANDS

Internal commands are contained in the COMMAND.COM file that is loaded to main memory when DOS is booted. Therefore, you do not need to have the system disk in a disk drive when attempting to execute any internal command. The internal commands, covered alphabetically in this chapter, include:

1. **BREAK** — turns the system break mode on or off
2. **CLS** — clears the screen of all data
3. **COPY** — makes a copy of a disk file
4. **DATE** — displays and/or changes the system date
5. **DEL** — deletes a file from disk
6. **DIR** — displays filenames from a disk
7. **PROMPT** — changes the system prompt
8. **RENAME** — changes the name of an existing file
9. **TIME** — displays and/or changes the system time
10. **TYPE** — displays the contents of a listable disk file
11. **VER** — displays the version number of DOS
12. **VERIFY** — turns the verify mode on or off
13. **VOL** — displays a disk volume label

BREAK (Control Break) Command

Format: `BREAK [ON | OFF]`

The **BREAK** command sets a switch in DOS that determines when the system should check for Ctrl-Break (or Ctrl-C) from the keyboard. With Break set off (the default setting), DOS only checks for Ctrl-Break during input/output operations to the keyboard, screen, or printer. To direct DOS to also check for Ctrl-Break during disk I/O operations, you must set Break on by executing the BREAK ON command. This option is useful when you run programs that are difficult to stop. If you enter BREAK with no parameters, the current status of Break is displayed on the screen. This command is often included in the CONFIG.SYS file (covered in Chapter 9).

EXAMPLES OF USAGE:

`A> Break on`
(directs DOS to "break out" of a program as soon as Ctrl-Break is pressed)

`A> BREAK OFF`
(directs DOS to check only for Ctrl-Break during the input/output operations)

`A> break`
(displays the current status of Break)

CLS (Clear Screen) Command

Format: `CLS`

The **CLS** command clears the monitor (display screen) to all blanks.

COPY Command

Format: `COPY [d:]filename[.ext] [d:][filename[.ext]][/V]`,
where the first filename is the source file and the second filename (optional) is the target file (the new file being created).

The **COPY** command lets you make copies of disk files to a previously formatted disk. It facilitates making backup or working copies of files without destroying existing files. Any files on the target disk with the same name as the target file are replaced by the contents of the source file after the COPY command executes.

If you omit the disk device designator (d:), DOS substitutes the default device. If you omit the optional target filename, the system uses the same filename as the source file. Be sure to include appropriate filename extensions. To verify that sectors written on the target disk were recorded properly, you can use the /V option. This option slows down the copy process, but it may be worthwhile if you are experiencing disk problems. For example:

```
COPY A:MYFILE.TXT B:MYFILE.BAK /V
```

Concatenation, the combining of two or more files, can be performed with the COPY command. Use COPY with the plus (+) symbol between multiple source filenames to combine two or more source files into a single, new target file. For example:

```
COPY A:FILEA.DOC+A:FILEB.DOC B:NEWFILE.DOC
```

Files combined with the other commands may have two **end-of-file** (or **EOF**) marks, displayed as **Ctrl-Z**. To make the file more useful, you can edit the file with the DOS line editor (EDLIN is covered in Chapter 6), removing the middle EOF mark.

You can copy a group of files with a single command by using wildcard characters with the COPY command in filenames and filename extensions. For example, to copy all files on Drive A with an extension of .DOC to Drive B, enter `COPY A:*.DOC B:`.

If you specify the source file as **CON** (for CONsole keyboard), the target file contains characters entered from the keyboard. Type characters as you would from a typewriter, pressing the Enter key at the end of each line. Lines are limited to 127 characters each. DOS uses the Ctrl-Z to mark the end of a text-only file. To stop recording characters and insert the Ctrl-Z character, press the F6 function key, <F6>, followed by the Enter key. For example:

```
COPY CON A:KBFILE.TXT  (followed by lines of text and <F6>)
```

There is a potential problem with using COPY CON. In the last example, if A:KBFILE were a preexisting file, it would be replaced by the lines entered from the keyboard. In addition, COPY CON can be used to type a "quick and dirty" small file of text, but it is not appropriate for larger files. DOS provides its own line editor called EDLIN (see Chapter 6) for larger text files.

EXAMPLES OF USAGE:

```
A> COPY *.* B:
```
(copies all files on the default disk, Drive A, to the disk in Drive B, without renaming files)

```
A> copy B:*.DOC
```
(copies all files on Drive B with an extension of .DOC to the default disk)

```
A> Copy filea.doc b:filea.bak
```
(copies FILEA.DOC on Drive A to Drive B, renaming it FILEA.BAK)

```
A> copy con B:read.me
```
(creates a file on Drive B named READ.ME consisting of data entered from console)

```
A> COPY A:FILE*.DOC B:
```
(copies all files on Drive A with a .DOC extension that begin with FILE to Drive B)

```
A> copy a.txt+b.txt c.txt
```
(creates a new file, C.TXT, as a combination of A.TXT and B.TXT)

```
A> copy filea.doc B:/V
```
(copies filea.doc on Drive A to Drive B and directs DOS to verify the copy)

```
A> COPY FILEA.DOC PRN
```
(copies an ASCII file to the printer)

```
A> copy con prn
```
(lets you use the keyboard like a typewriter)

```
A> COPY  B:MYDATA.TXT  COM1
```
(sends an ASCII file through the communications port, COM 1)

DATE Command

Format: `DATE [mm-dd-yy]`

The **DATE** command allows you to change the system date. If you specify a new date when you enter the command, it is changed immediately. If you omit this optional parameter, the system displays the current date and prompts you to enter a new date. Press the Enter key if you do not wish to change it. If you enter an invalid date, you are prompted to reenter a correct date.

EXAMPLES OF USAGE:

```
A> DATE 3/4/90
```
(changes the system date to March 4, 1990)
```
A> date 03-04-90
```
(also changes the system date to March 4, 1990)
```
A> Date
```
(displays the current date and prompts you to change it)

DEL (Delete) Command

Format: `DEL [d:]filename[.ext]`

The **DEL** command deletes the specified disk file. If the drive designator is not specified, the default drive is assumed. You can use wildcard characters (* and ?) in the filename and extension, but do so with caution, as multiple files can quickly be deleted with a single command. If you use *.* to specify the file, all files on the designated disk will be deleted. When you attempt to delete all files on a disk, the DEL command gives you some measure of protection against eliminating files by mistake. It pauses to ask you if you are sure. You are not allowed to delete read-only files without first changing the status with the ATTRIB command, which is discussed in Chapter 4. Also, you cannot delete DOS hidden files.

The term "delete" may be a little misleading, since files are not physically deleted from a disk file. The DEL command merely causes the file's entry on the disk directory to be flagged as removed, thereby allowing other data to be written over the "deleted" file. Technically, the command replaces the first character of the filename on the directory with an ASCII 229 character, which often displays as a question mark.

Since the **ERASE** command is identical to the DEL command, you may use either command. ERASE is treated as just another spelling of DEL.

EXAMPLES OF USAGE:

```
A> DEL a:Memo.txt
```
(deletes file MEMO.TXT from Drive A)
```
A> DEL memo.txt
```
(deletes file MEMO.TXT from the default drive)
```
A> del *.txt
```
(deletes all files on the default drive with a filename extension of .TXT)

DIR (Directory) Command

Format: `DIR [d:][filename[.ext]] [/P][/W]`

The **DIR** command displays a directory, or listing, of the files on a specified disk. The information provided in the listing includes the volume identification, the name of each file, the size in bytes of each file, the date and time each file was last written to, and the amount of free space left on the disk. If you do not designate a disk drive, DOS uses the default drive. If you specify a filename, the directory is limited to only that name. Since the filename can contain wildcard characters, the directory can be limited to a specific group of files.

Subdirectory names (to be covered later) are also displayed on the directory and are clearly identified with <DIR> in the file size field. Entries for the hidden system files are never listed, even when present.

Use the **/P** option to cause the computer to pause during the display of the directory when the screen is full. It continues displaying again after you press any key to signal you are ready to continue. Figure 3-1 shows what the screen might look like using the pause option.

Figure 3-1

Screen Display of DIR with Pause (/P) Option

```
COMMAND  COM     25532  10-31-88   2:54p
ANSI     SYS      1647   4-07-88   9:58a
APPEND   EXE      5794   6-08-88   9:52a
ASSIGN   COM      1530   5-11-88   2:10p
ATTRIB   EXE     10656   4-07-88   7:42a
CHKDSK   COM     11003   6-09-88   9:34a
COMP     COM      3759   9-10-87   8:40p
DEBUG    COM     15928   4-26-88  10:15a
DISKCOMP COM      6236   6-23-88  10:02a
DISKCOPY COM      6762   6-23-88  10:02a
EDLIN    COM      7495   6-02-88   2:06p
FASTOPEN EXE      3888   4-06-88   2:36p
FIND     EXE      6483   6-07-88   3:15p
FORMAT   COM     12206   8-19-88   4:02p
LABEL    COM      2346   4-07-88   7:43a
MODE     COM     15440  11-16-88   3:50p
MORE     COM       282   4-07-88   7:43a
PRINT    COM      9491   6-08-88   9:44a
RECOVER  COM      5262   5-06-88   4:01p
REPLACE  EXE     13234   4-07-88   7:44a
SORT     EXE      1946   4-07-88   7:42a
SUBST    EXE     10552   4-06-88   3:01p
SYS      COM      5045   6-14-88   3:45p
Strike a key when ready . . .
```

Use the **/W** option to display the directory in wide format, in which only the filenames are displayed across the screen, five files on a line. You can use the /W option to save display time and space.

EXAMPLES OF USAGE:

```
A> Dir
```
(displays a directory of all files on the default disk drive)

```
A> DIR B:
```
(displays a directory of all files in Drive B)

```
A> dir  a:dog*.*
```
(displays directory of Drive A, of only those files with filenames that begin with DOG)

```
A> Dir b:/p
```
(displays the Drive B directory, pausing whenever the screen fills up)
```
A> dir /w
```
(displays filenames on the default drive in wide format)
```
A> DIR B:/W/P
```
(displays filenames on Drive B in wide format, pausing after each screen)

PROMPT Command

Format: `PROMPT [text]`, where text is a variable-length string of characters. Text may contain special strings in the form of $c, where c represents one of the following character codes:

t — the system time
d — the system date
n — the default (current) drive
g — the > character
_ — an underline (used to skip to a new line)
p — display the current directory (Chapter 7)
e — send an ESCape character (Chapter 9)

The **PROMPT** command allows you to change the system prompt from the default (A>) to whatever you want to make it. Placing a PROMPT command in your AUTOEXEC.BAT file ensures that your customized prompt appears automatically each time you boot. If you enter PROMPT with no text, the system reverts back to the default prompt. For more information on PROMPT, consult the DOS manual. Figure 3-2 shows the effect on the screen of executing the last three examples of the PROMPT command shown below.

EXAMPLES OF USAGE:

```
A> PROMPT Command?
```
(changes the system prompt from A> to Command?)
```
A> prompt DATE = $d
```
(changes system prompt to display DATE = followed by the system date)

```
A> prompt Hi Fred $_$n$g
```
(displays Hi Fred on the first system prompt line followed by A>)
```
A> PROMPT
```
(returns to the normal system prompt, A>)

Figure 3-2

Screen Display of Changing Prompts

```
A>prompt DATE = $d

DATE = Tue 11-28-1989   prompt Hi Fred $_$n$g

Hi Fred
A>PROMPT

A>
```

RENAME Command

Format: `RENAME [d:]filename[.ext] filename[.ext]`

The **RENAME** command changes the name of the file specified in the first parameter to the filename and extension given in the second parameter. A drive designator is not allowed in the second parameter and is rejected if entered. RENAME gives you an easy way to make disguised copies of important files. For example, a spreadsheet file called BUDGET.WK1 could be renamed WORK.EXE.

A shortened and commonly used version of the RENAME command is REN. Wildcard characters can be used in the RENAME command, but can lead to problems. You will get an error message if a duplicate filename is created. The command `REN B:TEST.* NEWNAME` could easily lead to duplicate filenames being generated.

EXAMPLES OF USAGE:

```
A> RENAME b:ltr1.doc ltr1.bak
```
(renames LTR1.DOC on Drive B to LTR1.BAK)

```
A> ren Ltr1.doc Ltr1.bak
```
(renames LTR1.DOC on Drive A to LTR1.BAK)

```
A> Rename ltr1.doc *.bak
```
(renames LTR1.DOC on Drive A to LTR1.BAK—use of wildcard characters can save keystrokes)

```
A> REN *.TXT *.DOC
```
(renames all files with a .TXT extension to an extension of .DOC)

```
A> REN   B:*.DOC   B:*.BAK
```
(the target disk drive designator causes an error condition)

TIME Command

Format: `TIME [hh:mm[:ss]]`

The **TIME** command allows you to change the system time. It is important to keep the correct date and time on the system because it is recorded in the directory information of each file you save. If you omit the optional parameters, the current system time is displayed and you are prompted to change it. To leave the time as is, just press the Enter key. If you enter an invalid time, the system prompts you to reenter a new time.

EXAMPLES OF USAGE:

```
A> TIME 8:30
```
(changes the system time to 8:30 a.m.)

```
A> Time 14:15:35
```
(changes the time to 2:15 p.m. and 35 seconds)

```
A> time
```
(displays the current time and prompts you to enter a new time)

```
A> time 11.55.30
```
(changes the time to 11:55 a.m. and 30 seconds)

TYPE Command

Format: `TYPE [d:]filename[.ext]`

The **TYPE** command is used to display the contents of a "listable" file on the standard output device, normally the monitor. It does not alter files. This command should only be used for ASCII text files, not files that end with an extension of .EXE or .COM. Wildcard characters are not allowed. You can use TYPE in combination with the Print Screen capability of DOS to display the contents of a file on the printer. You can also redirect the output to a file or to a printer. Redirection of output is covered in Chapter 5.

EXAMPLES OF USAGE:

```
A> TYPE B:READ.ME
   (displays contents of READ.ME, stored on Drive B, on the monitor)
A> type read.me
   (displays READ.ME, stored on the default drive, on the monitor)
A> TYPE AUTOEXEC.BAT >PRN
   (The contents of AUTOEXEC.BAT are redirected to the printer)
```

VER (Version) Command

Format: `VER`

The **VER** command displays the DOS version number being used on the screen (i.e., IBM Personal Computer DOS Version 3.10).

VERIFY Command

Format: `VERIFY [ON | OFF]`

The **VERIFY** command sets the Verify mode in DOS. When the Verify mode is on, each character that is written to the disk is read to verify that it has been correctly recorded. With Verify set off (the default), DOS does not perform an additional read and compare operation each time data is written. To direct DOS to verify each write operation, you must set Verify on with the VERIFY command. Because of the extra time required to perform verification, it requires approximately twice the time to write data to a disk with Verify set on. Write verification is not recommended unless you are experiencing problems with your disks. If you enter VERIFY with no parameters, the current status of Verify is displayed on the screen.

EXAMPLES OF USAGE:

```
A> VERIFY on
```
(directs DOS to verify each disk write)
```
A> verify off
```
(sets the Verify mode off)
```
A> Verify
```
(displays the current status of the Verify mode)

VOL (Display Volume Label) Command

Format: `VOL [d:]`

The **VOL** command is used to display the internal disk volume label of the designated drive, so you don't have to physically remove the disk from the drive to identify it. If you do not specify a drive, the default drive is assumed. The volume label can be created with the FORMAT command and later changed with the LABEL command. Both of these external commands are covered in the next chapter.

EXAMPLES OF USAGE:

```
A> vol B:
```
(displays the volume label recorded on the disk in Drive B)
```
A> VOL
```
(displays the volume label of Drive A)

Review Questions

3

1. What is the function of brackets in this text for describing command formats?
2. What is the function of the slash in this text for describing command formats?
3. What is a delimiter and why is it required?
4. What does the message "Bad Command or Filename" mean?
5. Where are internal commands stored on a permanent basis?
6. When does DOS normally check for a Ctrl-Break (Ctrl-C) entered from the keyboard?
7. What DOS command blanks out the display screen?
8. What is the benefit of using wildcard characters in the COPY command?
9. What command lets you use the keyboard and printer as a typewriter?
10. What command removes the filename from the disk directory, but does not physically remove the file from the disk?
11. What option allows you to get a directory listing on the screen with filenames displayed in multiple columns?
12. What method is used to get a directory listing of a specific group of files?
13. What command is used to change the system prompt to display the system time?
14. How would you cause the system to display a customized system prompt automatically every time the system is booted?
15. What happens if you include a disk drive designator on both parameters of the RENAME command?

16. What do you have to enter to change the system time to 4:00 p.m.?
17. What type of file is considered listable with TYPE?
18. What command is used to verify the DOS version being used?
19. What command is used to verify that data written to disk is correct?
20. What command is used to display the internal label on a disk without getting a directory listing?

Chapter 3 Lab Exercises

1. Boot DOS (Drive A) and insert your data disk in Drive B. Then enter `DIR B:` to refresh your memory of the files on Drive B. You should see two files displayed that were previously created in the Chapter 2 Lab Exercise: COMMAND.COM and READ.ME.

2. Enter `VERIFY` to see whether or not DOS verifies each "write" with an automatic "verification read." The responding message should indicate that the Verify mode is set off. Test the effect by entering the following: `COPY A:COMMAND.COM B:TEST.1` and making a note of how long it takes to copy (i.e., 4 seconds). Enter `VERIFY ON` to set on the Verify mode. Enter `COPY A:COMMAND.COM B:TEST.2` and see if the copy with the verification takes a substantially longer time (i.e., 7 seconds). Then enter `VERIFY OFF`.

3. Enter `DIR B:` to get a current directory listing of your data disk. Enter `RENAME B:TEST.2 TEST.3` to change the filename of TEST.2 on your data disk to TEST.3. Enter `DIR B:` to verify the name change. To simplify the commands in this step, you could have first changed the default disk to Drive B and then omitted all reference to Drive B (B:).

4. Enter `VOL B:` to see if your data disk has an internal volume label. If you did not use the /V option when you formatted the disk (or have not yet used the LABEL command), your data disk should not contain a volume label.

5. Enter VER to see what DOS version you have (i.e., Ver. 3.2).

6. Enter DATE and follow the system prompts to change the current system date. Then do the same for TIME. Typically, you only need to enter hh:mm (i.e., 13:45) for the time, ignoring seconds.

7. Create a combined file: COPY B:READ.ME+B:READ.ME B:TEST.4 This creates a new file (TEST.4 on Drive B) as the sum, or concatenation, of two files on Drive B (READ.ME and READ.ME). Enter DIR B: to see if the combined new file is twice the size of READ.ME. In addition, enter TYPE B:TEST.4 to verify that the text in READ.ME was duplicated correctly. Your display screen should now look something like Figure 3-3.

Figure 3-3

Screen Display of DIR and TYPE

```
A>DIR B:

 Volume in drive B is YOUR NAME
 Directory of  B:\

COMMAND  COM     25532  10-31-88   2:54p
READ     ME        240  11-15-89  11:24a
TEST     1       25532  10-31-88   2:54p
TEST     3       25532  10-31-88   2:54p
TEST     4         481  11-15-89  11:44a
        5 File(s)    227328 bytes free

A>TYPE B:TEST.4
When entering DOS commands, the commands and parameters
must be separated by delimiters.  Delimiters are normally
either a space, a comma, or a semicolon.  They can be used
interchangeable within any command (i.e., Copy A:oldfile,b:).
When entering DOS commands, the commands and parameters
must be separated by delimiters.  Delimiters are normally
either a space, a comma, or a semicolon.  They can be used
interchangeable within any command (i.e., Copy A:oldfile,b:).

A>
```

8. Bonus exercise:
   - Use the **BREAK** command to check the status of the Ctrl-Break mode. Set Break off if it is already on. After it displays the status, clear the screen to blank with the **CLS** command.

- Use the **COPY** command to copy B:TEST.1 to B:TEST.BRK. While it is executing, try to abort the execution by entering a Ctrl-Break (or Ctrl-C) command. When Break is off the entire file is displayed before the Ctrl-Break is recognized by DOS.
- Set Break on and delete B:TEST.BRK. Rerun the **COPY** command and attempt to abort the execution as before. You must act quickly, but it can be done! If the Ctrl-Break was successful, get a directory listing of Drive B to verify that the file was not copied.
- Using the **RENAME** command, change all files on Drive B with a filename of TEST (any extension) to NEWNAME. Use wildcard characters whenever possible. Use the **DIR** command to confirm the results.
- Now, copy all files on Drive B with a filename of NEWNAME to a filename of TEST, without changing the filename extension. Display all the files on Drive B to confirm this operation. Finally, delete all files on Drive B with a filename of NEWNAME.

This concludes Chapter 3 Lab Exercises. When you are finished, remove your disks.

Chapter

4

External Floppy Disk Commands

DOS EXTERNAL COMMANDS

- ☐ ATTRIB (Attribute) Command
- CHKDSK (Check Disk) Command
- COMP (Compare Files) Command
- DISKCOMP Command
- DISKCOPY Command
- FIND Command
- FORMAT Command
- LABEL (Volume Label) Command
- RECOVER Command
- SYS (System) Command

Chapter

4

EXTERNAL FLOPPY DISK COMMANDS

DOS EXTERNAL COMMANDS

All external commands are preceded with an optional disk drive designator, which the computer needs in order to identify the drive containing the external commands. For example, if the default disk is Drive B and DOS is loaded in Drive A, then `A:FORMAT B:` would be required to execute the external command FORMAT and format the disk in Drive B. If the designator is omitted, the system assumes the external command will be found on the default drive. The ten external commands covered in this chapter include:

1. **ATTRIB** — sets the read-only status of a disk file
2. **CHKDSK** — provides a disk status report and fixes disks
3. **COMP** — compares the contents of two disk files
4. **DISKCOMP** — compares the contents of two disks
5. **DISKCOPY** — makes a duplicate copy of a disk
6. **FIND** — locates a file containing a set of characters
7. **FORMAT** — prepares a disk for recording DOS files
8. **LABEL** — creates, changes, or deletes a disk volume label
9. **RECOVER** — recovers a file with defective sectors
10. **SYS** — copies the hidden system files to a formatted disk

The ATTRIB and LABEL commands were added to DOS Version 3.0. They are not available in earlier versions.

ATTRIB (Attribute) Command

Format: `[d:]ATTRIB [±R][±A] [d:]filename[.ext]`

The **ATTRIB** command allows you to set or reset (clear) two file attributes: the read-only attribute and the archive attribute. The read-only attribute of a file can be set to help protect files in a shared or networked environment where you do not want others accidentally destroying your files. If you have important files that you don't want destroyed by overwriting, you can add a degree of safety by marking them as read-only with ATTRIB. Enter +R to set the read-only status and –R to remove it. Files identified as read-only cannot be altered (modified or deleted) without first clearing the read-only attribute with the ATTRIB command.

The archive attribute is automatically set on by DOS whenever a file is modified. The BACKUP, RESTORE, and XCOPY commands (covered later) use the archive attribute to allow for selective backup and restore operations, and for making copies of files that have been modified. The ATTRIB command can be used to manually set or clear the archive attribute of a file.

You may use wildcard characters with this command. If you enter ATTRIB without any parameters, the system displays the current status of the files specified. You must use utility support programs to set and clear the hidden file attribute.

EXAMPLES OF USAGE:

```
A> ATTRIB +r b:filea.txt
```
(sets FILEA.TXT on Drive B to read-only)

```
A> attrib -r B:*.txt
```
(sets all files on Drive B with a .TXT extension so they are not read-only)

```
A> ATTRIB +R A:COMMAND.COM
```
(protects your COMMAND.COM file from being changed accidentally or by a computer virus)

```
A> ATTRIB +A B:*.DOC
```
(sets the archive attribute for all .DOC files on Drive B)

```
A> ATTRIB B:*.*
```
(displays the attribute status of all files on Drive B)

CHKDSK (Check Disk) Command

Format: `[d:]CHKDSK [d:][filename[.ext]] [/F][/V]`

The **CHKDSK** command produces a disk status report for a specified disk and lists the memory status of the system. It can also be used to fix logical errors in the way data was written to disk. After checking the disk, CHKDSK displays any error messages, followed by a status report. Refer to your DOS manual for a description of any error messages that appear. The following is an example of a CHKDSK status report, where the drive designated was a floppy disk and there were no errors detected:

```
Volume DATADISK Created JUL 15, 1990 11:35

  362496 bytes total disk space
   23528 bytes in 3 hidden files
  311296 bytes in 29 user files
   27672 bytes available on disk

  655360 bytes total memory
  568112 bytes free
```

The three hidden files in the status report represent the volume label and the PC-DOS system files (IBMBIO.COM and IBMDOS.COM) that are hidden from normal directory lists. The bottom portion of the report represents the memory status of a computer with 640KB of RAM. RAM is the computer's internal memory used to temporarily store programs and data during processing. The difference between the 655,360 bytes of total memory shown above and the 568,112 bytes free is about 85KB. That is the amount of RAM space allocated to the resident portion of DOS, including the space required to load and execute the CHKDSK command.

As we discussed in Chapter 2, new files are written to contiguous clusters whenever the first unallocated space found is big enough. The recording of an existing file that has been enlarged can cause problems, however. When the original file space has been rewritten, DOS continues writing to the next unallocated cluster on the disk. It writes in available clusters, skipping over those that are already allocated. Consequently, files can easily become fragmented as they expand over time.

If you specify a filename or filenames, CHKDSK displays the number of noncontiguous areas occupied by the file(s). Wildcard characters can be used for the filename. Thus, you can use *.* to determine the extent of file fragmentation on a disk and then use the COPY command to rewrite fragmented files to a newly formatted disk. This process is recommended to improve access speed.

Lost allocation clusters are parts of files that are still recorded in the File Allocation Table (FAT), even though they have been deleted from the directory. This logical discrepancy occurs because of some malfunction, often during the file save process. The usual causes of this damage are loss of power, resetting the computer during a disk write, a program going berserk, or just by hitting Ctrl-Break at the wrong time.

You can use the **/F** option to combine any lost clusters on a disk into a file named FILEnnnn.CHK, where nnnn is a consecutive number beginning with 0001. This is a good command to use periodically, especially if a disk has experienced numerous problems. When the FAT is corrupted, it cannot accurately track files on disk. Therefore, the /F option could result in some loss of data during the fix process. When you use CHKDSK without the /F option and errors are detected, you must run it a second time with the /F option to correct those errors.

The **/V** option allows you to display all files and their complete pathname (paths are covered in Chapter 7) from a specified drive. This option can be very beneficial with hard disk systems.

EXAMPLES OF USAGE:

```
A> CHKDSK
```
(displays a status report for the default drive)

```
A> chkdsk /f
```
(displays a status report for Drive A and fixes any errors found in the FAT)

```
A> b:chkdsk a:*.*
```
(displays a status report for Drive A and lists any fragmented files)

```
A> chkdsk B:read.me
```
(displays a status report for Drive B and displays the number of non-contiguous areas contained in READ.ME)

COMP (Compare Files) Command

Format: `[d:]COMP [d:][filename[.ext]] [d:][filename [.ext]]`

The **COMP** command compares the contents of the first file specified to the contents of the second file. You can use this command to compare two supposedly identical files. When the files are not the same, DOS displays an error message for each location that does not match. After ten mismatches, DOS cancels the operation. Obviously, files must be of the same length to be compared. Wildcard characters can be used, allowing multiple sets of files to be compared. If you do not include any filenames, the system prompts you to enter them.

EXAMPLES OF USAGE:

```
A> B:COMP
```
(executes the COMP command from Drive B, directing the system to prompt you for the files — this method of execution gives you time to replace the system disk with another disk, if required)

```
A> comp filea.doc filea.bak
```
(compares the two files specified to see if they are identical)

```
A> Comp A:*.* b:
```
(compares all identically named files on the two drives specified)

DISKCOMP Command

Format: `[d:]DISKCOMP [d:[d:]]`

The **DISKCOMP** command compares the contents of the two drives specified. It is only used to compare two entire floppy disks of the same type, but not hard disks. It could be a good command to use if you were making numerous copies of a disk and wanted to verify that they were all identical.

EXAMPLES OF USAGE:

```
A> diskcomp a: b:
```
(compares the disk in Drive A with the one in Drive B)

```
A> DISKCOMP A:
```
(compares one disk in Drive A with another to be placed in Drive A later; required in single floppy disk systems)

DISKCOPY Command

Format: `[d:]DISKCOPY [d:[d:]]`

The **DISKCOPY** command copies the entire contents of one floppy disk, track by track, sector by sector, to another disk of the same type. It does not require that the target disk be formatted. For this reason, it is *not recommended* that you use this command. By copying from one disk to another, it is possible that any bad sectors on the target disk (those normally bypassed by using COPY or XCOPY) are overwritten during DISKCOPY. If you execute DISKCOPY, you should also execute DISKCOMP to verify that the disks are identical.

Another potential problem with using DISKCOPY can occur. Any fragmented files on the source disk are copied exactly to the target disk without correcting the fragmentation. You can easily fix file fragmentation and bad sector problems by copying all the files from a fragmented disk to a newly formatted disk.

EXAMPLES OF USAGE:

```
A> DISKCOPY A: B:
```
(makes an exact copy of the disk in Drive A onto a disk in Drive B; the target disk does not need to be previously formatted)

```
A> diskcopy a:
```
(makes an exact copy of the first disk inserted onto a second disk to be inserted into Drive A)

FIND Command

Format: `[d:]FIND "text" [d:]filename[.ext]`

The **FIND** command searches files for a given string of characters. It displays on the monitor all lines from the specified file(s) that contain the desired string of characters. The text entered must be enclosed in a set of double quote marks. Wildcard characters are not allowed in this command, but you may enter as many files as you wish, separating each with a space.

This command has some options, such as counting the number of lines containing the text or displaying the relative line number of each matching line. Consult the DOS manual for more information about these options. The **/I** option can be used to eliminate case sensitivity.

EXAMPLES OF USAGE:

```
A> FIND "Harvard" a:coll.doc b:filea.txt
```
(displays all lines in the files specified that contain the word Harvard, but not HARVARD or harvard — *case sensitivity is critical)*

```
A> FIND "*.*" B:DOS.DOC
```
(displays all lines in B:DOS.DOC that contain the string of characters *.*)

```
A> find "IBM" b:memo.txt
```
(displays all the lines in B:MEMO.TXT that contain IBM)

FORMAT Command

Format: `[d:]FORMAT [d:] [/S][/V]`

The **FORMAT** command prepares a disk in the designated drive to record data acceptable to DOS. It also examines the disk for defective sectors, making a note on the FAT of the sectors that are bad. If you wish to make the designated disk a bootable disk, use the **/S** option. It formats the disk and copies the three system files, IBMBIO.COM, IBMDOS.COM, and COMMAND.COM, from a PC-DOS disk. In MS-DOS, the three system files are named IO.SYS, MSDOS.SYS, and COMMAND.COM.

To uniquely identify each disk, use the **/V** option. This option gives the formatted disk an internal volume label, consisting of a maximum of 11 characters. You can later change a volume label with the LABEL command.

All disks must be formatted before they can be used by DOS. Whenever you format a disk, all previously recorded data is destroyed. For this reason, many organizations hide their format command so it can't be used accidentally. Later in this text you will see how this is accomplished.

FORMAT produces a status report indicating the following statistics for the formatted disk: total disk space, space marked as defective, space allocated to the system files (when /S is used), and amount of space left for your files.

EXAMPLES OF USAGE:

```
A> FORMAT b:/s
```
(formats the disk in Drive B so that it contains the system files, making it bootable)

```
A> format /s/v
```
(formats the disk in the default drive to include the system files and a volume label)

```
B> a:Format
```
(formats the disk in Drive B, the default disk)

LABEL (Volume Label) Command

Format: `[d:]LABEL [d:] [volume label]`

The **LABEL** command allows you to create, change, or delete a volume label on a disk. It is a good idea to label your disks internally so you can identify them using the VOL command without having to physically remove the disk from a drive. If you do not specify a label, the system displays the current label, if any. It then prompts you to enter a label or to press the Enter key to delete an existing label.

EXAMPLES OF USAGE:

```
A> Label b: Fred
```
(creates a volume label of Fred on Drive B)

```
A> LABEL
```
(displays the current volume label on the default drive and prompts you to modify it)

RECOVER Command

Format: `[d:]RECOVER [d:]filename[.ext]`
–or–
`[d:]RECOVER d:`

The **RECOVER** command lets you reconstruct files from a disk that has defective sectors. You can recover a file containing a bad sector, minus the data in the bad sector, by using the first format of RECOVER. In addition, you can recover all files on a disk if the directory has been damaged by using the second format of RECOVER. The second format of RECOVER *should only be used* if the directory of the disk has become unusable. Recovered files may need editing to return them to their original, corrected state. RECOVER cannot be used to recover accidentally erased files, however.

Use of the RECOVER command with no filename causes DOS to read the disk and recreate the FAT and the directory. It generates unique filenames in the form of FILEnnnn.REC, where nnnn is a consecutive number beginning with 0001. It is up to you to figure out the original filenames, however. You might try comparing file sizes from previous directory listings and using the TYPE command to display the contents of listable files.

EXAMPLES OF USAGE:

```
A> RECOVER b:pgm.bas
   (recovers a file called PGM.BAS on Drive B)
A> recover b:
   (recovers directory for disk on Drive B)
```

SYS (System) Command

Format: `[d:]SYS [d:]`

The **SYS** command transfers the hidden system files from the first drive designated to the second drive designated. If you do not specify the first drive, DOS assumes the default drive. It does not transfer the COMMAND.COM file, however. You must transfer COMMAND.COM onto the disk with the COPY command.

The SYS command can be used to transfer a copy of your operating system files to a program disk designed to use DOS, but sold without it. In this case, space at the beginning of the disk already would have been allocated by the manufacturer via formatting with the /B option. The SYS command transfers your hidden system files to the allocated space. It will be a bootable disk after you copy COMMAND.COM to it.

With SYS, you do not have to reformat your hard disk when you upgrade from one version of DOS to a higher one. Just boot the new version from a floppy disk and then use SYS to transfer the new version's hidden files to the hard disk. Then copy the COMMAND.COM file and the rest of the new version commands to the hard disk. Make sure that no older versions of DOS remain on the hard disk, or you will get an "Incorrect Version" error message.

EXAMPLES OF USAGE:

```
A> SYS b:
```
(transfers your system files to the disk in Drive B)
```
A> sys
```
(transfers your system files to the disk in Drive A)

Review Questions

4

1. What distinguishes external commands from internal commands?
2. What command is used to set a file to read-only status?
3. How can you view the read-only status of files on a disk?
4. What are the typical hidden files on a system disk?
5. How can you determine the amount of file fragmentation on a disk?
6. What command can be used to fix a corrupted FAT?
7. What command can be used to list just the filenames on a disk?
8. What is the purpose of the COMP command?
9. What command is used to see if two disks are identical?
10. What is the primary difference between using the COPY command as opposed to the DISKCOPY command?
11. What command is used to display the lines of a file called B:WORK.TXT containing the string of characters Rbt. Smith?
12. How do you make a disk bootable?
13. What is the purpose of a status report when formatting a disk?
14. Why would you want to use an internal volume label?
15. How can the internal volume label be changed?
16. When you reconstruct files from a disk with defective sectors, what must you typically do before you attempt to use those files?
17. What happens when you use the RECOVER command without any parameters?
18. What files are transferred to an existing disk with the SYS command?
19. Why would you likely use the SYS command?
20. What would be contained in a file called FILE0001.CHK?

Chapter 4 Lab Exercises

1. Boot DOS (Drive A) and insert your data disk into Drive B. Enter `DIR B:` to verify the contents of Drive B. To set a file to a read-only status, enter `ATTRIB +R B:TEST.3`. To verify that it was changed correctly, enter `ATTRIB B:TEST.*`. If done correctly, TEST.1 and TEST.4 will not indicate a read-only status and TEST.3 will be a read-only file. An R is displayed to the left of read-only filenames. Figure 4-1 shows you what the screen should look like at this point in the exercise. The A displayed for each file indicates the Archive attribute is on.

Figure 4-1

Screen Display of DIR and ATTRIB

```
A>DIR B:

 Volume in drive B is YOUR NAME
 Directory of  B:\

COMMAND  COM     25532  10-31-88   2:54p
READ     ME        240  11-15-89  11:24a
TEST     1       25532  10-31-88   2:54p
TEST     3       25532  10-31-88   2:54p
TEST     4         481  11-15-89  11:44a
        5 File(s)    227328 bytes free

A>ATTRIB +R B:TEST.3

A>ATTRIB B:TEST.*
  A        B:\TEST.1
  A    R   B:\TEST.3
  A        B:\TEST.4

A>
```

2. To see if the two files you created in the Chapter 3 Lab Exercises are equal, enter `COMP B:TEST.1 B:TEST.3`. If they indicate not being equal, you may have made a keying error. If you were comparing COM, EXE, or SYS files, you would get the following error message: "EOF mark not found".

3. To give a disk on Drive B a volume label, enter `LABEL B:` and follow the system prompts to enter a label of up to 11 characters (i.e., YOUR NAME). Enter `VOL B:` to verify the new volume label. Does your DOS disk have a volume label? Use the VOL command to check, but *don't try to write one* if the DOS disk you are using is not your own.

4. To display all the lines in a file that contain a given string of characters (i.e., "delimiters"), use the FIND command. Enter the DOS command `FIND "delimiters" B:READ.ME`. If you entered the text correctly in the Chapter 2 Lab Exercises, only one line of text should be displayed containing the string of characters "delimiters." Figure 4-2 shows you the results of executing FIND.

Figure 4-2

Screen Display of FIND

```
A>FIND "delimiters" B:READ.ME

---------- B:READ.ME
must be separated by delimiters.  Delimiters are normally

A>
```

5. Enter `CHKDSK` to get a status report of the default drive (A:). Then enter `CHKDSK B:*.*` to get a status report of your data disk, directing the system to check for any fragmented files. Figure 4-3 on the next page shows you the display screen after running CHKDSK. Your display screen should look similar to this one.

Figure 4-3

Screen Display of CHKDSK

```
A>CHKDSK

   362496 bytes total disk space
    56320 bytes in 2 hidden files
   145408 bytes in 19 user files
   160768 bytes available on disk

   655360 bytes total memory
   596624 bytes free

A>CHKDSK B:*.*
Volume YOUR NAME   created Nov 28, 1989 10:40a

   362496 bytes total disk space
        0 bytes in 1 hidden files
    78848 bytes in 5 user files
   283648 bytes available on disk

   655360 bytes total memory
   596624 bytes free

All specified file(s) are contiguous.

A>
```

6. Change the current DOS prompt to display an added message and the system date. Enter `PROMPT It is $d$g What is your command?` Execute a few commands (such as DIR, VOL, and VER) to see the effect of the new system prompt. Then return the system prompt back to normal by entering `PROMPT $n$g`. Figure 4-4 shows you the effect of experimenting with the PROMPT command.

Figure 4-4

Screen Display of PROMPT

```
A>PROMPT It is $d$g What is your command?

It is Tue 11-28-1989> What is your command?dir b:

 Volume in drive B is YOUR NAME
 Directory of  B:\

COMMAND  COM     25532  10-31-88   2:54p
READ     ME        239  11-28-89   9:58a
TEST     1       25532  10-31-88   2:54p
TEST     3       25532  10-31-88   2:54p
TEST     4         479  11-28-89  10:34a
        5 File(s)    283648 bytes free

It is Tue 11-28-1989> What is your command?vol

 Volume in drive A has no label

It is Tue 11-28-1989> What is your command?ver

MS-DOS Version 3.30

It is Tue 11-28-1989> What is your command?PROMPT  $n$g

A>
```

7. Use CHKDSK with the /V option to get a listing of all the files on your DOS disk. Then use the same command to list the files on your data disk.

8. Bonus exercise:
   - Copy B:TEST.1 to B:TEST.ROF. Set the new file to read-only status and try to delete it. Reset the read-only attribute (to allow writing to it) and then delete it.
   - Use the LABEL command to eliminate the volume label on your data disk. Use the VOL command to confirm that it is no longer there. Then change it back to its previous value. Use the DIR command to confirm the change. This exercise demonstrates that there are at least three commands that display a disk's volume label. Can you think of another DOS command that displays the label?

- Use the FIND command to locate all the lines in B:READ.ME that contain the string "comma". There should be three lines. Then use FIND to locate all the lines that contain the string "a comma". Did you get the same number of lines? Why or why not?
- Change the system prompt to "What's next?". Use the CHKDSK command to get the status of Drive B, the status of RAM, and the amount of disk fragmentation on B:READ.ME. Notice the new system prompt? Finally, set the system prompt back to the default.

This concludes Chapter 4 Lab Exercises. When you are finished, remove your disks.

Chapter

5

THREE IMPORTANT CONCEPTS

BATCH FILES

☐

REDIRECTION

☐

PIPING (FILTERS)

☐

Chapter

5

Three Important Concepts

This chapter introduces you to three different DOS concepts that can be very helpful. The first concept is the use of batch files, which allow you to automatically execute one or more DOS commands sequentially. Redirection is another important concept covered in this chapter. It is the technique used for changing the standard input or output device of a DOS command, thereby adding more flexibility to your commands. Closely related to redirection is piping (or the use of filters), a method of transferring the output of one command to the input of another, giving you a way to combine DOS commands.

BATCH FILES

Batch files are one of the most useful and powerful features of DOS. If you want the computer to perform a repeated, standard task or a set of DOS commands, you do not have to enter the commands every time you want to perform that task. With batch files, you don't have to duplicate your effort. When you create a batch file consisting of DOS commands, those commands can be sequentially executed by simply typing the batch file name. This concept works

the same way as an external command. Whenever you enter the name of an external command, such as FORMAT or CHKDSK, DOS goes looking for it on the appropriate disk and executes it for you.

To give you a good idea of how batch file processing works, let's suppose you were tired of having to enter CHKDSK every time you wanted DOS to display a status report. You could create a simple batch file called C.BAT that would enable you to execute the CHKDSK command by entering C, your batch file name. You could use the COPY command to create a one-line batch file from the keyboard (CONsole) as shown below. It is very important to remember to press the Enter key after each line of instruction keyed. Pressing the F6 function key as the only entry of a line of text terminates the process of entering lines of text from the keyboard to a named disk file (in this case, C.BAT).

```
COPY   CON   C.BAT
CHKDSK
<F6>
```

Now, whenever you want to execute the CHKDSK command (for the default drive), you only need to enter the batch file command C to execute the batch file. This may not seem like a big deal, yet it reduces the number of keystrokes for the CHKDSK command considerably.

The most common batch file is **AUTOEXEC.BAT**, appropriately named for the term AUTOmatic EXECution. Immediately after DOS is booted, a ROM chip searches for an AUTOEXEC.BAT file on the system disk. If found, it will execute the predefined set of DOS commands contained in that file. Additionally, it will bypass the automatic prompting for DATE and TIME. You can add these commands to your AUTOEXEC.BAT file, or you can set the correct date and time from a battery-powered clock, assuming one is installed on your system.

Suppose you were setting up an office accounting system that required a specific set of tasks to be done whenever the system was booted. You could create an AUTOEXEC.BAT file containing the specific DOS commands you require. This batch file might look like this:

```
SETCLOCK
COPY   B:BUDGET.WKS   B:BUDGET.BAK
LOTUS
```

The computer would then execute these commands each time the system was turned on. This approach would greatly simplify and standardize the start-up procedures for your accounting system.

The first example of a batch file discussed was a simple one used to reduce the CHKDSK command to the letter C. But it was only able to execute CHKDSK for Drive A. What if you wanted to get a CHKDSK status report for Drive B? Since many DOS commands must work with variable parameters, a nifty batch file feature lets you substitute variable data into batch files. Batch files utilize a special symbol, %n, that allows variable data to be substituted in its place. You may often need more than one variable parameter in batch files, so a number (ranging from 1 to 9) follows the percent sign (%) to indicate which parameter is used. This gives you nine different values that can be substituted into your batch file.

The C.BAT batch file could have been just as easily created with a single **replaceable parameter** (%1) that would allow you to designate a particular disk drive when you executed the batch file. You would need to change the command CHKDSK (in the batch file) to CHKDSK %1. Then, whenever the batch file is executed, the disk drive must be included as a parameter to be substituted for the %1 entry in the batch file. For example, you could enter `C B:` to display a CHKDSK status report for Drive B. The value B: would be substituted for %1 in the batch file during executing of the CHKDSK command. Likewise, `C A:` would cause the batch file to execute the CHKDSK command for Drive A.

The following sample batch file (called COPYTXT.BAT) allows you to make a backup copy of any .TXT file you chose and then obtain a directory listing to visually verify that the copy was completed. It contains two statements and three replaceable parameters as follows:

```
COPY   %2%1.TXT   %3%1.BAK
DIR   %3   /P
```

This batch file looks a lot more complicated than the previous example because it contains three replaceable parameters. Remember that the %n goes in the batch file when it is created and is replaced by the appropriate value when it is executed. The parameters to be replaced in this batch file are as follows:

%1 represents the filename to be copied.
%2 represents the disk drive of the original (source) file.
%3 represents the disk drive of the new (target) file.

A batch file is executed by entering the batch file name followed by the required parameters. Each parameter must be entered in the correct order. For example, if you wanted to use this batch file to make a backup copy of BUDGET.TXT on Drive A by copying it to Drive B with a new extension of .BAK, you could enter:

```
COPYTXT BUDGET A: B:
```

The three parameters supplied with the batch file name during execution of the batch file (BUDGET, A:, and B:) are substituted for the replaceable parameters (%1, %2, and %3) in the batch file. In effect, the execution of this batch file with these three parameters would be identical to entering the commands:

```
COPY   A:BUDGET.TXT   B:BUDGET.BAK
DIR   B:/P
```

Besides the normal DOS commands available for use in batch files, several additional commands improve the power and usefulness of batch file processing. In this chapter we introduce you to three useful batch file commands:

REM — provides for remarks in batch files
PAUSE — pauses to allow operator input into a batch file
ECHO — sets the batch file echo feature on or off

The **REM** (or **remark**) command allows you to document your batch files, making them more readable. These remarks can be displayed on the screen during execution, or shown only when the batch file is displayed with the TYPE command. Normally, we do not want to confuse operators with unnecessary messages on the screen. However, if you wanted to direct an inexperienced operator to insert a specific disk during the processing of a batch file, you could include the following command in the batch file:

```
REM  INSERT THE DATA DISK INTO DRIVE B AND PRESS ENTER
```

However, the operator would not have time to read the message, much less take the appropriate action, before the next batch file command was executed. Fortunately, DOS provides a method for pausing execution long enough to allow you to do something before continuing. The command used is named PAUSE. Whenever the **PAUSE** command executes, it halts execution until you signal it to continue. It automatically displays the message: "Strike a key when ready. ." on the screen.

Another batch file command used to improve communication during execution is the **ECHO** command. It has several variations. If you put the command ECHO OFF in your batch file, DOS will not display (or echo) any of the batch file commands that follow it during execution. This situation can be reversed at any time. If you put ECHO ON in your batch file, all subsequent commands (including REMs) will be displayed on the screen during execution. If you use the ECHO command with a message after it, that message will be displayed, even if you had previously indicated ECHO OFF. An example of a batch file that uses several of these commands is provided in Figure 5-1.

Figure 5-1

Sample Batch File (L.BAT)

```
ECHO OFF
REM  BATCH FILE NAMED L.BAT
ECHO PROCEDURE TO EXECUTE LOTUS 1-2-3
ECHO PLACE THE LOTUS DATA DISK IN DRIVE B
PAUSE
LOTUS
```

To execute the sample batch file in Figure 5-1, you would enter the batch file name (L) without the extension. When DOS executes the first command in the batch file, it sets the echo mode off, so the following REM command would not be displayed. The REM is included in the batch file for documentation purposes only and could be displayed whenever the TYPE command is used to view the batch file contents. In this example, the two ECHO messages would be displayed and the system would pause, waiting for any key to be pressed before continuing. It would then execute the last batch file command, the one that runs Lotus 1-2-3. Figure 5-2 on the next page shows you what the screen displays would look like after executing the first five lines of the batch file.

Figure 5-2

Screen Messages from the Batch File (L.BAT)

```
A>L

A>ECHO OFF
PROCEDURE TO EXECUTE LOTUS 1-2-3
PLACE THE LOTUS DISK IN DRIVE B
Strike a key when ready . .
```

Several fine points relating to batch files are worth mentioning:

1. If you begin a batch file statement with a *colon* (:), it will act as a remark that prints *only* when the file is displayed (with TYPE or EDLIN).
2. To issue a "beep" in a batch file (for alerting the operator to some action or problem), use `ECHO <Ctrl-G>`.
3. If you enter `ECHO OFF` at the system prompt, the system prompt will disappear until you enter `ECHO ON` again or reboot the system.
4. If you press `Ctrl-C (Ctrl-Break)` when a batch file is executing, you will be prompted with a message that asks you if you wish to terminate the batch job.
5. Other batch files can be executed from within a batch file. The last command of a batch file can even be another batch file.
6. When you first execute ECHO OFF in a batch file, the message "ECHO OFF" displays on the screen. DOS 3.3 enhanced this command slightly. You can now use @ECHO OFF to keep "ECHO OFF" from displaying on the screeen.
7. Although the COPY command (with CON) can be used to create relatively simple batch files, it cannot be used for modifying existing batch files. For this you need a text editor like the DOS EDLIN command (covered in Chapter 6) or a word processor.

For a more detailed explanation of batch files and associated batch file commands, refer to the DOS manual.

REDIRECTION

Redirection is the concept of reorienting the standard input or output device used by a command to another input or output device. The standard input device is the keyboard, while the standard output device is the display screen. The "less than" sign (<) establishes a new source of input and the "greater than" sign (>) establishes a new target of output. In the examples that follow, the first > symbol is part of the system prompt.

EXAMPLES OF USAGE:

```
A> DIR > B:DIR.LST
```
(redirects the output display of the DIR command from the screen to a file on Drive B named DIR.LST)

```
A> dir > prn
```
(directs the DIR display to go to the printer)

```
A> Sort < B:filea.txt > B:fileb.txt
```
(uses FILEA.TXT as the input for a SORT command that outputs the sorted results to FILEB.TXT — SORT is one of the filters, covered next)

PIPING (FILTERS)

Piping is a way of telling DOS to transfer the output from one command to become the input for another command. It accomplishes this transferal by creating a temporary file on the default disk for each "piped" set of data. The first command outputs to a temporary pipe file that is read by the second command. If piping requires many operations, multiple pipes are created. When the operation is completed, DOS deletes the temporary files. You can think of piping as a form of redirection. The major difference is that with piping, the temporary file is created by DOS and not by the user.

Piping usually involves the use of special commands, termed filters, that accept data, do something with it, and then pass it to the next step. DOS uses three standard filters in piping:

FIND — searches a file for a specified string of text
MORE — displays only one screen of output at a time, waiting for the user to press any key to continue
SORT — sorts disk records into a desired sequence

The symbol used by DOS to indicate a piping operation is the broken vertical bar (¦). The standard output device for both the DIR and SORT command is the monitor. If you pipe the output from a DIR command into the SORT command, then the output displayed on the monitor will be the sorted listing of the directory in ascending sequence. For example:

```
DIR ¦ SORT
```

The above example is equivalent to the following set of commands:

```
DIR > A:TEMPFILE.$$$
SORT < A:TEMPFILE.$$$
DEL A:TEMPFILE.$$$
```

To reverse the order of a sort, just tack on /R. The following command will produce a directory in descending order:

```
DIR ¦ SORT /R
```

If you find that the sorted directory is too large to fit on a single screen, you can pipe it to an additional command as follows:

```
DIR ¦ SORT ¦ MORE
```

Figure 5-3 shows you what the first page might look like when the above command is executed. The two strangely named files at the beginning of the directory are temporary files used by the piping process. DOS will automatically delete them at the end of the piping process.

Figure 5-3

Screen Display of Piping

```
      21 File(s)    159744 bytes free
 Directory of  A:\
 Volume in drive A has no label
0A300C1E            0  11-28-89  10:48a
0A300C3F            0  11-28-89  10:48a
ANSI     SYS     1647   4-07-88   9:58a
APPEND   EXE     5794   6-08-88   9:52a
ASSIGN   COM     1530   5-11-88   2:10p
ATTRIB   EXE    10656   4-07-88   7:42a
CHKDSK   COM    11003   6-09-88   9:34a
COMMAND  COM    25532  10-31-88   2:54p
COMP     COM     3759   9-10-87   8:40p
DEBUG    COM    15928   4-26-88  10:15a
DISKCOMP COM     6236   6-23-88  10:02a
EDLIN    COM     7495   6-02-88   2:06p
FIND     EXE     6483   6-07-88   3:15p
FORMAT   COM    12206   8-19-88   4:02p
LABEL    COM     2346   4-07-88   7:43a
MORE     COM      282   4-07-88   7:43a
RECOVER  COM     5262   5-06-88   4:01p
--More--
```

If you want to display a directory of files that were created on a given date, you can use the piping concept as follows:

```
DIR ¦ FIND "11-16-90"
```

Likewise, if you wanted a directory of files with an extension of .DOC, you could enter:

```
DIR ¦ FIND "DOC"
```

In this example, FIND would not be able to locate any files if the search string was .DOC, because directory listings do not contain the period before the extension. In addition, it would not find any extensions of doc, because directory listings only contain uppercase filenames.

Piping and redirection can be combined in a single operation. If you wanted a sorted directory listing to be saved on a file called B:SORTED.DIR, you could enter the following:

```
DIR ¦ SORT > B:SORTED.DIR
```

Sometimes it is useful to display data from a listable file on the screen, but only one screen at a time. If ACCOUNT.TXT was a large file on Drive B that you wanted to display on the screen, you could enter:

```
MORE < B:ACCOUNT.TXT
```

This command directs MORE to get its input from B:ACCOUNT.TXT. Unless otherwise directed, MORE sends its output to the monitor one screen at a time.

The efficient use of batch files, redirection, and piping can be quite helpful. It provides you with a tremendous amount of flexibility to create your own customized commands.

Review Questions

5

1. What is the purpose of batch files?
2. How are batch files identified by DOS?
3. What are two typical methods for creating batch files?
4. How are variable data (parameters) included in batch files?
5. When is an AUTOEXEC.BAT file executed?
6. Why are REM statements used in batch files?
7. When are REM statements in batch files displayed?
8. What is the function of a PAUSE statement in a batch file?
9. What happens when you enter ECHO OFF at the system prompt?
10. What is the standard input device in DOS?
11. What is the standard output device in DOS?
12. How would you use redirection to print a directory listing?
13. What kind of file is created with piping?
14. What is the purpose of the MORE filter?
15. What would be displayed by the command DIR ¦ FIND ".COM"?

Chapter 5 Lab Exercises

1. For this exercise, you will be creating some temporary files on the DOS disk, so first you should remove any write-protect tab that is on the disk. Boot DOS (Drive A) and insert your data disk in Drive B. Then enter `DIR B:`. To experiment with redirection and piping, enter the following:

   ```
   DIR > B:TEST1.DIR
   ```
   (to redirect the directory from the monitor to a file named TEST1.DIR)

   ```
   SORT < B:TEST1.DIR > B:TEST2.DIR
   ```
   (to sort the previously created file and redirect it to another filename)

   ```
   DIR | SORT
   ```
   (to list the sorted directory of the default disk)

   ```
   DIR B: | SORT
   ```
   (to list a sorted directory of Drive B—the screen should look like Figure 5-4)

Figure 5-4

Screen Display of SORT

```
A>DIR B:|SORT


       7 File(s)    281600 bytes free
 Directory of  B:\
 Volume in drive B is YOUR NAME
COMMAND  COM     25532  10-31-88   2:54p
READ     ME        239  11-28-89   9:58a
TEST     1       25532  10-31-88   2:54p
TEST     3       25532  10-31-88   2:54p
TEST     4         479  11-28-89  10:34a
TEST1    DIR       876  11-28-89  10:50a
TEST2    DIR       876  11-28-89  10:50a

A>
```

```
DIR ¦ SORT > B:TEST3.DIR
```
(to output sorted directory to a file)

```
TYPE B:TEST3.DIR ¦ MORE
```
(to display a file, one screen at a time)

Figure 5-5 shows you what the screen might look like after executing the last TYPE command and displaying the last screen.

Figure 5-5

Screen Display of MORE

```
      21 File(s)    159744 bytes free
 Directory of  A:\
 Volume in drive A has no label
0A33350A           0  11-28-89  10:51a
0A333525           0  11-28-89  10:51a
ANSI     SYS    1647   4-07-88   9:58a
APPEND   EXE    5794   6-08-88   9:52a
ASSIGN   COM    1530   5-11-88   2:10p
ATTRIB   EXE   10656   4-07-88   7:42a
CHKDSK   COM   11003   6-09-88   9:34a
COMMAND  COM   25532  10-31-88   2:54p
COMP     COM    3759   9-10-87   8:40p
DEBUG    COM   15928   4-26-88  10:15a
DISKCOMP COM    6236   6-23-88  10:02a
EDLIN    COM    7495   6-02-88   2:06p
FIND     EXE    6483   6-07-88   3:15p
FORMAT   COM   12206   8-19-88   4:02p
LABEL    COM    2346   4-07-88   7:43a
MORE     COM     282   4-07-88   7:43a
RECOVER  COM    5262   5-06-88   4:01p
--More--
```

2. If you are currently connected to a printer, you can complete this portion of the lab exercise. Otherwise, just read through it and use your imagination. To print a sorted directory listing on the printer rather than on the monitor, enter:

```
DIR ¦ SORT > PRN
```

To print a sorted directory of just those filenames on Drive B with a filename containing "test" on the printer, enter:

```
DIR B: ¦ FIND "TEST" ¦ SORT > PRN
```

3. Create a single command batch file that will automatically sort the directory of files on the default drive into alphabetical sequence before displaying it one screen at a time:

```
COPY  CON ASORT.BAT
DIR ¦ SORT ¦ MORE
<F6>
```

Enter the batch file name of ASORT to get a sorted directory of the default drive, one screen at a time. The output should look like what you displayed from the last command in exercise 1 (Figure 5-5), except that this directory includes ASORT.BAT.

Before you continue to the next exercise, please remember to delete ASORT.BAT from the default drive (`DEL ASORT.BAT`).

4. Set up a batch file to prevent anyone from formatting a disk on Drive A by mistake. This batch file will only permit a FORMAT command to format a disk on Drive B. Enter:

```
RENAME  FORMAT.COM  FORMATB.COM

COPY  CON  FORMAT.BAT
FORMATB  B:
<F6>
```

Notice that the *original* FORMAT command had to be renamed so you could use FORMAT as the new batch file name. Then the batch file is used to execute the renamed FORMAT command (FORMATB) with the desired option, the formatting of Drive B only.

Remove the disk in Drive B and enter `FORMAT` to execute the batch file just created. When you get the message to insert a blank disk in Drive B, you can press Ctrl-Break to escape from completing the execution of this command. Figure 5-6 on the next page shows what the screen should look like after you terminate the batch file named FORMAT.BAT.

Figure 5-6

Screen Display after Executing FORMAT.BAT

```
A>RENAME FORMAT.COM FORMATB.COM

A>COPY CON FORMAT.BAT
FORMATB B:
^Z
        1 File(s) copied

A>FORMAT

A>FORMATB B:

                FORMAT version 3.30.05
Copyright(C) 1988, Zenith Data Systems Corporation

Insert new disk in drive B:
and press RETURN when ready^C


A>
```

Before you continue to the next step, delete FORMAT.BAT and then rename FORMATB.COM back to FORMAT.COM. Can you think of any other commands you would like to create?

Before you begin exercise 5, check your DOS disk to see if there is an AUTOEXEC.BAT file present. This can easily be accomplished with the command DIR AUTO*.*. If one exists, rename it AUTOEXEC.BAK. Then when you are finished with exercise 5 rename it back to AUTOEXEC.BAT.

5. Use `COPY CON AUTOEXEC.BAT` to create a batch file on your DOS disk containing the following commands:

```
REM SAMPLE INITIALIZATION PROCEDURE
ECHO  OFF
DATE
TIME
ECHO  ON
PAUSE  PLACE DATA DISK IN DRIVE B
DIR  A: > B:DISKA.DIR
DIR  B:
REM  END OF INITIALIZATION
ECHO  OFF
ECHO  HAVE A NICE DAY
PROMPT  DATE IS $D TIME IS $T $_WHAT NEXT?
```

Don't forget to press F6 and the Enter key to cause these commands to be written to the AUTOEXEC.BAT file. If you make an error after you pressed the Enter key on any line, you will have to cancel the process with Ctrl-Break and redo COPY CON from the beginning. When you are done, do a warm boot (`Ctrl-Alt-Del`) to boot DOS and have this new batch file automatically executed.

Experiment entering a few commands such as TIME, VER, and VOL with the new prompt. To change back to the original prompt, enter `PROMPT`.

When you are finished, copy AUTOEXEC.BAT to Drive B and then delete it from the DOS disk.

6. Bonus exercise:
   - Enter `SORT <B:READ.ME >B:READ.SRT` to create a new file, the sorted equivalent of the READ.ME file on Drive B. Use the TYPE command to display it on the screen. Notice that only the complete lines of text were sorted, not each word.

- Enter `TYPE B:READ.ME | FIND "comma"` to display only those lines in the file containing the string "comma". If you are currently connected to a printer, you could redirect the output of this command to the printer by adding >PRN to the end of this command.
- Enter `DIR | MORE`. This causes one or two temporary files to be created by the MORE command, depending on the length of the directory listing. These filenames are created using randomly generated characters to avoid the probability of replacing any existing files. Use the `DIR` command a second time (without MORE) to confirm that DOS automatically deleted the temporary files after it was done using them.

This concludes Chapter 5 Lab Exercises. When you are finished, remove your disks.

Chapter

6

DOS Editing Capabilities

EDLIN

DOS EDITING KEYS

Chapter

6

DOS Editing Capabilities

An effective operating system must provide the user with a relatively easy way to create and modify text files. In addition, it should have some facility for recalling previously entered keystrokes, thus saving the user from having to enter them again.

This chapter covers EDLIN, the DOS line editor. EDLIN is a primitive word processor that lets you create and modify text files. This chapter also covers the use of the DOS editing keys. DOS editing keys let you quickly make changes to the last command entered, or to an EDLIN line of text, without having to reenter keystrokes.

EDLIN

□

EDLIN is the DOS line editor that facilitates the creation and modification of batch files. EDLIN contains its own set of one-letter commands, which enable you to create and save files; update and edit existing files; and search, delete,

or replace multiple lines or characters of text. You can use EDLIN as a rudimentary word processor. However, because EDLIN is line oriented, it is not recommended for the typical memo or document.

Line numbers are included for reference and are only displayed by EDLIN. Line numbers are never saved with the file. When you add or delete lines of text, the line numbers are *automatically updated* by EDLIN. If you refer to a line number greater than the highest line number in the text, EDLIN substitutes the highest number for the one referenced. The asterisk denotes the **current line**, which helps you keep track of where you are in an EDLIN file. The format of the DOS command to execute EDLIN is as follows:

```
[d:]EDLIN [d:]filename[.ext]
```

If the file specified does not exist, EDLIN assumes you want to create a new file and displays "New file." At this point, you can enter **I** (for **Insert**) to begin entering text a line at a time. If the file specified is found, EDLIN displays "End of input file" and awaits your first command.

Although there are several other EDLIN commands, Figure 6-1 contains a summary of some of the most often used EDLIN commands with an accompanying explanation of each. These commands are entered at the EDLIN prompt, which is also an asterisk. This asterisk is displayed on the left side of the screen at position 1. The other asterisk, denoting the current line, is displayed indented at position 11.

The general form of an EDLIN command is a line number (or range of numbers), followed immediately by an EDLIN command. In some cases, these single-letter commands include optional parameters that must be separated with either a period or a comma. Commands may be entered using both uppercase and lowercase characters.

In the event that you are editing a very large file, refer to the DOS manual for a description of the two applicable commands, Append Lines and Write Lines.

Figure 6-1
Summary of EDLIN Commands

C (Copy Lines) — Copies a range of lines to a specified line number and places them just prior to the specified line. For example, to copy lines 2 through 7 and place them before line 17, enter `2,7,17C`.

D (Delete Lines) — Deletes a specified range of lines. For example, to delete lines 4 through 16, enter `4,16D`. To delete line 4 only, enter `4D`. Enter `D` to delete just the current line (the one denoted with the asterisk).

E (End Edlin) — Ends an edit session and saves the edited file. When you exit EDLIN, the original file (if any) is saved with a .BAK extension. The new modified file is saved with the name of the original file.

I (Insert Lines) — Inserts lines of text immediately before the specified line. When you create a new file, you must enter `I` before you begin entering text. To exit from the insert mode, enter `<F6>` or `Ctrl-Break`. For example, to begin inserting lines of text before line 15, enter `15I`.

L (List Lines)— Displays lines of text from the file based on a specified range of lines. For example, to display lines 16–19, enter `16,19L`. To display the 23 lines that surround the current line, enter `L`. To display a total of 23 lines starting with line 34, enter `34L`.

M (Move Lines) — Moves a specified range of lines immediately before a specified line. For example, to move lines 14–27 immediately before line 4, enter `14,27,4M`.

Q (Quit Edlin) — Quits the editing session *without* saving changes.

R (Replace Text) — Replaces all occurrences of a text string with another text string. For example, to change each occurrence of the word MS-DOS to PC-DOS in the first 25 lines of text, enter `1,25RMS-DOS<F6>PC-DOS`, where `<F6>` is the F6 function key used to separate the two strings of text.

S (Search Text) — Searches the file beginning with a specified line to locate a specified text string. For example, to search the entire file for all occurrences of PC-DOS, enter `1SPC-DOS`. The search command allows you to continue locating all occurrences of the specified string until the "Not found" message is displayed.

 # (line number) — Directs EDLIN to display a given line number and lets you change it on the line below where it is displayed. This line number becomes the current line number. If you enter a new line of text, it will replace the old line as soon as you press the Enter key. You may also use the DOS editing keys to quickly make changes within the existing line of text. Characters to the right of the cursor when the Enter key is pressed are erased. Prior to pressing the Enter key, however, you can press Esc to cancel any changes to that line.

DOS EDITING KEYS

Frequently you will need to make minor modifications either to a line of text in a batch file (with EDLIN) or to the last DOS command entered. Rather than having to rekey the entire line of text or command, you can use the **DOS editing keys** to change only the characters that require modification.

Whenever you enter a line, DOS puts a copy of that line in a temporary storage location called an **input buffer**, so you can recall it and make modifications without duplicating keystrokes. The DOS editing keys are used to display the *last* line currently stored in the input buffer and to edit that line. The process of editing involves inserting and/or deleting characters in the line. You cannot just type over existing characters using the DOS editing keys. Figure 6-2 is a summary of the DOS editing keys that you should find most helpful.

Figure 6-2
DOS Editing Keys

F1 — Display one character at a time from the buffer. (You can use the Right Arrow key to do the same thing.)
F2 *x* — Display all characters in the buffer up to the given character (*x*).
F3 — Display all characters in the buffer.
Ins — Insert one or more characters in the buffer at the cursor location.
Del — Delete one character at a time from the buffer.
Esc — Escape (cancel) the operation, leaving the input buffer unchanged.

Perhaps the best way to understand how to use the DOS editing keys is by example. The situations that follow demonstrate what you can do using the DOS editing keys. Practice these examples to reinforce what you have learned thus far.

1. Suppose you entered `COPY A:TEST1.XT B:TEST1.BAK` and got an error message because TEST1.XT was not found on Drive A (it should have been TEST1.TXT). To correct with the editing keys:
   - Press **F2 X** (the F2 function key and the letter X) to move the cursor to the period just before the X, displaying COPY A:TEST1.
   - Press **INS** and the letter **T**.

- Press **F3** to display the rest of the buffer. Now it should show `COPY A:TEST1.TXT B:TEST1.BAK`.
- Press **Enter** to execute the command and get back to the system prompt. It is okay to get an error message at this point since the file TEST1.TXT does not exist on Drive A.

2. Suppose you entered `COPY A:TEST3.DOC B:TEST.BAK` and wanted to redo it because you meant to copy to B:TEST3.BAK. To correct:
   - Press **F3** to display all of the input buffer.
   - Use the **Back Arrow** key to delete the last four characters (.BAK).
   - Reenter `3.BAK` and press the **Enter** key.

3. Suppose you entered `TYPE B:TEST3.DOC` and got what you wanted, but now you also want to display B:TEST2.DOC.
   - Press **F1** until the 3 in the filename is the next to be displayed.
   - Press **DEL** to delete the 3.
   - Press **INS** and **2** to replace the 3 that was deleted.
   - Press **F3** to finish displaying the buffer and **Enter**.

4. Now suppose you want to change the last command entered above from a TYPE command to a DEL command, to delete B:TEST2.DOC.
   - Enter the letters **DEL** to replace the first 3 characters in the buffer.
   - Press **DEL** one time to delete the fourth character.
   - Now press **F3** to finish displaying the buffer and **Enter**.

Review Questions

6

1. What is the dual purpose of the asterisk (*) in EDLIN?
2. What does the EDLIN command 6,12D accomplish?
3. What happens when you exit EDLIN with the command E?
4. What does the command 1,1,8M do?
5. Why is EDLIN considered a very limited word processor?

6. What happens to EDLIN line numbers when lines are deleted?
7. What happens when you enter just a line number as an EDLIN command?
8. When can DOS editing keys be used?
9. What DOS editing key displays all the characters in the buffer?
10. How are characters deleted from the input buffer?

Chapter 6 Lab Exercises

1. This exercise should be done in class as a group. First, experiment with "beginning" EDLIN commands such as Insert, Delete, List, Quit, and End. For this part of the exercise, boot DOS, place your data disk in Drive B, and enter:

```
EDLIN B:LINEDIT.DOC
```

You are now ready to create a new text file. Enter the following commands (clarification comments are noted in parentheses):

I (begin inserting lines of text)
ONE (this is the first line of text)
TWO (continue with THREE through NINE)
.
.
.
TEN
<F6> (F6 function key and Enter to end inserting lines of text)
L (list the lines of text entered)
4D (delete line 4 of the text)
L (list text again to verify deletion)
99I (insert lines of text at the end)
ELEVEN (this is the first line of text to be added)
TWELVE (this is the last line of text)
<F6> (stop inserting text)
L (list text as it now stands)
4 (display line 4 and allow it to be changed)
FOUR (change previous text in line 4 to FOUR)

L (verify change)
5I (insert text before line 5)
FIVE (text to be inserted)
<F6> (stop inserting text)
4,8L (list only lines 4 through 8)
10,12D (delete lines 10 through 12)
Q (instruct EDLIN to quit without saving changes)
N (abort the quit, useful if you entered Q by mistake)
E (instruct EDLIN to exit and save the file)

2. Experiment with DOS editing keys by entering the following:

```
DIR A:   (pressing Enter to execute this command)
DIR A:/P  (using F3 to allow you to easily add /P)

COPY B:TEST3.DR B:TEST3.BAK  (using no editing keys)
COPY B:TEST3.DIR B:TEST3.BAK  (using the DOS editing
   keys to make corrections to previous entry)
```

EDLIN B:LINEDIT.DOC (recall previous text file)
6I (enter the insert mode)
NOW IS THE TIME FOR EVERYONE TO LEARN TO USE COMPUTERS.
<F6> (end inserting text)
6 (list line 6 and allow for change)
<F1> . . . <F1> (press F1 20 times until just before the E in EVERYONE is displayed)
<DEL> . . . <DEL> (press DEL five times to delete EVERY in buffer)
<F3> (display the remainder of the buffer)
L (list the file to verify the change)
Q (quit EDLIN)
Y (respond Y to verify the abort)

3. Experiment with "more advanced" EDLIN commands such as Copy, Move, Replace, Search, and Transfer. Enter the following:

```
EDLIN B:LINEDIT.DOC  (to retrieve the file previously created)
```

L (list the file to refresh your memory)
1,5,10C (copy lines 1–5 before line 10)
7,11,14C (copy lines 7–11 before line 14)
1L (list lines of text beginning with line 1)
1,19RTWO<F6>TO (in lines 1–19, replace every occurrence of TWO with TO)
1L (list lines of text to verify change)
9,19SFIVE (in lines 9–19, search for and locate FIVE)
18TB:READ.ME (at line 18, transfer a file called READ.ME)
1L (list text)
15I (begin inserting before line 15)
SIX (line of text to be inserted)
<F6> (stop inserting)
1L (list text)
16L (list text beginning with line 16)
18 (list line 18 and allow for changes to that line)
<F1><F1><F1> (press F1 three times to display the first three characters of line 18)
TH (add two more characters to that line)
15L (list lines of text beginning with line 15)
E (exit EDLIN and save changes to data disk)

4. Delete the AUTOEXEC.BAT file on Drive B that you created with COPY CON in the Chapter 5 Lab Exercises. Use EDLIN to recreate it.

5. For additional work, build a batch file on your data disk with EDLIN to create a new command for you. This batch file, called MOVE.BAT, will allow you to copy a file, give it another name, and then delete the original name. In other words, you will be creating a substitute for the RENAME command that allows it to rename a file and have the original file copied to another disk. The new batch file on Drive B should contain the following commands:

```
COPY %1 %2
DEL %1
```

This new batch file will be executed by entering the batch file name (B:MOVE) followed by two parameters (for %1 and %2). The parameter %1 represents the original filename and %2 represents the new filename. For example:

```
B:MOVE B:TEST3.DIR B:TEST5.DIR
```

Once executed, B:TEST3.DIR would be renamed B:TEST5.DIR. If you had desired, you could have easily specified that the new file was to be created on Drive A instead of Drive B.

6. Bonus exercise:

   Use EDLIN to create an AUTOEXEC.BAT file on Drive B to display a welcome message, request the system date and time, set the system prompt to display the current date followed by the > character, give a status report on available memory, pause until you press a key to continue, and clear the screen prior to displaying a directory listing of Drive B. This AUTOEXEC.BAT file should look something like this:

```
ECHO OFF
ECHO WELCOME TO DOS
DATE
TIME
PROMPT $D$G
CHKDSK
PAUSE
CLS
DIR B:/P
```

This concludes Chapter 6 Lab Exercises. When you are finished, remove your disks.

Chapter

7

Hard Disk Concepts and Commands

DIRECTORIES AND SUBDIRECTORIES

☐

HARD DISK COMMANDS

☐ FDISK Command
CD (Change Directory) Command
MD (Make Directory) Command
RD (Remove Directory) Command
PATH (Set Search Directory) Command
TREE Command
BACKUP Command
RESTORE Command

Chapter

7

Hard Disk Concepts and Commands

Once you begin using a hard disk system, you will notice two significant improvements over floppy disk processing. First, the speed at which data can be transferred with hard disks is approximately 20 times that of the 5 1/4 inch floppy disks. The second improvement is the amount of data that can be stored on hard disks. A 20MB hard disk can hold the equivalent of about 55 floppy disks. These advantages can significantly change the way you use your computer.

The use of hard disks creates some considerations that are not necessarily relevant when using floppy disks. The first consideration is the need for a good power supply that does not permit loss of electrical power. A temporary power loss can cause the disk's read/write heads to "crash" on the surface of the disk, causing permanent damage. To ensure continuous power, a standby power supply (SPS) is often recommended for hard disk systems. The SPS, usually costing around $400, will protect both the data stored on the disk as well as the hard disk itself.

Another consideration is the need for periodic backup of the data stored on your hard disk to floppy disks. Although this consideration is equally valid for data stored on floppy disks, history shows that hard disk users tend too overlook this process. Get into the habit of backing up your hard disks regularly. You will be glad you did the day you turn on your computer and hear a noise like a spoon in a blender.

One other consideration is the need to move the read/write heads to an unused area of the hard disk before you "power down" the system. When the power is turned off, the read/write heads on some hard disk drives do not retract automatically. Instead, they settle down on the surface of the hard disk. Over time, this process can damage data stored on the disk. If you have a hard disk that does not automatically retract the heads, you should execute a program like PARK or SHIPDISK that moves the heads to a vacant cylinder just prior to turning off the power.

DIRECTORIES AND SUBDIRECTORIES

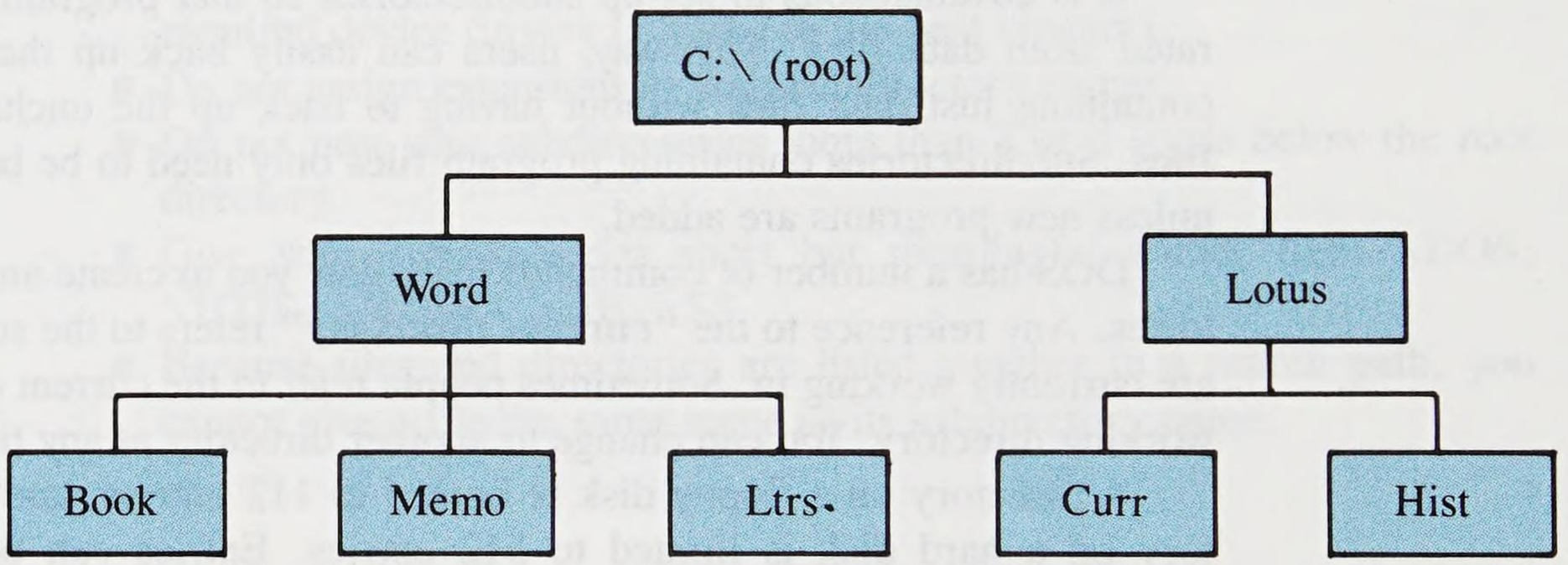
Because large amounts of data can be stored on a hard disk, it is extremely helpful to divide the total space into uniquely named areas. Each area can be reserved to store certain groups of files, thereby allowing you to organize and classify files by area. DOS uses a **root directory** and optional **subdirectories** (directories within a directory) to keep track of the name and location of all files on disk. You can establish subdirectories for floppy disks, but the use of subdirectories is generally more practical for hard disks.

One way to visualize the concept of subdirectories is to compare a single 20MB hard disk to a set of 55 floppy disks. Conceptually, each subdirectory could represent a single floppy disk, without the physical limitations of 360KB, however. And like floppy disks, each subdirectory could be devoted to a given application, like word processing, spreadsheets, accounting, and so on. Just as you change floppy disks, so you can change to another subdirectory. Additionally, just as commands like DEL, DIR, and COPY all relate to a given floppy disk, they apply to a given subdirectory.

With the release of DOS 2.x, you can organize and control hundreds of files on hard disk by adopting a tree-structured file directory system. The root directory branches into subdirectories. The subdirectories, in turn, can branch into further subdirectories in a hierarchy much like that of a family tree. Each subdirectory is assigned a unique name using the same rules we use with filenames. With floppy disks, files are often organized manually by recording selected groups of files on a disk and identifying each disk with a label. Subdirectories provide a big advantage in that they allow you to organize all the files on a hard disk "electronically." Within each directory, files can be added and new subdirectories can be created. Figure 7-1 shows a graphic example of a tree-structured directory.

Figure 7-1

Sample Hierarchy of Subdirectories



In the sample tree structure in Figure 7-1, the root directory is divided into two subdirectories: one for word processing and one for spreadsheets. The word processing subdirectory, WORD, is further subdivided into three subdirectories: one for manuscripts, one for memos, and one for letters. It is possible for you to navigate through this structure by starting at the root and traveling down any of the desired branches to get to the desired subdirectory.

DOS uses the backslash (\) to identify a subdirectory name. Thus, the file BOSS.DOC stored on the MEMO subdirectory is identified with the full name C:\WORD\MEMO\BOSS.DOC. The full filename identifies a **path** of subdirectories that must be taken by DOS to find the file in the hierarchy. If the path is not included in the filename, the system will look for the file on the current directory. The backslash at the beginning of the path directs DOS to begin the path with the root directory. This is considered an explicit path. If the backslash is not included, the path will begin with the current directory. This is known as a relative path. If the file you need is not on the current directory, you must provide DOS the path to find it. This path must be part of the full filename, just as the disk designator is supplied when a file is not on the default drive. Later in this chapter we discuss the PATH command. The PATH command is used as a default path for program files.

It is advantageous to set up subdirectories so that program files are separated from data files. This way, users can easily back up the subdirectories containing just data files without having to back up the unchanged program files. Subdirectories containing program files only need to be backed up once, unless new programs are added.

DOS has a number of commands that allow you to create and use subdirectories. Any reference to the **"current directory"** refers to the subdirectory you are currently working in. Sometimes people refer to the current directory as the working directory. You can change to another directory at any time.

A directory on a floppy disk is limited to 112 entries, and the root directory on a hard disk is limited to 512 entries. Entries can be filenames or subdirectory names. Subdirectories are identified on a directory listing (via the DIR command) with the symbol <DIR>. The command DIR *. can be used to display all the subdirectories contained in the current directory. In subdirectory listings you see "dot" and "double-dot" directory entries, like this:

```
.       <DIR>     7-15-90     9:45a
..      <DIR>     7-15-90     9:45a
```

The single dot represents the current directory and the double-dot entry represents the "parent" directory, one level up from the current directory. These two entries can be used as shorthand notation when referencing a directory. For example, the command DEL. is the same as DEL *.*. Now to change to the LOTUS\HIST directory from the LOTUS\CURR directory, enter `CD ..\HIST` (because both subdirectories have the same parent).

DOS uses a search path to locate a file if it is not on the current directory. In DOS 3.3, if you specify just a subdirectory in a command without specifying any files, DOS will apply that command to all files in that subdirectory. Thus, the following two commands are treated the same by DOS 3.3:

```
DEL \LOTUS\HIST\*.*
DEL \LOTUS\HIST
```

The following hints will assist you in correctly setting up the directory structure on a hard disk:

- Do not clutter up your root directory with lots of programs or data files. Generally, just a few files need to be in the root directory: hidden files, COMMAND.COM, AUTOEXEC.BAT, and CONFIG.SYS with its required device drivers (covered in the next chapter).
- Do not assign extensions to your subdirectory names.
- Do not nest your subdirectories more than 2 or 3 levels below the root directory.
- Give your subdirectories short but meaningful names like: \DOS, \UTIL, \WORD, \DB, \SS.
- Because files and directories are listed together in a search path, you cannot give a file the same name as its subdirectory name.

HARD DISK COMMANDS

Figure 7-2 lists and briefly defines eight of the commands you will be most likely to use with hard disk systems. Each command is explained in detail in this chapter, along with examples of usage. The commands are listed and discussed in a logical sequence rather than in alphabetical sequence to facilitate understanding.

Figure 7-2
Hard Disk Command Summary

FDISK – Fixed Disk Setup, used to divide a fixed disk into one or more separate areas (partitions).
CD – Change Directory, used to change to another subdirectory or the root directory.
MD – Make Directory, used to create a subdirectory.
RD – Remove Directory, used to eliminate a subdirectory.
PATH – Instructs DOS where to look for command files.
TREE – Displays the complete directory tree structure.
BACKUP – Copies hard disk files to floppy disks.
RESTORE – Restores hard disk files from floppy disks.

The format for all commands designated as external includes the optional parameter [d:][path] just before the command. These parameters may be used to designate the disk drive and/or the subdirectory path that contains the external command.

FDISK Command (external) — PART Command (MS-DOS)

Format: `FDISK` (requires the DOS disk in Drive A)

The **FDISK** command creates and manages partitions on a hard disk. Normally, you will only have a single hard disk partition, devoted exclusively to DOS. However, the capability exists to have multiple operating systems

installed on a hard disk, in which only one can be active at any time. The FDISK command allows you to create partitions, change from one partition to another, or delete partitions. Even if you want just one partition, you must still run FDISK to allocate all of your hard disk to DOS.

When you first execute FDISK, choose the option that "Creates DOS partition." The system will then ask you if you want to use the entire hard disk system for DOS. If you wish to have DOS as the only operating system, respond with a Y. Otherwise, respond with an N and consult the DOS manual for further instructions. After the hard disk partition is established, you must run FORMAT to prepare it for recording data. Run FORMAT with the /S option to place the two DOS hidden system files on the disk. You should copy COMMAND.COM and all the DOS external commands to your hard disk and then put away your DOS floppy disk for safekeeping. Once the system files are transferred to the hard disk, you can boot the system from the hard disk. When you turn on a computer with a bootable hard disk, do not have any disk in Drive A or the system will try to boot using the floppy disk.

CD (Change Directory) Command (internal) — also CHDIR

Format: `CD [d:][path]`

The **CD** command lets you change from the current subdirectory to another one. The path is used to identify the directory you want to change to. For example, if you want to change to the root directory, you enter CD\. A leading backslash (\) in the path directs DOS to start the path at the root directory. Normally, you want to start at the root directory to make sure DOS will be able to locate the appropriate subdirectory. You can enter the CD command with no parameters to display the current directory. The optional disk drive designator is used if the subdirectory you wish to change to is not on the default disk.

You can save keystrokes by recognizing that the symbols "\" and ".."are considered delimiters in DOS commands, just like a space and the slash (/). Thus, the command CD\ is interpreted the same as CD \. Likewise, the command CD.. is the same as CD ...

EXAMPLES OF USAGE:

```
C> CD \
```
(changes to the root directory)

```
C> Cd
```
(displays the current directory)

```
C> cd\word\memo
```
(changes to the subdirectory named MEMO on the WORD subdirectory, starting from the root directory)

```
C> CD WORD\MEMO
```
(changes to the subdirectory identified as MEMO on the WORD subdirectory, starting from the current directory)

```
C> cd..
```
(changes to the parent subdirectory)

MD (Make Directory) Command (internal) — also MKDIR

Format: `MD [d:]path`

The **MD** command creates a subdirectory in a given location. You may create as many subdirectories as you want, but keep in mind that too many could cause confusion. Each subdirectory can contain both filenames and subdirectory names that can also occur on other subdirectories, but all names must be unique within a subdirectory.

EXAMPLES OF USAGE:

```
C> MD\word
```
(creates a subdirectory named WORD one level down from the root directory)

```
C> md \word\MEMO
```
(creates the subdirectory MEMO one level down from the subdirectory named WORD)

```
C> Md games
```
(creates a subdirectory called GAMES one level down from the current directory)

RD (Remove Directory) Command (internal) — also RMDIR

Format: `RD [d:]path`

The **RD** command removes a subdirectory from disk. However, before you can remove a subdirectory, all files within that subdirectory must be deleted. In addition, all of its subdirectories must be removed. You can only remove a subdirectory from another subdirectory. You cannot remove the root directory.

EXAMPLES OF USAGE:

```
C> rd \word\memo
```
(removes the subdirectory named MEMO from the WORD subdirectory)

```
C> RD\WORD
```
(removes the subdirectory WORD from the root directory)

```
C> rd memo
```
(removes subdirectory MEMO from the current directory)

PATH (Set Search Directory) Command (internal)

Format: `PATH [d:][path][;path][;path]`

The **PATH** command allows you to direct the system to search one or more subdirectories for commands (or batch files) not found in the current subdirectory. With a tree-structured directory, you cannot always access any command just by entering the command name. You must specify the path by which DOS can locate the command if it resides on a subdirectory other than the current directory. The PATH command instructs DOS which subdirectories to search and in what order to find a command that is not on the current directory.

PATH only locates files that can be executed. Within each directory, DOS always looks for a matching command with a .COM extension first. Then it tries the .EXE extension, and finally a .BAT extension. Beginning with DOS 3.2, the APPEND command was added to allow users to set up a search path for data files. Consult Chapter 8 or your DOS manual for additional details.

A few of other points concerning the PATH command should be mentioned. You are allowed to have only one search path active at a time. Typing PATH with no parameters will display the current path. Entering PATH with just a semicolon tells the system you do not want any search path. And finally, issuing a PATH command does not change the current directory.

EXAMPLES OF USAGE:

```
C> Path \DOS3
```
(directs the system to look on the subdirectory named DOS3 if it cannot find the desired command or batch file name on the current subdirectory)

```
C> PATH\word\ltrs
```
(directs the system to look on LTRS within WORD to locate a command, if it is not on the current subdirectory)

```
C> path
```
(displays the current search path setting)

```
C> Path ;
```
(deletes any previous search path setting)

```
C> PATH \;\UTIL;\DOS
```
(directs DOS to search 3 directories in the order given: root, \UTIL, and \DOS)

TREE Command (external)

Format: `[d:][path]TREE [d:][/F]`

The **TREE** command displays all the subdirectory paths on the specified drive. When the **/F** option is used, TREE will optionally list all the files in the root directory and each subdirectory. However, the TREE command does not create a very readable listing with the /F option. If you don't like TREE /F, try using CHKDSK /V. It lists all files, showing the complete pathname.

EXAMPLES OF USAGE:

```
C> TREE /F
```
(displays all the subdirectories and files in the root directory and each subdirectory)

```
C> tree >prn
```
(prints a listing of all the subdirectories)

```
C> CHKDSK /V ¦ FIND ".BAK" ¦ MORE
```
(locates every .BAK file in every subdirectory on Drive C, displaying them a screen at a time)

BACKUP Command (external)

Format: `[d:][path]BACKUP d:[path][filename[.ext]] d: [/S][/M]`, where the first disk drive designator after the command is the source drive and the second designator is the target.

The **BACKUP** command copies your hard disk files to as many formatted floppy disks as are needed to hold it in a compressed form. Beginning with DOS 3.3, the disks no longer have to be formatted before you execute the BACKUP command; it automatically formats new disks. You may include a filename to restrict the files you want to back up. Wildcard characters can be used in the filename and extension.

The BACKUP command provides you with several options that make it easier to control how you copy your files. You can use the **/S** option to copy subdirectory files in addition to the files in the specified directory. The **/M** option can be used to back up only those files that have been modified since the last backup.

Whenever BACKUP fills a floppy disk, you will be prompted to insert a new diskette. Label each backup disk in consecutive order so you can restore them in the same order. It is important to remember that *BACKUP is not the same as COPY*. When files are recorded on the backup disks with BACKUP, they cannot be used until they are restored to the hard disk with the RESTORE command.

EXAMPLES OF USAGE:

```
C> BACKUP C:\*.* A:/S
```
(backs up all files in all subdirectories on Drive C to Drive A)

```
C> backup c:\word\book\*.doc a:/m
```
(backs up only files in the BOOK subdirectory within WORD that have an extension of .DOC and have been modified since the last backup on that subdirectory)

```
C> Backup C:\*.txt a:/s
```
(backs up all files on the hard disk with an extension of .TXT)

RESTORE Command (external)

Format: `[d:][path]RESTORE d: d:[path]filename[.ext] [/S]`, where the first disk drive designator after the command is the source drive and the second is the target drive.

The **RESTORE** command is used to restore one or more files from backup disks created with the BACKUP command. When multiple backup disks are involved, the will system prompt you to insert the next diskette. You may use the wildcard characters for filenames and extensions with the RESTORE command.

With the **/S** option, RESTORE will automatically recreate deleted subdirectories on a hard disk, if needed. RESTORE can be used along with BACKUP to transfer hard disk files to another hard disk. This feature could come in handy if you wanted to store your files on a "loaner disk" while your disk is being repaired.

EXAMPLES OF USAGE:

```
C> RESTORE A: C:\LOTUS\*.*
```
(restores all files on the LOTUS subdirectory from the backup disks on Drive A)

```
C> restore a: c:\*.doc/s
```
(restores all files from Drive A with an extension of .DOC for all subdirectories on Drive C)

Review Questions

7

1. How are files arranged or organized on hard disks?
2. Why is a constant supply of good electrical power especially important when using hard disks?
3. What is the purpose of the PARK or SHIPDISK commands?
4. How are subdirectories designated in DOS?
5. What is a DOS path?
6. Why might you want to have relatively short subdirectory names?
7. What is the primary function of the FDISK command?
8. What command allows you to switch from the current subdirectory to another subdirectory?
9. How do you switch to the root directory?
10. How can identical filenames (i.e., FORMAT.COM) exist multiple times on the same disk?
11. What command creates a subdirectory named "DOS" from the root directory?
12. What command creates a subdirectory named "DOS" from the current directory?
13. What is necessary to be able to remove a subdirectory from a disk?
14. If multiple subdirectories are included in a search path, which one is searched first?
15. How are multiple subdirectories specified in the PATH command?
16. What command is used to view the current search path?
17. What command displays all subdirectory names on a disk?
18. What command is used to display all subdirectory names and their files on a disk?
19. What kinds of files can be recreated with the RESTORE command?
20. How can subdirectories be restored?

Chapter 7 Lab Exercises

These exercises do not require a hard disk system.

1. Make a hierarchy of subdirectories on Drive B according to Figure 7-1 of this text as follows:

```
A> MD B:\WORD
A> MD B:\WORD\BOOK
A> MD B:\WORD\MEMO
A> MD B:\WORD\LTRS
A> MD B:\LOTUS
A> MD B:\LOTUS\CURR
A> MD B:\LOTUS\HIST
```

   Did you remember to use the DOS editing keys to assist you?

2. Copy LINEDIT.DOC residing on the root directory of Drive B to each of the seven subdirectories created above.

   Example: `COPY B:LINEDIT.DOC B:\WORD`

3. Check out your new directory structure by entering:

```
A> DIR B:
```
   (lists files and directories in root directory)
```
A> DIR B:\word
```
   (lists all files and directories in \WORD — the screen should look similar to Figure 7-3)
```
A> TREE B:/F
```
   (lists all files and directories on Drive B — the last part of the screen should look similar to Figure 7-4)

Figure 7-3

Screen Display of Subdirectories on \WORD Directory

```
A>DIR B:\WORD

 Volume in drive B is YOUR NAME
 Directory of  B:\WORD

.            <DIR>      11-28-89  11:35a
..           <DIR>      11-28-89  11:35a
BOOK         <DIR>      11-28-89  11:35a
MEMO         <DIR>      11-28-89  11:35a
LTRS         <DIR>      11-28-89  11:35a
LINEDIT  DOC       362  11-15-89   3:07p
        6 File(s)    262144 bytes free

A>
```

Figure 7-4

Screen Display of TREE

```
Files:              LINEDIT.DOC


Path: \LOTUS\CURR
Sub-directories:  None

Files:              LINEDIT.DOC


Path: \LOTUS\HIST
Sub-directories:  None

Files:              LINEDIT.DOC


A>
```

4. Make the current directory MEMO:

```
A> B:
     (changes default drive to B)
B> CD\WORD\MEMO
     (makes MEMO the current directory)
B> DIR
     (verifies change to desired directory)
B> CD\
     (changes back to root directory)
B> DIR
     (lists all files and subdirectories on the root directory of Drive B)
```

5. Delete the subdirectory named HIST:

```
B> DEL \LOTUS\HIST\*.*
     (deletes all files from HIST first)
B> RD \LOTUS\HIST
     (removes HIST subdirectory)
B> A:TREE B:
     (tests the removal of HIST by running TREE — an external command on Drive A)
```

6. Set up a path to your DOS external commands:

```
B> PATH A:
     (sets path to include Drive A)
B> TREE B:
     (tests the path, noting that it will find TREE on Drive A after first searching Drive B)
```

7. Change the system prompt to display the current subdirectory. Then change to several subdirectories to see the effect:

```
B> PROMPT $P$G
B> CD\WORD
B> CD\LOTUS\CURR
B> CD\
```

8. Bonus exercise:
   - Make sure the default disk is Drive B. Copy at least three files from the root directory of Drive B to the WORD subdirectory created previously in this lab exercise.
   - From the root directory of Drive B, create subdirectory TEMP.
   - Change to TEMP, copy all files from B:\WORD to TEMP, and use the DIR command to verify the copy process. Use wildcard characters whenever possible to save keystrokes and minimize errors.
   - Finally, remove the TEMP subdirectory and change back to the A> prompt. Did you remember to delete all the files in TEMP first?

This concludes Chapter 7 Lab Exercises. When you are finished, remove your disks, and remember, "practice is the best teacher."

Chapter

8

MANAGING HARD DISK SYSTEMS

HARD DISK MANAGEMENT TECHNIQUES

- ☐

MORE HARD DISK COMMANDS

- ☐ APPEND Commnad
  ASSIGN Command
  XCOPY Command

Chapter 8

MANAGING HARD DISK SYSTEMS

HARD DISK MANAGEMENT TECHNIQUES

One of the most difficult areas of DOS to master deals with the ability to effectively control and manage all the files that accumulate on hard disks. If you are not well organized, you will find that a computer can quickly make you disorganized. Several techniques for managing hard disk systems were mentioned in previous chapters, such as using good file naming conventions and creating appropriate subdirectories. But there are many more techniques that can be used to improve your hard disk system. The overview in Figure 8-1 on the next page provides a good checklist of hard disk management techniques.

Figure 8-1

Hard Disk Management Tips

- Develop good file naming conventions to help you better organize files.
- Use standard filename extensions to aid in classifying file types.
- Make backup copies of all important data files regularly.
- Organize all hard disk files into logical tree subdirectories.
- Place data files on a separate directory from program files. This will shorten the time it takes to back up data files. (Program files do not need to be backed up on a regular basis.)
- Use AUTOEXEC.BAT and CONFIG.SYS files to customize your boot process.
- Limit your root directory files to just those needed for booting, such as the COMMAND.COM, CONFIG.SYS, and AUTOEXEC.BAT files.
- Use `PROMPT $P$G` in an AUTOEXEC.BAT file to change the system prompt to display the current subdirectory location (full path).
- Use the PATH command in an AUTOEXEC.BAT file to tell DOS where to look to find program files that are not on the current (default) subdirectory.
- Regularly issue the DIR command to verify what files are on the current subdirectory.
- Use `DIR *.` to display subdirectory names found in the current subdirectory (subdirectories do not have extensions).
- Use `CD..` to change to the parent subdirectory in the tree, one up from the current subdirectory in the tree structure.
- Periodically use `CHKDSK *.* /F` to identify file fragmentation and combine (fix) any lost allocation clusters on the disk.
- Use a menu system (like Automenu) to assist you in executing tasks.
- If you have only one floppy drive, create a temporary subdirectory on hard disk to serve as a transfer area for copying from one floppy disk to another.
- Always park the read/write heads of your hard disk before you turn off your computer, especially before you attempt to move it.
- Use a standby power supply to ensure a good supply of power to your computer. This will minimize any physical damage to your hard disk.
- Plug your computer, monitor, and printer into a common power switch and use its on-off switch as a master power switch for the system. It is much better to replace a $10 power switch than it is to have to replace a power switch on a computer component.
- Use write-protect tabs on floppy disks that you don't intend to use to record more data. You can always remove them if you need to later.
- Use a file management utility program like Treeview (Chapter 12) to simplify the process of hard disk management.

As suggested in Figure 8-1, a menu program (like Automenu) can be very helpful in managing your hard disk. Automenu is covered in detail in Chapter 13. It makes your system much friendlier and easier to use. You may find it difficult to remember how to execute applications when you normally run many different ones. The menu approach lets you select an application from a menu displayed on the screen and have it automatically executed. All the parameters necessary to run each application are included in separate batch files. The appropriate batch file is then executed based on the application selected from the menu. To help you better understand menus, the bonus exercise at the end of this chapter shows you how to you create a simple menu system.

MORE HARD DISK COMMANDS

In addition to a menu system, three DOS commands can be very helpful in managing a hard disk system. These are the APPEND, ASSIGN, and XCOPY commands.

APPEND Command (external)—DOS Version 3.3

Format: `[d:][path]APPEND [d:][path][;[d:][path]...]`

The **APPEND** command sets a search path for data files. In other words, it is the "PATH" command for data files. Remember, the PATH command only works for executable files. It is very useful when application programs need to find files on other directories. The APPEND command is an excellent command to place in your AUTOEXEC.BAT file, along with PATH and PROMPT.

You can specify more than one route to search by separating each path desired with a semicolon. If you use the APPEND command without any parameters, it will display the current data path. If you use it with just a semicolon, it will clear the data path to its original default value of NUL. With a NUL data path, DOS searches only the current directory for data files.

EXAMPLES OF USAGE:

```
A> APPEND b:\word\ltrs;b:\word\memos
```
(if a data file needed by DOS is not on the current directory, DOS will first search b:\word\ltrs and then b:\word\memos to find the needed data file)

```
A> APPEND
```
(displays the current set of data paths)

```
A> APPEND ;
```
(sets the data path to NUL)

ASSIGN Command (external)

Format: `[d:][path]ASSIGN d=d [d=d]`

The **ASSIGN** command lets the user reroute a disk drive designator. Some older application programs were designed under the assumption that the program disk would always reside in Drive A and the data disk would always be in Drive B. However, with newer hard disk systems, it is likely that you would want both the programs and data to reside on the hard disk (Drive C). To execute those older programs, you could trick DOS by using the ASSIGN command as follows:

```
ASSIGN A=C B=C
```

This command reassigns both Drive A and Drive B to your hard disk, as required by the program. To cancel drive assignments, enter the command ASSIGN without any parameters. A helpful hint: Whenever you use the ASSIGN command, incorporate it into a batch file that executes the program and automatically returns the assignments back to normal. The execution of any subsequent commands would become very confusing if you forgot to reset the disk drives.

With the release of DOS Version 3.1, you can use the SUBST command, covered later in Chapter 10, to accomplish the same type of logical disk reassignment. Thus, the following commands are equivalent:

```
ASSIGN A=C
SUBST A: C:\
```

XCOPY (External Copy) Command (external)—DOS Version 3.2

Format: `[d:][path]XCOPY [d:][path]filename[.ext] [d:][path][filename[.ext]] [/M] [/P] [/S]`, where the first filename is the source file and the second (optional) filename is the target file.

The **XCOPY** command is used to selectively copy files and directories, including any lower level subdirectories and their files. If you do not specify a disk drive or path, DOS uses normal default values. If you do not specify any target file, DOS assumes you wish to copy to the default directory using the same filename. If the specified path does not exist on the target disk, XCOPY creates the directories before it copies files to the disk.

The source can include multiple files with the use of wildcard characters. It will not copy hidden files, however, nor can it copy a single file that is bigger than the capacity of the target disk.

XCOPY uses an effective buffering technique to copy files. Multiple files are read into RAM (as much as the RAM buffers will hold) and then are copied to the target disk as a group. When you have a large number of files to copy, this approach is faster than using the COPY command, which copies only one file at a time.

The **/M** switch copies only source files that have their archive bit set on, and then turns off the archive bit in the source file, just like the BACKUP command. This lets you copy only those files that have not yet been backed up. This option makes it rather easy to copy many files onto multiple backup disks. You can set the archive bit of any file on by using the ATTRIB command with the "+A" parameter. Use XCOPY to copy all files on the disk with the archive bit on. As each disk fills up, the files copied will have their archive bit set off. Repeat the XCOPY command until no more files have their archive bit on.

The **/P** switch prompts you with a message allowing you to selectively confirm whether you want to copy each file. This option allows you to issue a single XCOPY command with wildcard characters and then selectively copy desired files from that set of files.

The **/S** switch copies files from directories and all lower level subdirectories. *If you omit this switch, DOS will only copy files in the single directory specified.* If the source is a directory, then DOS makes the target a directory. If a target directory does not exist, DOS will automatically create one.

The XCOPY command is a powerful command with lots of flexibility and execution options. If you run out of room on the target disk, the unfinished file being copied at that time will be erased from the destination, but the previous files copied will remain intact. The buffering techniques used by XCOPY tend to make it faster than the COPY command for copying large or multiple files. Refer to your DOS manual for additional information.

EXAMPLES OF USAGE:

```
A> XCOPY \word\*.DOC b:\bkup\
```
(copies all files in the WORD directory on the default disk to a directory on Drive B named \BKUP, without changing any filename)

```
C> xcopy \*.* b:/m /s
```
(copies all files on the default disk with the archive bit set on from all directories to Drive B, without changing any filenames)

```
A> XCOPY \*.* B:*.BAK /S/P
```
(copies all files on Drive A from all directories to Drive B, changing the extension to .BAK and pausing to confirm that each file should be copied)

Review Questions

8

1. What advantages are gained by using a menu program?
2. When would you likely need to use the ASSIGN command?
3. In what way is the APPEND command like a DOS PATH command?
4. In what ways are the XCOPY and COPY commands alike?
5. In what ways are the XCOPY and COPY commands different?
6. What happens when the following command is executed?

   ```
   XCOPY A:\LOTUS\*.* B: /P
   ```
7. What happens when you add the /S option to the command in the previous question?
8. What benefits are obtained by placing all of a given application's data files on separate subdirectories?
9. What files should you put on the root directory of a hard disk?
10. What DOS commands would you likely find on the AUTOEXEC.BAT file?

Chapter 8 Lab Exercises

1. The SORT filter may be used to sort data in either a text file or a directory listing. Since it is able to sort beginning with any column you specify, it becomes very useful when you have records to sort. To specify column, use the /+n option, where *n* represents the number of the column that you wish to begin sorting. For example, to sort a directory by filename extension, enter:

```
DIR ¦ SORT /+10
```
(extensions begin in column 10 of a directory listing)

This will yield a rather strange-looking directory listing because all the heading lines are included in the sort. Because filename entries on a directory listing include the date stamp, you can limit the directory lines to be displayed by including the FIND filter as follows:

```
DIR ¦ SORT /+10 ¦ FIND "-"
```

Create a batch file called **SIZEDIR.BAT** that will sort and display a directory in sequence by file size. The size of the file, in bytes, begins in position 14 of the directory display. Use the following commands:

```
ECHO OFF
REM SIZEDIR.BAT TO DISPLAY DIR SORTED BY FILE SIZE
DIR %1 ¦ SORT /+14 ¦ FIND "-" ¦ MORE
REM END OF SIZEDIR.BAT
```

To execute this batch file, remember to include the disk drive designator as a parameter when entering the batch file command.

2. Assume for the moment that you have a system with a hard disk and only one floppy disk. Drive A will serve as a "hard disk" for this exercise. You want to make a backup copy of one of your floppy disks which contains

data that is not on your hard disk. Because the current DOS disk in Drive A has a limited amount of space left, this exercise may work only if you put a formatted disk that contains no files in Drive A.

Create a batch file called **BKUP.BAT** to facilitate this copy process. Echo should not be turned off when this batch file is being executed, so you can see the batch file steps during its execution. The subdirectory name BKUP8765 was chosen because it would be unlikely that it already exists on Drive A. Remember, Drive A is substituting for a hard disk. This batch file should contain the following statements:

```
REM  BACKUP A FLOPPY DISK ONTO ANOTHER FLOPPY USING
REM  A ONE FLOPPY, SINGLE HARD DISK SYSTEM.
REM
MD  A:\BKUP8765
ECHO  INSERT ORIGINAL FLOPPY IN DRIVE B:
PAUSE
COPY  B:*.* A:\BKUP8765
ECHO  INSERT BLANK FLOPPY IN DRIVE B:
PAUSE
FORMAT  B:
COPY  A:\BKUP8765\*.* B:
ECHO Y ¦ DEL A:\BKUP8765\*.*
RD   A:\BKUP8765
ECHO  END OF BACKUP
DIR  B:/P
```

3. Suppose you ran the CHKDSK command and were informed by DOS that you had some bad sectors on your disk. Unfortunately, the CHKDSK command does not identify which files are bad. It is only when you read a bad file that DOS identifies the bad files. Using the NUL device in DOS provides a clever way to overcome this problem: you can copy all the files on a disk to NUL. This command reads each file on the disk, but does not actually copy them anywhere. If a file being copied contains a bad sector, DOS displays the filename along with the "Abort, Ignore, or Retry?" message. When this occurs, simply enter I to ignore the error and continue. Try this command on your data disk by entering `COPY *.* NUL`.

4. No one enjoys backing up a hard disk onto floppies. It is time-consuming and boring. But backing up just the essential files is better than no backup at all. Create the following batch file (**BACK.BAT**) that can be run periodically to back up just your essential files to one or more floppy disks:

```
ECHO  OFF
REM   BACK.BAT
ATTRIB  +A C:\*.*
ECHO  PLACE FIRST BACKUP DISK IN DRIVE A
PAUSE
XCOPY  C:\*.* A: /M
ATTRIB  +A C:\WORD\*.DOC /S
ECHO  PLACE SECOND BACKUP DISK IN DRIVE A
PAUSE
XCOPY  C:\WORD\*.DOC A: /M/S
ECHO  BACKUP COMPLETED IF NO ERRORS OCCURRED
```

Do not execute this batch file, because it may not be applicable to your DOS disk. This is a good batch file to create now and save, as you might wish to modify and use it later. Perhaps you can think of other essential files that should be added to this batch file.

You may have already realized that this batch file has a major limitation, however. If any set of files to be copied exceeded the remaining capacity of either backup disk, some of the files would not be copied. An exercise in Chapter 11 using the FOR IN DO batch file command resolves this limitation.

5. Bonus exercise:

   Earlier we discussed the benefits of using a menu program to automatically run selected applications from a menu. In this exercise you will use a less sophisticated approach to create a batch file that acts as "the world's simplest menu." First you need to create a series of batch files. Each one should represent the commands necessary to run a particular application. Make up several batch files and store them on a subdirectory called \MENU. Then create a batch file (**APPL.BAT**) on \MENU that can be used to display a list of all these batch file names. This batch file should turn ECHO off, clear the screen, and display a sorted directory listing (by filename) of all the batch files on \MENU, pausing at the end of each full screen. After the list of possible batch files to be executed displays on the screen, key in the filename you want to execute to be certain it works.

This concludes Chapter 8 Lab Exercises. When you are finished, remove your disks.

Chapter

9

CUSTOMIZING DOS

CONFIG.SYS (CONFIGURATION COMMAND) FILE

- ☐ ANSI.SYS
  VDISK.SYS (RAMDRIVE.SYS)

MODE COMMAND

- ☐

Chapter

9

CUSTOMIZING DOS

CUSTOMIZING DOS

In this chapter we are going to cover one of the most intriguing aspects of DOS: how to customize, or tailor, DOS to meet your needs. We have already discussed the AUTOEXEC.BAT file, which is one way to customize the system boot process. This chapter discusses two more ways: the use of the CONFIG.SYS file and the MODE command.

CONFIG.SYS (CONFIGURATION COMMAND) FILE

Format: (a list of special commands to configure the system)

CONFIG.SYS is a special file used in the last stages of the boot process just prior to the execution of AUTOEXEC.BAT. CONFIG.SYS allows you to specify how your system should operate and be configured. In its earliest form, DOS was a rigid creation with all of its devices and configuration constraints predefined. As the PC evolved, it was important that DOS be able to reconfigure itself easily.

The CONFIG.SYS file gives DOS the flexibility to adapt. It lets you improve system performance by specifying how memory is used, and lets you install device driver programs for controlling other hardware devices. Like a batch file, it is a text file of commands, usually created with EDLIN. The order of these configuration commands is irrelevant.

Even though the CONFIG.SYS file is optional and may not be required on your system, many DOS users find it very beneficial. Here are the configuration commands that you are most likely to use.

BREAK = ON

Directs DOS to check for a Ctrl-Break (Ctrl-C) from the keyboard during all disk read/write operations. Normally, DOS only checks for a Ctrl-Break during keyboard, printer, screen, or COM port operations. BREAK is also one of the DOS internal commands discussed in Chapter 3.

BUFFERS = nn

Specifies how many disk buffers should be set up by DOS. The parameter nn is the number of buffers desired to improve disk performance. By specifying a relatively high number of buffers, you tell DOS to read a larger chunk of data from your disk at one time. The next time your program needs data, DOS checks to see if it is already in the RAM buffer. If it is, access is almost immediate. Because the first microcomputers had a limited amount of RAM, the DOS default was 2 buffers for the PC-XT and 3 for an AT. Beginning with DOS 3.3, the default number is automatically allocated by DOS (up to a maximum of 15) based on the amount of RAM it finds available. A recommended setting of 20 buffers is sufficient for most circumstances. Each buffer uses 528 bytes (about 1/2 KB) of memory. It usually requires some experimenting to find the most effective buffer size for your system. The maximum number of buffers allowed by DOS is 99.

FILES = nn

Specifies how many files can be open at any given time. DOS defaults to 8 files unless you specify otherwise, with each file requiring 48 bytes of RAM. Because the total number of open files includes the two hidden system files, COMMAND.COM, and RAM-resident programs, you should plan on specifying at least 20 files. Database applications often require more than 20 open files.

DEVICE = x

In this CONFIG.SYS entry, x represents a device driver file, such as the ANSI.SYS, MOUSE.SYS, DRIVER.SYS, or the VDISK.SYS file. A device driver is a special program that tells DOS how to handle input/output from a given peripheral device, such as a keyboard, mouse, or disk. The mouse driver is a classic example of how a device driver gives DOS the ability to work with hardware that was not designed into the original I/O system routines of DOS. A device driver must be supplied for each device that requires it to be installed. The commands shown in Figure 9-1 are typically included in a CONFIG.SYS file.

Figure 9-1

Typical CONFIG.SYS File

```
BREAK = ON
DEVICE = ANSI.SYS
DEVICE = VDISK.SYS
FILES = 20
BUFFERS = 15
```

Two commonly used device drivers are VDISK.SYS and ANSI.SYS. VDISK.SYS creates a RAM disk and ANSI.SYS is an extended screen and keyboard driver. They are both RAM-resident programs.

ANSI.SYS

ANSI.SYS is an interface between the monitor and DOS, as well as between the keyboard and DOS. It can be used to change the colors on the screen or redefine the function keys on the keyboard. The American National Standards Institute (ANSI) has a specified set of codes that can be used to control the keyboard and monitor. These codes are activated by a special "escape sequence" of control codes and are used by ANSI.SYS to modify the keyboard and monitor. Since most personal computers follow the ANSI standards, these escape sequences can be used whenever the ANSI.SYS device driver is installed. This chapter discusses some of the ways you can use ANSI.SYS.

First you must install the device driver by placing DEVICE = ANSI.SYS in the CONFIG.SYS file. After you change the CONFIG.SYS file, you must reboot the system to install the device driver. If your ANSI.SYS file resides on a DOS subdirectory, place the file's complete pathname in the CONFIG.SYS entry. For example:

```
DEVICE = C:\DOS\ANSI.SYS
```

By using the ANSI.SYS file supplied with DOS, you can assign virtually any DOS command, program name, or batch file name to a function key. There are 10 function keys on the keyboard. These can be used in combination with the Shift, Control, or Alternate keys to provide 30 additional functions. DOS uses standard keycodes, shown in Figure 9-2, to represent the combinations of function keys.

Figure 9-2

ANSI.SYS Standard Keycodes

Keyboard Keys	Keycodes
F1 through F10	59 - 68
Shift-F1 through Shift-F10	84 - 93
Ctrl-F1 through Ctrl-F10	94 - 103
Alt-F1 through Alt-F10	104 - 113

There are two sets of ANSI codes. The first set defines the standard ASCII keys like A and a on keyboards that are used by all operating systems. The other set of codes defines the keys that are specific to DOS-compatible computers. These include the function keys, used independently or in combination with either the Shift, Alternate, or Control keys. The ANSI device driver uses the following escape sequence to redefine a key on the keyboard:

```
Esc[0;keycode;"redefined value";13p
```

This sequence requires entering Escape as the first character. Because DOS immediately exits COPY CON or EDLIN when the Esc key is pressed, the PROMPT command is normally used to send an escape sequence. For example, to change Alt-F2 to represent DIR B:, enter the following PROMPT:

```
PROMPT $E[0;105;"DIR B:";13p$P$G
```

The first three characters in the PROMPT parameter ($E[) are used by DOS to identify an escape sequence to ANSI.SYS. The next character, a zero, is used to redefine the extended ASCII character codes (see Appendix D) on the keyboard. The number 105 is the specific keycode for Alt-F2. The set of characters enclosed in quote marks is the redefined value for the keycode. The next three characters in the PROMPT command (13p) are the code for the return key and are needed to complete the command. The character p must be entered in lowercase. The remaining characters are needed to reset the system prompt back to its current status. In this case, it is reset to display the current directory followed by >. To reassign Alt-F2 back to its original keyboard setting, you can either reboot the system or enter:

```
PROMPT $E[0;105;0;105;13p$P$G
```

When entering multiple PROMPT commands to reassign more than one key on the keyboard, you only need to reset the system prompt on the last command. However, because ANSI.SYS is limited to the total number of keystrokes it can remember for key definitions (depending on the DOS version), you may run out of space. If you redefine only a few keys and limit the length of the definition, this may not pose much of a problem. Another limitation of using ANSI.SYS is that these definitions only work at the DOS level. Normally, you cannot use ANSI.SYS to define keys within application programs. You can use a utility support program like Superkey or Prokey for more extensive control over your keyboard.

ANSI.SYS is a special device driver that not only controls the keyboard, but also controls screen cursor positioning and sets display colors. You can change the foreground and background colors of the screen. One set of ANSI codes that is of particular interest to users of color monitors is the list of color codes shown in Figure 9-3.

Figure 9-3

ANSI Color Codes

Color Code	Attribute Setting
0	All attributes off
1	High intensity on
4	Underline on
5	Blinking on
7	Inverse video on
8	Invisible display
30	Foreground black
31	Foreground red
32	Foreground green
33	Foreground yellow
34	Foreground blue
35	Foreground magenta
36	Foreground cyan
37	Foreground white
40	Background black
41	Background red
42	Background green
43	Background yellow
44	Background blue
45	Background magenta
46	Background cyan
47	Background white

The form of the PROMPT command used to change the attributes of the monitor is: `PROMPT $E[a..am`

As in the previous examples, the $E[initiates the escape sequence for ANSI.SYS. The *a* values in this command represent a variable number of screen attributes, each separated with a semicolon (;). The lowercase m is used to terminate the screen attributes. Of course, the color attributes are only applicable when using a color monitor.

EXAMPLES OF USAGE:

```
C> PROMPT $E[7m$P$G
```
(sets the monitor to display dark text on a light (reverse) background)

```
C> PROMPT $E[0m$P$G
```
(resets the screen attributes back to normal)

```
C> PROMPT $E[37;44m$P$G
```
(displays white foreground characters on a blue background)

```
C> PROMPT $E[0;30;46m$P$G
```
(resets all display attributes back to normal before displaying black characters on a cyan background)

```
C> PROMPT $E[7m$P$G$E[m
```
(sets the system prompt in inverse video mode and then returns to the normal mode for other text)

```
C> PROMPT $E[1mWhat Next?$E[0m
```
(turns high intensity on for displaying the system prompt "What Next?" and then turns it off for other characters)

```
C> PROMPT $E[0;68;"Dir B:*.";13p$P$G
```
(reassigns the F10 key to enter DIR B:*. when pressed)

VDISK.SYS (RAMDRIVE.SYS)

In addition to ANSI.SYS, DOS has a device driver that lets you allocate a portion of main memory as an extra disk device. This portion of RAM is often referred to as a RAM disk or an "electronic disk." The device driver is called **VDISK.SYS** (in PC-DOS) or **RAMDRIVE.SYS** (in MS-DOS). Because access speeds in RAM are much faster than those of a hard disk, any files loaded into a RAM disk are accessed immediately. DOS automatically creates the RAM disk drive designator using the next available disk drive letter, such as D or E.

To install the RAM disk capability you must have VDISK.SYS included in your CONFIG.SYS file as a device driver. If you want more than the default of 64KB of RAM allocated to the RAM disk, you can enter DEVICE = VDISK.SYS 360. In this case, 360KB of RAM would be allocated to the RAM disk (about the size of a floppy disk). If you have a hard disk, you should keep all DOS commands on a separate subdirectory. Then you should include the full pathname when entering the device name in a CONFIG.SYS entry. For example: `DEVICE = C:\DOS\VDISK.SYS`.

To begin using the RAM disk, you should copy any DOS commands, batch files, and programs that you expect to execute most often into the RAM disk area. For example: `COPY C:\WP\WORD.COM D:`.

This copy process should be part of your AUTOEXEC.BAT file so that it is done automatically each time you boot the system. In addition, you should make sure your path command includes the new RAM disk drive. It should be included as the first search entry so that files on the RAM disk are accessed first. For example: `PATH = D:\;C:\DOS;C:\UTIL`.

Because valuable data could easily be lost during power failures or surges, it is a dangerous practice to put data files on a RAM disk. Be sure to save any data files you have on a RAM disk to permanent disk storage with the COPY command before turning off the CPU.

MODE COMMAND (external)

□

The **MODE** command sets the operational modes of your computer and lets you reconfigure the way your system devices operate. It lets you set options for printers, monitors, and communications ports. Consequently, there are three general formats of the MODE command.

Format for the Printer: `[d:]MODE LPTx: cpl,lpi`

This format of the MODE command lets you set and control several characteristics for a printer that is IBM- or Epson-compatible. The character x is the printer number (1–3), cpl is the characters per line (80 or 132), and lpi is the number of lines per vertical inch (6 or 8). When executed, the MODE command sends the appropriate control codes to the printer. The default settings are 80 characters per line and 6 lines per inch.

Format for the Monitor: `[d:]MODE display type`

This format of the MODE command lets you set the number of characters to be displayed per line on the screen and either enable or disable color. The display types are shown in Figure 9-4.

Figure 9-4
MODE Display Types

Type	Monitor Setting
40	40 characters per line (cpl) displayed
80	80 cpl displayed
BW40	40 cpl with color disabled
BW80	80 cpl with color disabled
CO40	40 cpl with color applications enabled
CO80	80 cpl with color applications enabled

Format for COM Ports: `[d:]MODE COMx: br [,p,db,sb]`

This format of MODE lets you define the asynchronous communication parameters for your system's communications ports, like COM1 and COM2. The character x = the COM port, br = the transmission rate, p = parity type, db = the number of data bits, and sb = the number of stop bits.

MODE allows a variety of other command options. Refer to the DOS manual for further information on the MODE command.

EXAMPLES OF USAGE:

```
A> Mode LPT1:132,8
```
(sets the printer assigned to LPT1 to print 132 characters per line and 8 lines per inch)

```
A> MODE lpt1:132
```
(sets the LPT1 printer to print at 132 characters per inch and uses the default of 6 lines per inch)

```
A> MODE LPT1:,8
```
(sets the LPT1 printer to print 8 lines per inch—the first parameter is skipped, causing the default of 80 characters per inch to be used)

```
A> mode co80
```
(sets the monitor to display 80 characters per line with color enabled)
```
B> a:mode 40
```
(sets the monitor to display only 40 characters per line)
```
A> mode com1:1200,e,8,1
```
(sets the COM1 port to communicate at 1200 baud, with even parity, 8 data bits, and 1 stop bit)

Review Questions

9

1. What is the purpose of the CONFIG.SYS file?
2. How is the CONFIG.SYS file typically created?
3. What is the importance of increasing the default values for the number of buffers and the number of open files in DOS?
4. What is a "device driver"?
5. What device driver is used to create a RAM disk?
6. Why might it be dangerous to load data files to a RAM disk?
7. What device driver must be loaded to allow you to reassign function keys on the keyboard?
8. What benefit could be gained by reassigning function keys?
9. Why is the PROMPT command normally used to modify the keyboard?
10. Specifically, what PROMPT command would be required to display black text on a yellow background?
11. What color code could be used by DOS to allow a password to be entered without being displayed on the screen?
12. What command is used to load a mouse driver in DOS?
13. How is the CONFIG.SYS file loaded by DOS?
14. What is the purpose of the MODE command?
15. What does the command MODE COM1:110,E,7,2 do when executed?

Chapter 9 Lab Exercises

1. Set up a batch file to reassign two special function keys. Call this batch file **FKEY.BAT** and create it as follows, remembering to enter the p in lowercase:

```
ECHO OFF
REM  BATCH FILE TO REDEFINE FUNCTION KEYS
REM
REM  KEYCODE 99 (CTRL-F6) = "CHKDSK B:"
REM  KEYCODE 100 (CTRL-F7) = "VER"
REM
PROMPT $e[0;99;"chkdsk b:";13p$e[0;100;"ver";13p$n$g
```

Execute the batch file and test it. The ng at the end of the last prompt statement allows the system prompt to be displayed. You can easily modify this batch file with EDLIN to reassign other function keys. Early versions of DOS limited you to reassigning a single key at a time, however.

2. It might be handy to reassign Alt-A to display a directory of Drive A, sorted by file size, with the largest files displayed first. This new command should have the ability to filter out unimportant directory entries and pause when the screen is full. It should also let Alt-B and Alt-C produce the same type of directory listings for Drive B and Drive C, respectively.

   Create a text file (**SDIR.TXT**) that can be used to reassign the Alt keys as described above. To assist you, the text file commands are as follows:

```
ECHO ESC[0;30;"DIR A: | SORT /R/+14 | FIND ";34;"-";34;" | MORE";13P
ECHO ESC[0;48;"DIR B: | SORT /R/+14 | FIND ";34;"-";34;" | MORE";13P
ECHO ESC[0;46;"DIR C: | SORT /R/+14 | FIND ";34;"-";34;" | MORE";13P
```

Note: Key code 30 may not work with all versions of DOS. If this is the case with the version you are using, substitute Alt-A with Shift-F1 (key code 84).

Another way to send an Esc key to DOS (to reassign keys) without causing an escape or exit of an operation is with the ECHO command. The ESC in each ECHO command represents the Escape key. You can enter ESC in EDLIN without causing an exit (escape) by entering a Ctrl-V keystroke followed by a [. Then enter the rest of the escape sequence beginning with the [after the ESC. Each ";34;" in the batch file entries is used to provide the decimal ASCII value of a quotation mark, required to surround the dash (-) in the FIND command. To invoke these keyboard reassignments, enter `TYPE SDIR.TXT`.

3. Create a batch file (**PRT132.BAT**) that lets you print any text file on the printer with compressed print (132 characters/line). This is handy for displaying spreadsheets or wide documents on 8 1/2-by-11-inch paper. If you are attached to a printer, test the following batch file using a text file like FKEY.BAT as the replaceable parameter:

```
ECHO OFF
REM  PRT123.BAT TO PRINT COMPRESSED
REM  PARAMETER USED IS THE FILE TO BE PRINTED
MODE LPT1:132
COPY %1 LPT1
MODE LPT1:80
```

4. Bonus exercise:
   - Assume you are leaving your computer for a short time and want to discourage novice users from using your system. By entering ECHO OFF followed by CLS, your screen becomes blank with just the flashing cursor. Experiment with this by running some commands without any prompt displayed. Then restore the prompt by turning echo back on.
   - Use EDLIN to create a **CONFIG.SYS** file on Drive B that provides for up to 20 files and 15 buffers in DOS, allows the user to redefine the use of the keyboard keys, allocates 256K of RAM for a RAM disk, and sets Break ON. The order of the CONFIG.SYS file entries is irrelevant.

This concludes Chapter 9 Lab Exercises. When you are finished, remove your disks.

Chapter

10

ADVANCED DOS COMMANDS

ADVANCED DOS COMMANDS

- ☐ FASTOPEN Command
  PRINT Command
  REPLACE Command
  SUBST Command

USING DEBUG

- ☐

Chapter

10

ADVANCED DOS COMMANDS

The more you know about the capabilities of DOS, the more power and control you have over it. In this chapter we will acquaint you with four advanced commands that can be very helpful: FASTOPEN, PRINT, REPLACE, and SUBST. They are all external commands and some are also TSR (terminate and stay resident) programs. Once a TSR program is executed, it remains in RAM until you reboot the system. More information on TSR programs is included in Chapter 12. Figure 10-1 on the next page provides a brief overview of the four advanced commands.

Figure 10-1

Summary of the Advanced DOS Commands Covered in Chapter 10

FASTOPEN improves disk performance by loading disk information into RAM for faster access.

PRINT allows files to be printed concurrently with the execution of other commands.

REPLACE selectively copies files from one disk/directory to another.

SUBST substitutes a disk drive designator for a lengthy pathname.

In addition to these commands, Chapter 10 introduces you to a special utility program called DEBUG. DEBUG is a powerful program that operates at the machine language level of your computer. It can be used for a wide variety of rather unusual tasks.

ADVANCED DOS COMMANDS

□

FASTOPEN Command (external, TSR) DOS Version 3.3

Format: `[d:][path]FASTOPEN d:[=size] [d:[=size]]`

When DOS needs information about files and directories on a disk it must spend precious time accessing the disk. The **FASTOPEN** command is used to improve the performance of a hard disk by minimizing the number of disk accesses required. It utilizes RAM to keep track of files and directories on a specified disk. Once FASTOPEN is activated for a specified disk, the disk's file and directory information is loaded into a RAM buffer by DOS, making it immediately available. This is especially helpful if you have a large number of subdirectories that contain many small files.

The FASTOPEN command allows you to specify multiple disk drives. A size parameter may be specified for each disk. The size controls the number of files and directories DOS can monitor in RAM. The default size is 34 entries, but a more practical range is 50–200 per disk. Each entry uses 35 bytes, so a size of 100 would occupy 3500 bytes of RAM. You can increase the size up to a maximum of 999 entries.

There are some limitations to this command, however. The disk drive you specify with FASTOPEN cannot be one that you have used with ASSIGN, SUBST, or JOIN. To avoid conflicts with other batch file commands, it should be included last in your AUTOEXEC.BAT file. FASTOPEN can only be used once per boot session, so the only way to modify a FASTOPEN setting is to reboot the system. And finally, if you issue FASTOPEN with several disk drives, the sum of all the sizes cannot exceed 999 entries.

EXAMPLES OF USAGE:

```
A> FASTOPEN C:
```
(increases performance of Drive C using a default size of 34 entries)
```
A> fastopen d:=100
```
(increases performance of Drive D using a size of 100 entries)
```
A> FASTOPEN C:=100 D:=50
```
(increases performance of both Drive C and Drive D)

PRINT Command (external, TSR)

Format: `[d:][path]PRINT [/B:buffer] [/S:time] [/D:device] [d:]filename[.ext]`

Normally, when you use the TYPE or COPY command to print a disk file, you must wait until all printing is completed before executing another DOS command or application program. You can avoid lengthy delays when you have long documents to be printed by utilizing the **PRINT** command. The printer is extremely slow relative to the internal speed of the computer. Thus, a small portion of computer resources (RAM space and CPU time) can be allocated to the print process, while the majority of resources can be used for other processing at the same time. It is analogous to chewing gum while reading your DOS manual. Your mind spends relatively little time directing your mouth, concentrating most of your thinking power on understanding the DOS manual.

PRINT is a RAM-resident program. It is loaded into RAM the first time it is executed and stays there until you reboot the system. Another program or command (that you want to run at the same time) is also loaded into RAM. DOS executes the programs concurrently, allocating a defined slice of time and a buffer size to the PRINT program.

The maximum buffer size is 32,767 (32KB) with a default of 512 bytes (1/2 KB). The maximum time slice is 255 slices per second with a default of 8 time slices. When using PRINT in a batch file, the **/D** option lets you predefine the printing device (normally PRN). If you do not use the /D option, DOS prompts you to identify the printing device. Of course, you must also specify the file to be printed.

Like many of the advanced DOS commands, the PRINT command has some limitations. It may only be used to print the output of files previously stored on disk in a printable format. Unless your application programs (i.e., word processing, spreadsheet, and database) can output "print files" to a disk, the PRINT command is of limited help. It cannot be used to print the output of a currently executing program. In addition, you cannot change or delete a file while it is being printed with PRINT, nor can you attempt to use the printer for another operation while PRINT is using the printer. Because the description of the PRINT command in this text does not cover all its capabilities, you should refer to your DOS manual for other options.

As an alternative to PRINT, you can purchase a utility support program called a print spooler. Print spooling programs greatly simplify the process of concurrent printing. A print spooler automatically captures the output that DOS sends to the printer and stores it either on a disk or in a RAM buffer. The spooler then sends the captured output to the printer whenever the printer becomes available. A typical print spooler, like Print-Q, costs about $100.

Another alternative for concurrent printing is a print buffer, a hardware device connected between your computer and the printer. A print buffer has its own RAM, usually ranging from 256KB to 1MB. Text sent to the printer from the CPU can fill up the print buffer very quickly. The print buffer then releases the text to the printer at a much slower rate, depending on the speed of the printer. Once a complete document has been sent to the print buffer, the CPU can execute other commands. Those commands could be executed concurrently with printing from the buffer. The cost of a print buffer varies greatly depending on the size of the buffer, but it is generally more expensive to purchase than a print spooler.

EXAMPLES OF USAGE:

```
A> PRINT /b:1024/s:64 B:Fed.txt
```
(places the file FED.TXT from Drive B into a buffer of 1024 bytes, allocates 64 time slices, and prints it concurrently with another command)

```
A> print fileb.doc
```
(prints FILEB.DOC on the default drive at the same time it executes other processing operations with a default of 512 bytes and 8 time slices/second)

```
A> PRINT /D:PRN FILEA.TXT
```
(prints FILEA.TXT concurrently without being prompted for the device, using the default buffer size and number of time slices)

REPLACE Command (external) DOS Version 3.2

Format: `[d:][path]REPLACE [d:][path]filename[.ext] [d:][path] [/A][/P]`

The **REPLACE** command is simply an advanced, selective version of the COPY command. It simplifies the process of file management by allowing for selective updating of files. It is especially helpful for updating your new or modified files to a backup disk at the end of the day.

REPLACE has two distinct forms of operation: (1) It copies (replaces) matching files by filename, and (2) it copies (adds) only nonmatching filenames. It does not copy hidden or DOS system files, however. Wildcard characters can be used to define a set of filenames to be copied.

To illustrate the first form of operation, let's develop a likely scenario. Suppose you wanted to replace all .DOC files on Drive B with only those files from Drive A that have matching filenames. You could use the following REPLACE command:

```
REPLACE A:*.DOC B:
```

The **/A** option is used for the second form of operation. The REPLACE command with /A adds files from the source directory to the target directory (or disk) that do not have matching filenames on the target disk. This option can add newly created files from a working directory to a backup disk without having to update all matching files on the target disk.

The **/P** option directs DOS to pause before copying each file specified and asks you if it is okay to copy that file. The /P option can be used with either form of operation (i.e., with or without the /A option).

EXAMPLES OF USAGE:

```
A> REPLACE *.TXT B:
```
(replaces .TXT files on Drive B with the contents of Drive A files that have the same filename)

```
A> replace *.* C:\word /a
```
(adds all files on Drive A to C:\WORD that are not currently on C:\WORD)

```
A> replace c:\word\*.doc c:\bkup /p
```
(allows selective replacement of any .DOC files on the BKUP directory with matching filenames on the WORD directory, prompting for each replacement)

SUBST (Substitute) Command (external) DOS Version 3.1

Format: `[d:][path]SUBST d: [d:]path`
(the specified disk drive designator is substituted for the path name)

—or—

Format: `[d:][path]SUBST`
(SUBST with no parameters displays a list of current substitutions)

—or—

Format: `[d:][path]SUBST d: /D`
(the **/D** option disconnects the specified substitution)

The **SUBST** command lets you substitute a disk drive designator for a pathname. This can be used to replace a long and complex pathname with a logical disk drive designator that is much shorter and easier to remember. This is especially helpful when application programs have limits on the length of pathnames.

Before you use the SUBST command you should be aware that DOS limits the number of disk drives allowed. DOS supports Drives A through E (by default) unless you reconfigure DOS otherwise. Suppose Drive E was already assigned to some function, such as a RAMdisk, and you wanted to substitute Drive F for a pathname. In this case, you must include LASTDRIVE = F as an entry in your CONFIG.SYS file when DOS is booted.

EXAMPLES OF USAGE:

```
A> SUBST E: C:\LOTUS\HIST\DEPT1
   (substitutes Drive E for the pathname)
A> subst
   (lists all current substitutions)
A> SUBST E:/D
   (disconnects Drive E from its path)
```

USING DEBUG

Format: `[d:][path]DEBUG[d:][path][filename[.ext]]`

DEBUG is a machine-level utility program for snooping around in DOS and making all sorts of subtle modifications. It can display memory values, modify them, and view executable files like those with .EXE or .COM extensions. If you know what you are doing, you can use DEBUG to recover the lost disk files, edit the COMMAND.COM file and direct DOS to replace the AUTOEXEC.BAT file with another file during the boot process. With an understanding of assembly language, you can even use DEBUG to create new executable commands. Several chapters could easily be devoted to this extraordinary utility program. Our intent, however, is to gently introduce you to DEBUG, not to make you a DEBUG expert!

One use of the DEBUG command is to view the contents of an executable file. If you entered `DEBUG A:LABEL.EXE`, you can then enter several DEBUG commands that let you view the contents of the LABEL command. The first thing you would see displayed on the screen is the hyphen (-), which is the DEBUG command prompt. You can use the DEBUG command D to instruct it to display the next 128 bytes from any file. Figure 10-2 shows you what the screen looks like when you use DEBUG to view the first 128 bytes of LABEL.EXE from Version 3.2 of MS-DOS.

Figure 10-2

Screen Display of DEBUG

```
A>debug label.exe
-d
1486:0190                                               EB 2A 4D              .*M
1486:01A0  53 2D 44 4F 53 20 4C 41-42 45 4C 20 50 72 6F 67   S-DOS LABEL Prog
1486:01B0  72 61 6D 20 56 65 72 73-69 6F 6E 20 31 2E 30 32   ram Version 1.02
1486:01C0  20 34 2F 33 30 2F 38 35-00 B4 30 CD 21 86 E0 2E    4/30/85..0.!...
1486:01D0  A3 00 00 3D 0A 03 72 05-3D 14 03 76 0E BA 3D 04   ...=..r.=..v..=.
1486:01E0  0E 1F B4 09 CD 21 06 33-C0 50 CB B4 0D CD 21 BE   .....!.3.P....!.
1486:01F0  80 00 AC 8A C8 32 ED E3-27 AC E8 88 FF 75 05 E2   .....2..'....u..
1486:0200  F8 EB 1D 90 4E 83 F9 02-72 23 80 7C 01 3A 75 1D   ....N...r#.|.:u.
1486:0210  46 46 83 E9 02 24 DF 2C-40 32 E4 E8 81            FF...$.,@2...
-
```

The DEBUG display is in two parts. The left side displays the data in hexadecimal form (base 16) and the right side shows the character translations in ASCII. You can keep on pressing D as long as you want to continue displaying a file. Use the DEBUG command Q (quit) to exit from DEBUG.

Because DEBUG was designed for advanced DOS users (those who don't mind getting their feet wet), novice users typically find that it is too technical and complicated. Caution is advised before using DEBUG. You should back up the file you are changing or you may find yourself stepping off into deep water. Fortunately, when an application requires the use of DEBUG, detailed instructions are normally provided. The introduction to DEBUG in this chapter should make it easier to use and understand. Chapter 11 illustrates several uses for DEBUG. If you want additional information on this high-powered utility program, refer to your DOS manual.

Review Questions

10

1. What is the purpose of FASTOPEN?
2. What is a TSR program?
3. How does FASTOPEN improve disk performance?
4. What is the purpose of the PRINT command?
5. What is the purpose of the time slice and buffer size options?
6. What is the function of a print spooler?
7. How does a print buffer work?
8. What type of files can be printed with the PRINT command?
9. What are the limitations of PRINT?
10. What are the two forms of operation of the REPLACE command?
11. What effect does the /P option have for the REPLACE command?
12. What does the following command do?

```
REPLACE C:\WORD\BOOK\*.DOC A:/A
```

13. Under what circumstances would the SUBST command be useful?
14. How do you disconnect a current substitution?
15. How can you instruct DOS that you have a disk drive with a drive designator greater than Drive E?
16. What kinds of operations require the use of the DEBUG utility program?
17. What is the DEBUG prompt?
18. Identify two DEBUG commands and their function.

Chapter 10 Lab Exercises

1. If you are currently connected on-line to a printer, you can complete this portion of the exercise. Otherwise, just read through it to see how the PRINT command would execute.

   Without special software, concurrent processing with DOS is limited to two operations at a time. This can be accomplished by executing one operation while printing using the PRINT command. Try a concurrent operation by first entering `PRINT B:TEST3.DIR`. Identify the printing device as PRN when prompted by DOS.

   As soon as printing begins, enter `DIR ¦ SORT ¦ MORE`. DOS overlaps execution of these operations, using somewhat less time to finish the concurrent processing than if they were executed separately.

2. There are many small utility programs you can purchase that accomplish a variety of common tasks. One such task is to issue a form feed command to the printer that causes the paper to advance to the top of the next page. Most printers recognize Ctrl-L as a form feed command. Generate a batch file (**TOF.BAT**) on Drive B that contains the three commands below. If you are attached to a printer, run B:TOF to verify that it works.

```
ECHO OFF
REM  BATCH FILE (TOF) TO ADVANCE PAGE TO TOP OF FORM
ECHO (press Ctrl-L here) >PRN
```

3. Assume you have been creating a number of files that contain tests you want to keep updated on a backup subdirectory. Normally the original test files would be stored on a hard disk and the backup copies would be saved on a floppy disk. However, not all students using this text have a hard disk. Therefore, this exercise is designed to store the backup copies from the root directory of Drive B to a subdirectory of the disk in Drive B.

   - Using the MD command, create a subdirectory on Drive B called \BKUP.
   - Create a small file with EDLIN called B:\BKUP\TEST.EX3 that contains 3 lines of text as follows:

This is a test of using the REPLACE command for backups. The system date will show when this file was created — today's date.

- TYPE the file you just created to verify it was saved correctly.

  Note: Skip to exercise 4 (experimenting with SUBST) if you are not using DOS Version 3.2 or higher.
- From the A> system prompt, use the following REPLACE command to copy (add) all test files except TEST.3 from the root directory of Drive B to the \BKUP subdirectory on Drive B (do not replace TEST.3 when prompted):

```
A> REPLACE B:\TEST*.* B:\BKUP /A/P
```

- Now get a directory listing of B:\BKUP. It should look like Figure 10-3. The dates shown will vary, but TEST.EX3 should be the only file with the current system date.
- Now reenter the previous REPLACE command to copy (add) the remaining test file (TEST.3) to \BKUP. This time it should only prompt you to add one file, because the remaining test files were added during the first REPLACE operation. Get a final directory listing to verify that all test files are on \BKUP. On this directory listing TEST.3 should be shown as the last file added.

Figure 10-3

Screen Display of B:\BKUP after REPLACE Command

```
A>dir b:\bkup

 Volume in drive B is YOUR NAME
 Directory of  B:\BKUP

.            <DIR>       12-19-89   1:30p
..           <DIR>       12-19-89   1:30p
TEST     EX3       132   12-19-89   1:33p
TEST1    DIR      1821   11-27-89   2:28p
TEST2    DIR      1821   11-27-89   2:28p
TEST3    BAK      1821   11-27-89   2:28p
TEST5    DIR      1821   11-27-89   2:28p
TEST     1       25532   10-31-88   2:54p
TEST     4         481   12-15-89  11:44a
        9 File(s)      68608 bytes free

A>
```

4. If you are working with DOS 3.1 or higher you can do this exercise, which experiments with SUBST. Make sure that your system prompt displays the default disk and the current directory before you proceed. If Drive D is already assigned on your computer, use Drive E as the substitute drive.
   - Use SUBST to substitute Drive D for the subdirectory called \BKUP on Drive B. At the system prompt enter `SUBST D: B:\BKUP`.
   - Change the default disk drive by entering D: at the system prompt. Then get a directory listing to verify that the substitution was made.
   - Change back to the original system prompt (A>) and enter SUBST to direct DOS to list the current substitutions in effect. Then enter `SUBST D: /D` to deactivate Drive D as the substitution for B:\BKUP.

5. Bonus exercise:
   - Create an executable program that toggles off the Num Lock key on your keyboard. Then this program could later be included in an AUTOEXEC.BAT file to automatically turn off the Num Lock key every time you boot your system. This program, called **NONUM.COM**, is easily created in two rather unusual steps as follows:
     1) The first step is to use EDLIN to create an ASCII file called **NONUM.DAT**. Because this file represents an assembly language program, the 11 specific statements to be entered are provided for you in Figure 10-4. Enter them exactly as shown, including the blank line after line 5.
     2) The next step is to enter `DEBUG <NONUM.DAT` at the system prompt. This command executes DEBUG, redirecting as input the 11 statements required to create the executable file NONUM.COM.
   - After you create NONUM.COM, test it out on your computer with different settings of the Num Lock key. Whenever you enter NONUM, the Num Lock key is toggled off. Depending on your keyboard, however, the Num Lock status light may not be affected by this command.

Figure 10-4
NONUM.DAT

```
a 100
mov ax,0000
mov ds,ax
and byte ptr[0417],df
int 20
 (blank line)
rcx
c
n nonum.com
w
q
```

This concludes Chapter 10 Lab Exercises. When you are finished, remove your disks.

Chapter

11

ADVANCED DOS BATCH FILES

ADVANCED BATCH FILE COMMANDS

- ☐ IF Command
  FOR Command
  CALL Command

CREATING EFFECTIVE BATCH FILES

- ☐

Chapter 11

Advanced DOS Batch Files

This chapter deals with applying advanced batch file commands to create sophisticated batch files. Not all users agree on exactly what constitutes a sophisticated batch file, but they do agree that batch files can be very powerful and useful. The batch file techniques presented in this chapter should prove beneficial to you.

ADVANCED BATCH FILE COMMANDS

Batch files have their own set of commands. Three of these, REM, PAUSE, and ECHO, were covered in Chapter 5. This chapter explores the following advanced batch file commands:

- IF and GOTO (commands for conditional processing)
- FOR (a mechanism for repeating operations)
- CALL (used to execute another batch file from within a batch file)

In addition, the last section of this chapter discusses various techniques for creating effective batch files, demonstrated in a variety of practical batch files. Once you understand these advanced batch file techniques, you can begin creating your own sophisticated batch files.

IF (Batch File Command)

Format: `IF [NOT] condition command`

The **IF** statement allows for selective execution of a given command. The designated command is executed only when the condition is true. Otherwise, DOS executes the statement in the next line of the batch file. The NOT condition is interpreted in reverse, causing the designated command to be executed only when the condition is false.

The most common use of the IF statement gives batch files the flexibility to branch, depending on the various processing conditions that occur during execution. A GOTO directive is normally used as the designated command in the IF statement. The GOTO directs DOS to branch to another location of the batch file. Branch locations are called labels. A label is a batch file line that starts with a colon (:) and is followed by a name one to eight characters long, such as :OK or :STOPHERE. Branch location names can be longer than eight characters, but only the first eight are used by DOS. Thus, a long label (i.e., :SORTDIR.BAT BY R.J. KING 02/14/90) can be used as a remark that does not print when Echo is on.

Perhaps the most difficult part of using the IF statement is formulating the conditions used to result in the desired branching. There are three different categories of IF statement conditions: EXIST file, ERRORLEVEL number, and string1 == string2 condition. The paragraphs that follow describe these conditions and their use in the IF statement.

EXIST File Condition

If a required file does not exist, the **EXIST file condition** lets you direct further processing of that batch file. Appropriate measures can be taken to locate the correct file before proceeding. If the required file exists, the EXIST file condition can be evaluated as true and the batch file can branch to a location that can continue normal execution. Figure 11-1 illustrates a set of batch file commands used to verify that a file is available before continuing.

Figure 11-1
IF EXIST File Condition

```
:LOOP
IF EXIST A:FILEA.DOC GOTO OK
PAUSE PUT CORRECT DISK IN DRIVE A
GOTO LOOP
:OK
-
- (remaining batch file commands)
-
```

If A:FILEA.DOC cannot be found by DOS, the batch file pauses and directs the user to place the correct disk in Drive A. It then branches back to the label :LOOP and resumes checking for the file. Once found, the IF statement branches around the error condition commands and begins executing the set of commands following the label :OK.

The use of replaceable parameters gives added flexibility to batch files. In this example, %1 could have been substituted for the filename A:FILEA.DOC. In other words, the IF statement in Figure 11-1 could be changed to:

```
IF EXIST A:%1 GOTO OK
```

ERRORLEVEL Number Condition

The **ERRORLEVEL number condition** evaluates as true when the previously executed batch file command has an error condition equal to or greater than some number specified. DOS reserves a buffer in RAM to keep track of error level codes. These codes vary from 0 to 255. The error level buffer is normally updated by application programs. Only DOS commands like FORMAT, BACKUP, and RESTORE update this buffer. A value of zero usually means that the preceding command was executed successfully.

Because error level codes vary by application, an error level of 1 is normally used to indicate that some error has occurred. If an error has occurred, you could direct DOS to skip to the end of the batch file without executing the remainder of the commands. The following portion of a batch file demonstrates how the ERRORLEVEL condition is used:

```
FORMAT B:/S
IF ERRORLEVEL 1 GOTO END
-
-(additional batch file commands)
-
:END
```

This IF statement directs DOS to branch to the :END label in the batch file if any error in the format operation occurred. If the error level were zero (signifying no errors), DOS would continue normally to the next batch file command.

String1 == String2 Condition

Whenever you execute a batch file with replaceable parameters, error checking is highly recommended to ensure that the required parameters were provided. DOS does not do any error checking automatically, but it does allow you to check for missing parameters with the **string1 == string2 condition**. This condition is evaluated as true when the two strings provided are identical. Two equals signs (==) are required in a batch file to represent the equal condition.

The technique for using the string1 == string2 condition is best explained with an example. If a replaceable parameter is missing in a batch file, you could direct DOS to branch to another location with the following IF statement:

```
IF "%2"=="" GOTO ERROR
```

In this example, string1 is the value of the replaceable parameter (%2) and must be enclosed in quote marks. It is compared with string2, a null value (or nothing), which is also enclosed in quote marks. It is only when the parameter is missing that the first string equals the second string.

The IF statement gives you a great deal of flexibility to control the proper execution of batch files. Perhaps you have already thought of some ways to use the IF statement in other batch files. Additional examples of the IF statement are included in the exercises at the end of the chapter. But the IF statement is not the only batch file command that can be used to increase the power of batch files. The FOR command, discussed next, lets you repeat a command using variable parameters.

FOR (Batch File Command)

Format: `FOR %%variable IN(set) DO command`

The variable is a one-letter name preceded by two percent signs (%%). The set represents successive values that the command (specified after the DO) will act on. Each value in the set must be separated by a space or a comma. For example, the following batch file statement deletes all .BAK files in \WORD, \LOTUS, and \HIST:

```
FOR %%G IN(\WORD,\LOTUS,\HIST) DO DEL %%G\*.BAK
```

In this example, the three values in the set are substituted for the variable in the DEL command. The variable used here is the letter G. As a result, three DOS commands are executed as follows:

```
DEL \WORD\*.BAK
DEL \LOTUS\*.BAK
DEL \HIST\*.BAK
```

The **FOR** statement adds significant capability to your bátch files. It lets you perform a single DOS command on a group (or set) of filenames. This is very beneficial because some DOS commands (like FIND and TYPE) do not accept wildcard characters. Without the capability of the FOR statement, those DOS commands must be entered multiple times to be executed with multiple files. As previously demonstrated, wildcard characters can also be used with the FOR statement, adding to its power and flexibility.

Suppose you wanted to copy all executable files from one disk to another, but wanted them copied in sequence by filename extension. Specifically, you wanted to copy all COM files, then all EXE files, and finally all BAT files. You could create the following batch file called KOPY.BAT:

```
ECHO OFF
FOR %%F IN(COM,EXE,BAT) DO COPY *.%%F B:
ECHO COPY COMPLETE
```

You can obtain added flexibility by making the set of filename extensions replaceable parameters. In other words, you could substitute (%1,%2,%3) for (COM,EXE,BAT) in the FOR statement. It is not mandatory that you use all the replaceable parameters in a set. Thus, the command KOPY DOC TXT would copy all the .DOC files from the current directory to Drive B, followed by a copy of all the .TXT files.

Another interesting example is one that allows you to change to different subdirectories on your hard disk without having to find and enter the backslash key. Anyone who frequently uses different keyboards knows how frustrating it can be to locate the backslash key. To avoid this problem, enter the following FOR statement in a batch file called TO.BAT:

```
FOR %%X IN(\%1 %2 %3 %4) DO CD %%X
```

In this example, the four values in the set are delimited by a space rather than a comma. Be sure to place this batch file in a subdirectory that is specified in your current path, so DOS will know where to find it from any directory. To change to the subdirectory \WORD\MEMOS\HIST from anywhere on the hard disk, you could enter TO WORD MEMOS HIST. In effect, this would execute the CD command three times as follows:

```
CD \WORD
CD MEMOS
CD HIST
```

In addition to being used in a batch file, FOR can also be entered as an interactive command at the DOS system prompt. It does not have to reside in a batch file. When the FOR variable is used as a DOS command, only one % is required to identify the variable name. For example, to list the contents of all batch files on Drive B, you could execute the following FOR statement at the system prompt:

```
A> for %x in(b:*.bat) do type %x
```

Although the FOR statement is quite versatile, letting a single command process many files, there is one minor limitation. You cannot use the FOR statement with redirection. This limitation applies to FOR used in a batch file or as a DOS command. The following statement is invalid because the output of TYPE is being redirected to the printer:

```
A> for %x in(b:*.bat) do type >prn %x
```

In Chapter 5 we learned that other batch files can be executed from within a batch file. However, when you enter just the batch file name for execution, DOS transfers control to that batch file with no way to return to the original batch file. This is okay if the next batch file you want to execute is entered as the last statement in the original batch file. But it becomes a major limitation if the statement used to execute the next batch file is not the last statement. The CALL statement was added to Version 3.3 of DOS to resolve this problem.

CALL (Batch File Command)

Format: `CALL [d:][path]filename [parameters]`

You can use **CALL** to specify a second batch file to be executed anywhere in the first batch file. You can also include any parameters required by the second batch file. When the second batch file is finished, DOS continues processing the remaining statements in the first batch file. However, you cannot redirect input or output with the "called" batch file.

If you have an earlier version of DOS you can duplicate the same capability by using the COMMAND/C batch file command instead of CALL. Entered in the same format, COMMAND/C makes a duplicate copy of DOS in memory for the express purpose of executing the second batch file. For efficiency reasons, the CALL command is preferred over COMMAND/C. Refer to the DOS manual for further information on these two commands.

EXAMPLES OF USAGE:

```
CALL SUPERMAT
```
(executes the batch file SUPERMAT.BAT with no replaceable parameters, returning the control back to the original batch file when SUPERMAT is finished)
```
CALL BACK \WORD\LTRS\*.DOC B:
```
(executes BACK.BAT with the two parameters specified)
```
COMMAND/C SORTDIR A: SIZE
```
(makes one extra copy of DOS before executing the batch file SORTDIR.BAT with the two parameters specified)

CREATING EFFECTIVE BATCH FILES

The first question to ask is, what is an effective batch file? Any batch file that improves your ability to be productive and minimizes errors is a likely candidate. Many techniques can be used to create effective batch files. Here we discuss several of these methods to assist you in creating better batch files.

You should create batch files that give you added protection from making a mistake. For example, a listing of files you want to delete should always be displayed on the screen before you delete them with wildcard characters. The following batch file (**PURGE.BAT**) can be used as an effective substitute for the DEL command:

```
ECHO OFF
REM  IMPROVED DEL COMMAND (PURGE.BAT)
DIR  %1/W
ECHO THESE FILES WILL BE DELETED (PRESS CTRL-BREAK TO ABORT)
PAUSE
DEL  %1
```

It is always a good idea to include a REM statement in your batch files that identify the batch file name and briefly describe its use. That way, whenever the batch file is displayed on the screen, the user is reminded of its function.

ECHO OFF is normally included as the first statement in a batch file to suppress batch file commands from being displayed on the screen during execution. Beginning with DOS 3.3, if you preface any batch file command with the character @, it does not display on the screen. Thus, to keep the ECHO OFF command from displaying, it can be entered as @ECHO OFF. If you want to get the user's attention when reading displayed messages on the screen, you can use the following technique:

```
@ECHO OFF
ECHO  Be very
ECHO       very
ECHO          very
ECHO             very careful when you use DEL *.*
```

There are times in a batch file when you don't want the user to enter responses to DOS prompts. When you format a disk, change the volume label, or delete all files on a directory, DOS request information from you. If you want to automatically enter a single response to a DOS command, you can utilize ECHO to pipe input to a DOS command. The following statement automatically deletes all files on the current directory without waiting for a response to the "Are you sure" message:

```
ECHO Y ¦ DEL *.*
```

In this example, the response Y is filtered immediately into the operation when needed. However, the DOS message "Are you sure (Y/N)?" still displays on the screen. If you don't want unnecessary DOS messages displayed on the screen when running a batch file command, you can redirect them to NUL. For example:

```
ECHO Y ¦ DEL *.* >NUL
```

If you have multiple responses to enter to a DOS command in a batch file, you must use a different technique. First you need to enter all the command responses as entries in an ASCII text file, one entry line per response. Then the text file can be redirected as input into the DOS command. Suppose you wanted to format only one disk in Drive B with a volume label of DATADISK. In addition, you did not want the execution of the batch file slowed while the user entered responses to the FORMAT command. You could create the following file called **RESP.DAT**, where <cr> represents the Enter key:

```
<cr>
DATADISK<cr>
N<cr>
```

This text file contains the three responses necessary to automatically execute the FORMAT /V command. The first response (an Enter key) directs DOS that the disk is ready to be formatted. The second response supplies the volume label. And the last one signifies that there are no more disks to be formatted. To supply the responses to the FORMAT command in the batch file, redirect this file as input to FORMAT as follows:

```
FORMAT B: /V <RESP.DAT
```

Redirection can also be used to quickly create a one-line batch file that does not contain any piping or redirection. With the ECHO command you can redirect a valid DOS command as input to a batch file. This does not execute the DOS command, but merely places it in the specified batch file. Suppose you wanted to create a batch file that formats a disk on Drive A with the /S option. You could redirect the appropriate DOS command to a batch file called FORMATA.BAT as follows:

```
ECHO FORMAT A:/S >FORMATA.BAT
```

Once the batch file exists, you could append a DOS command after the last line of the file by using the append redirection symbol (>>) instead of the replace redirection symbol (>). For example:

```
ECHO CHKDSK A: >>FORMATA.BAT
```

Now the contents of FORMATA.BAT would be:

```
FORMAT A:/S
CHKDSK A:
```

It is very important that batch files detect errors, especially when replaceable parameters are used. Suppose you wanted a batch file to allow the user to format only Drive A or Drive B, but no other disk drives. In other words, you wanted to have control when anyone entered FORMAT. First you must rename the FORMAT command to something else, like FORMATX.COM. Then you must create a batch file named FORMAT.BAT that executes whenever FORMAT is entered at the system prompt. Figure 11-2 shows you the statements to be included in FORMAT.BAT.

Figure 11-2
FORMAT.BAT

```
ECHO OFF
REM  FORMAT.BAT
ECHO %0
IF "%1"=="" GOTO ERROR
IF %1==A: GOTO OK
IF %1==a: GOTO OK
IF %1==B: GOTO OK
IF %1==b: GOTO OK
ECHO YOU CANNOT FORMAT DRIVE %1
GOTO END
:OK
FORMATX %1
GOTO END
:ERROR
ECHO YOU MUST ENTER THE DRIVE TO FORMAT
:END
```

This batch file demonstrates how you can test for several errors and control the process of formatting disks. Remember, FORMAT.COM was renamed so that whenever FORMAT is entered, the .BAT file is executed rather than the original FORMAT.COM file. A second parameter could easily be added to the end of FORMATX %1 to allow for a formatting option like /S, /V, or /S/V to be included. In any batch file, when you ECHO %0 (parameter zero), the batch file name is displayed on the screen.

Sometimes you may need to execute a batch file differently depending on a user response. In order to do this type of selective branching, you must have the ability to capture a keystroke from the user. DEBUG can be used to create an executable program to do this. Enter the commands shown in Figure 11-3 to create **GETKEY.COM**.

Figure 11-3

Use of DEBUG to Create GETKEY-.COM

```
DEBUG
-E 100 B4 00 CD 16 B4 4C CD 21
-N GETKEY.COM
-RCX
:8
-W
-Q
```

Now that GETKEY exists, you can use it in a batch file to capture a user keystroke. Strange as it may seem, error level codes are used to identify a keystroke entered in response to the GETKEY command. Some of the error level codes you may want to use are shown in Figure 11-4.

Figure 11-4

Sample Error Level Codes

0	=	NUL
7	=	bell (Ctrl-G)
13	=	Enter key
27	=	Esc key
32	=	Space bar
48-57	=	digits (0-9)
63	=	question mark (?)
65-90	=	uppercase alphabet (A-Z)
97-122	=	lowercase alphabet (a-z)

When testing for an error level code in the IF statement, DOS looks for a code equal to or greater than the code specified to evaluate the condition as true. Therefore, to identify the Enter key (error level code 13), you must test

for a error level code that is not equal to or greater than 14, yet equal to or greater than 13. Figure 11-5 illustrates one approach showing how the errorlevel codes can be tested.

Figure 11-5

Testing of Error Level Codes

```
ECHO OFF
:TOP
ECHO  BREAK OUT OF THIS BATCH FILE BY PRESSING THE ENTER KEY ONLY
GETKEY
IF  ERRORLEVEL 14  GOTO TOP     (code is greater than 13 if true)
IF  ERRORLEVEL 13  GOTO CONT    (code must be 13 if true)
GOTO  TOP
:CONT
```

If the error level is 14 or more the keystroke cannot be an Enter key, so the batch file branches to the label named :TOP. If the next statement is evaluated as true, then the keystroke was the Enter key.

An alternative approach, shown in Figure 11-6, is interpreted as follows: If the error level is equal to or greater than 13, but not equal to or greater than 14, then it must be equal to 13.

Figure 11-6

Alternate Method of Error Level Testing

```
ECHO  OFF
:TOP
ECHO  BREAK OUT OF THIS BATCH FILE BY PRESSING THE ENTER KEY ONLY
GETKEY
IF  ERRORLEVEL 13  IF NOT  ERRORLEVEL 14  GOTO CONT
GOTO  TOP
:CONT
```

Although batch file processing adds power and convenience to the operation of a computer, it is not meant to be used as a replacement for a programming language. DOS batch files operate relatively slowly because each command must be interpreted by DOS during execution. However, for the user who does not want to spend long hours learning a programming language, batch files offer an easy way to improve the overall operation of your system.

Review Questions

11

1. What is the main purpose of the IF statement in a batch file?
2. What does a GOTO statement in a batch file do?
3. How can a batch file detect a missing replaceable parameter?
4. How can batch files be used to redefine DOS commands and create your own commands?
5. What is the benefit of using the batch file command @ECHO OFF?
6. What does the following command do?

```
TYPE XYZ.DOC ¦ FIND "ABC" >PRN
```

7. Suppose that WP executes a word processing program and a batch file called DOBOILER.BAT contains the following:

```
CD\WORD
COPY %1 %2
WP %2
```

   What does the command DOBOILER CREDIT.DOC SMITH.DOC likely do?
8. Under what conditions would DOS execute a batch file that has the same name (i.e., CHKDSK) as a DOS command?
9. What is the purpose of the FOR batch file command?
10. What does the following batch file command do?

```
FOR %%Q IN (*.BAT,*.TXT) DO TYPE %%Q
```

11. How can you display the batch file name when it is executed?
12. What happens when you redirect screen output to NUL?
13. How can you automatically enter multiple predefined responses to a DOS command in a batch file?
14. What one-line statement can be used to create a batch file called DELBAK.BAT that erases all .BAK files on the current directory?
15. Specify the IF statement that can be used after a GETKEY command to branch to :END when the number 1 is entered from the keyboard.

Chapter 11 Lab Exercises

1. Rather than purchasing a utility program to search your hard disk for a particular filename, you can create your own customized batch file named PHIND.BAT. You should not name this file **FIND.BAT** because entering the batch file name of FIND might cause DOS to execute FIND.COM. Create the batch file (shown in Figure 11-7) on Drive B without entering the explanatory comments.

   This useful batch file directs CHKDSK to locate every filename on your hard disk and pipe it to FIND. FIND filters out all filenames that do not contain the string of characters specified by the replaceable parameter (%1). Because wildcard characters do not appear in the listing generated by CHKDSK /V, they cannot be used in the replaceable parameter. The IF test allows the batch file to skip over the CHKDSK and FIND operations, if the parameter was not included with the batch file name (PHIND) during execution. Figure 11-8 on the next page shows you what the screen might look like if you executed PHIND.BAT, first with no parameter and then with BAT as a parameter.

   This batch file could take several minutes to execute if there were many files to be piped through the FIND filter. However, just knowing you can create a customized batch file like this should make it a beneficial exercise.

   Use this batch file to display all files on your DOS disk with the characters DIS in the filename. Because CHKDSK pipes all filenames in uppercase letters, the PHIND parameter (DIS) must also be in uppercase.

Figure 11-7
PHIND.BAT

```
ECHO OFF
REM  PHIND.BAT
IF  "%1"==""  GOTO  ERROR   (if no parameter, branch to ERROR)
CHKDSK /V  ¦ FIND  "%1"     (sends all filenames to the FIND filter
GOTO  END                    displaying matching lines on the screen)
:ERROR
ECHO  FILENAME PARAMETER REQUIRED
:END
```

Figure 11-8

Screen Display after Executing PHIN.BAT

```
B:\>phind

B:\>ECHO OFF
THE FILE NAME PARAMETER IS REQUIRED

B:\>phind BAT

B:\>ECHO OFF
      B:\ASORT.BAT
      B:\PHIND.BAT
      B:\AUTOEXEC.BAT
      B:\SUPERMAT.BAT

B:\>
```

Figure 11-9

SUPERMAT.BAT

```
@ECHO OFF
REM SUPERMAT.BAT USED TO CREATE BOOTABLE DISKS
CLS
FORMAT B:/S
COPY FORMAT.COM B: >NUL
COPY CHKDSK.COM B: >NUL
DIR  B:
ECHO END OF SUPERMAT
```

2. Create a batch file (**SUPERMAT.BAT**) that formats a disk with the /S option and automatically copies FORMAT.COM and CHKDSK.COM to the newly formatted disk. This would be a useful program to format floppy disks that are bootable and contain the FORMAT and CHKDSK commands. Specific application programs could later be copied to these bootable disks. The batch file commands you should enter are shown in Figure 11-9. When you execute this batch file, you will still have to respond to several prompts while the FORMAT command is being executed.

   If you are working with an earlier version of DOS than DOS 3.3, omit the @ in the first batch file command. When you redirect the screen output of the COPY commands to NUL, the DOS message stating that a file was copied is suppressed. If you have a blank disk handy, execute this program to verify that it works correctly.

3. Modify the batch file you created in exercise 4 of Chapter 8, (**BACK.BAT**), so that it can be used as a feasible replacement for the BACKUP command. The modified batch file (see Figure 11-10) uses the GETKEY command discussed earlier in this chapter and two replaceable parameters. To test this batch file, enter `BACK B:\WORD\MEMO\*.DOC A:`.

Figure 11-10
BACK.BAT Modified

```
@ECHO OFF
REM BACK.BAT WITH 2 REPLACEABLE PARAMETERS
REM    1) SUBDIRECTORY & FILES YOU WANT TO BACK UP
REM    2) TARGET DISK FILE DESIGNATOR
CLS
ATTRIB  +A  %1
:LOOP
ECHO  PLACE TARGET DISK IN DRIVE %2
PAUSE
XCOPY  %1  %2  /M
ECHO  IF NO MORE FILES TO BE COPIED (NO ERROR MESSAGES), PRESS ESC
ECHO  OTHERWISE, INSERT ANOTHER DISK & PRESS ANY KEY TO CONTINUE
GETKEY
IF ERRORLEVEL 27 IF NOT ERRORLEVEL 28 GOTO END
GOTO LOOP
:END
CLS
DIR %2/W
ECHO END OF BACK UP OPERATION
```

4. Create a batch file called **SORTDIR.BAT** (see Figure 11-11 on the next page) that allows the user to specify the sequence of the directory listing. The available options are to sort by filename, by extension, or by file size. This batch file does not require the user to know the directory listing column number of the category to be sorted.

   After you create this batch file, test it using several valid and invalid parameters. For example, entering SORTDIR A: SIZE would cause the batch file to display a directory listing of Drive A in sequence by file size. Perhaps you have thought of other additions you would like to make to this batch file.

Figure 11-11
SORTDIR.BAT

```
@ECHO OFF
REM SORTDIR.BAT
CLS
IF "%2"=="" GOTO ERROR
IF NOT %2==NAME IF NOT %2==EXT IF NOT %2==SIZE GOTO ERROR
GOTO %2
:ERROR
ECHO VALID PARAMETERS ARE NAME, EXT, OR SIZE.
GOTO END
:NAME
DIR %1 ¦ SORT ¦ FIND "-" ¦ MORE
GOTO END
:EXT
DIR %1 ¦ SORT /+10 ¦ FIND "-" ¦ MORE
GOTO END
:SIZE
DIR %1 ¦ SORT /+17 ¦ FIND "-" ¦ MORE
:END
```

5. Create a batch file that can be used as an on-line HELP command. Whenever you entered HELP followed by a DOS command, it would display the format of the command on the screen. But first you must use EDLIN to create an ASCII file (**FORMAT.TXT**) containing the formats for all the DOS commands you want included in the on-line help. For example:

```
ATTRIB [+/-A] [+/-R] [d:]filename[.ext]
BREAK [ON/OFF]
XCOPY [d:]filename[.ext] [d:][filename[.ext]] [/M] [/P] [/S]
```

Then create a batch file (**HELP.BAT**) that lets a user specify one of the DOS commands by name and have the format of that command displayed on the screen. You must use the FIND command and replaceable parameters to complete this helpful batch file. When you complete the batch file, shown below, test it using some commands you entered in the .TXT file. For example: HELP XCOPY.

```
ECHO OFF
REM  HELP.BAT
CLS
ECHO  LOCATING FORMAT OF THE %1 COMMAND
FIND  "%1" FORMAT.TXT
```

6. Bonus exercise:

   Sometimes experienced users like a certain amount of protection from novice users accessing their system without permission. Some type of password protection system can help. Although few protection systems can deter an experienced user, Figure 11-12 shows you a batch file that can be placed in your AUTOEXEC.BAT file. This batch file will stop most novice users from getting past the boot process without knowing the password.

   Create **PASSWORD.BAT** and test it by executing the batch file name. You can use Figure 11-4 to determine how to successfully get past this password protection. If you later execute this batch file from your AUTOEXEC.BAT file, don't forget to use the CALL command. This command allows DOS to return to AUTOEXEC.BAT after executing PASSWORD.BAT.

   Have you determined the purpose of the first PROMPT command in this batch file yet? If you terminate execution of PASSWORD.BAT with Ctrl-Break, the reason may become evident to you.

Figure 11-12

Protection Batch File (PASSWORD.BAT)

```
@echo off
rem password.bat
prompt Fatal Disk Error #23 - Reboot
:top
cls
echo  Unrecoverable read error on drive C:
echo  Abort, Retry, or Fail?
getkey
if  errorlevel 63  if not  errorlevel 64  goto end
:cause the disk to spin for a second
copy  command.com  command.bak  >nul
del  command.bak  >nul
goto  top
:end
prompt  $p$g
```

This concludes Chapter 11 Lab Exercises. When you are finished, remove your disks. We hope you enjoy creating some useful and powerful batch files!

Chapter

12

UTILITY SUPPORT PROGRAMS

EXTENDING DOS

- ☐ Improvements to DOS
  Additions to DOS

DOS SHELLS, THE FRIENDLY APPROACH

- ☐ Terminate and Stay Resident Programs
  The Benefits of Using Windows

COMMERCIAL UTILITY PROGRAMS

- ☐ The Norton Utilities Advanced Edition
  SideKick Plus
  Treeview
  PC Tools Deluxe
  Other Commercial Utility Programs

NONCOMMERCIAL PROGRAMS

- ☐ Freeware, Public-Domain, and Shareware Programs
  Sampling of Utility Support Programs

CONSIDERATIONS, CAUTIONS, AND CONCLUSIONS

- ☐

Chapter

12

UTILITY SUPPORT PROGRAMS

DOS supplies many useful commands and programs to assist us in controlling the computer and making it an effective tool. But DOS clearly falls short in meeting all our computing needs. Frustration with DOS is a fairly common occurrence. It results from misplaced files, awkward commands, and a general lack of capabilities. Yet for under $100, you can supplement DOS and overcome many of its shortcomings. This chapter discusses how utility support programs can be used to extend the capabilities of DOS. It covers existing DOS functions that are improved as well as new capabilities that are added.

EXTENDING DOS

There are thousands of additional utilities, obtainable from many different places, that extend the power and capabilities of DOS dramatically. They can be used to improve the performance of DOS, facilitate running an application program, and increase control over peripheral devices such as a keyboard, printer, or disk drives. Utility programs are probably the fastest growing category of microcomputer software.

Utility programs can be classified into two groups, depending on their use of main memory. Some utility programs are similar to DOS external commands. They require a certain amount of memory to run and then release it back to DOS when they are finished. Other programs are like DOS internal commands. They are loaded once and stay in RAM until you power down the system. These programs are called terminate and stay resident (TSR) programs, a concept covered shortly.

Another way to look at utility programs is by their function. In an overall sense, utility programs either improve an existing DOS capability, or they supplement DOS with additional capabilities. In either case, they have found their place in a DOS world.

Improvements to DOS

Many utility programs improve DOS by making it easier to use. They utilize techniques like menus, windows, and DOS shells to assist the novice user through the execution of tasks. Windows and DOS shells are covered in detail later in this chapter and menus are discussed in the next chapter.

In addition, there are a number of DOS commands that are too limited, like TREE, PRINT, BACKUP, and RESTORE. Numerous companies cashed in on the weaknesses of DOS commands by developing improved backup and restore programs.

Most of your interaction with a microcomputer is done by typing on the keyboard, and much of the typing is repetitive. But the method DOS uses to redefine the keys on the keyboard is primitive compared with the methods used by utility programs. Rather than retype complex commands each time, you can use a utility program called a **macro generator**. A macro is a single command that represents a sequence of commands.

Even a simple DOS command like DIR can be significantly improved. DOS displays only one subdirectory at a time. Sequencing the directory listing by name, extension, size or date is cumbersome, and unless you remember to use the /P option, file names fly by too quickly to read. Fortunately there are many utility programs that overcome these limitations.

Additions to DOS

In addition to improving DOS functions, utility support programs add new features. These programs provide capabilities that many users think should have been included in DOS in the first place. Early utility programs were written to include the following additions to DOS:

- Recover files that have been accidentally deleted.
- Browse up and down through any ASCII file a page at a time.
- Modify all file attributes (hidden, system, archive, and read-only).
- Find a file anywhere on a hard disk from any subdirectory.
- Blank the screen during periods of inactivity to reduce screen burnout.

As more and more users needed increased capabilities, small companies formed to fill the void. They developed more sophisticated programs that provided for the following enhancements:

- Execute multiple DOS commands or programs concurrently.
- Optimize the performance of disks and disk access.
- Improve productivity via a desktop manager.
- Generate and complete complex business forms.

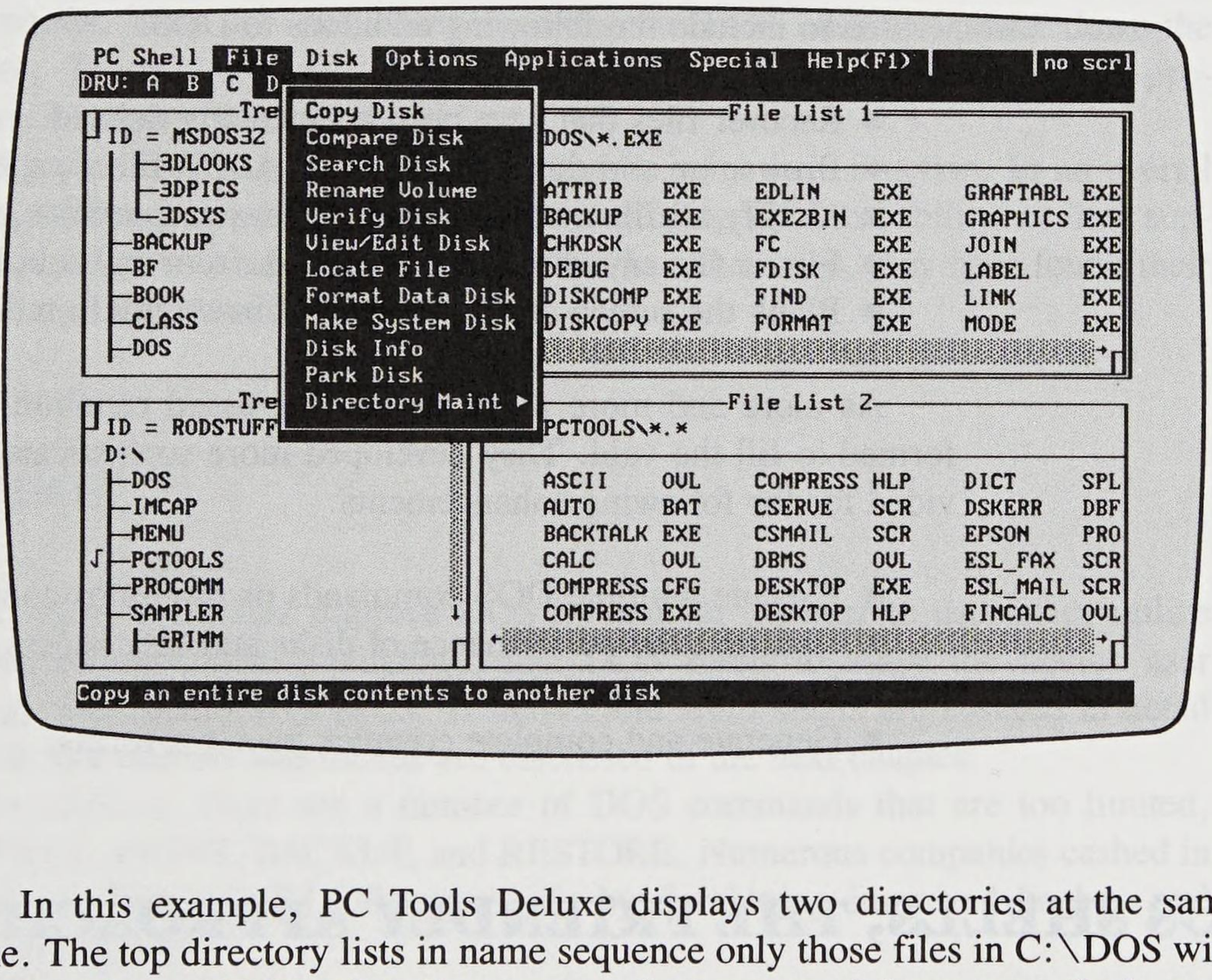
DOS SHELLS, THE FRIENDLY APPROACH

To beginning DOS users, the wide range and complexity of DOS commands can be intimidating. In addition, DOS has no on-line help facility to assist them when they need it. It is only after a great deal of effort that you become comfortable using DOS. But you can become a proficient user another way—by using a friendly DOS shell.

A **DOS shell**, a popular alternative to DOS, is specifically designed to guide users through the process of working on the computer. It is an interactive layer between you and the operating system. The shell's extra processing layer extends DOS by adding features that DOS doesn't have. Although DOS shells come in a wide variety of flavors and approaches, they have two things in common. They are normally RAM-resident programs and the shell employs a window environment to help users. A DOS shell window (covered later) makes the DOS commands easier to execute by giving you a visual representation of the operating system and disk structure. Figure 12-1 shows a screen using two windows from PC Tools Deluxe, a powerful DOS shell utility program.

Figure 12-1

PC Tools Deluxe Screen



In this example, PC Tools Deluxe displays two directories at the same time. The top directory lists in name sequence only those files in C:\DOS with an extension of EXE. The directory on the bottom of the screen displays all filenames in D:\PCTOOLS in name sequence. A pull-down menu of available disk operations is superimposed on top of these directories. The top of the screen displays the main menu selection of activities for the DOS shell, as well as information relating to the current disk. PC Tools Deluxe is discussed in more detail later in this chapter.

Terminate and Stay Resident Programs

Many utility support programs are RAM-resident, which means that they stay in RAM until you need them. These programs are referred to as **terminate** and **stay resident (TSR)** programs. Some TSR programs are activated automatically when a particular event occurs, such as when the system prompt is displayed. Others are activated in response to a particular keystroke, called a hot key, even while you are using another program. TSR programs are fast and convenient, but they require extra space in main memory. To accommodate these types of programs, today's microcomputer systems have a minimum of 640KB of RAM, and often contain several megabytes of RAM.

Using TSR software carries some dangers, especially if you are using more than one TSR program. Some application programs take control of the keyboard so tightly that they don't allow you to access your TSR programs. TSR programs can take up valuable space that might be better allocated to the main program. Finally, some TSR programs try to utilize the same area in RAM and may cause your system to lock up. Sometimes this problem is resolved by shuffling the order in which TSR programs are loaded in RAM.

The Benefits of Using Windows

A **window** is a portion of the display screen devoted to a particular task, such as a subdirectory listing, a set of commands to be executed, or the activities of an application program being executed. It is possible to divide your screen into many windows; some can be side by side, others can overlay existing windows much like a cascading set of menus overlay each other as they are selected. The windows concept is an important one to understand because many utility programs and application programs employ this concept.

The most common benefit of windows is to relieve users of the burden of having to remember so much. DOS commands and their associated options are built into windows and can be selected from the screen for executing. In addition, directory listings from multiple subdirectories can be displayed to assist users in locating and identifying files to be processed. Directory listings may be displayed in a variety of user-defined sequences.

A second benefit of windows is generally associated with a desktop manager. It allows a user to create windows that can be used to execute a desktop accessory. Desktop accessories include such useful tasks as writing notes to

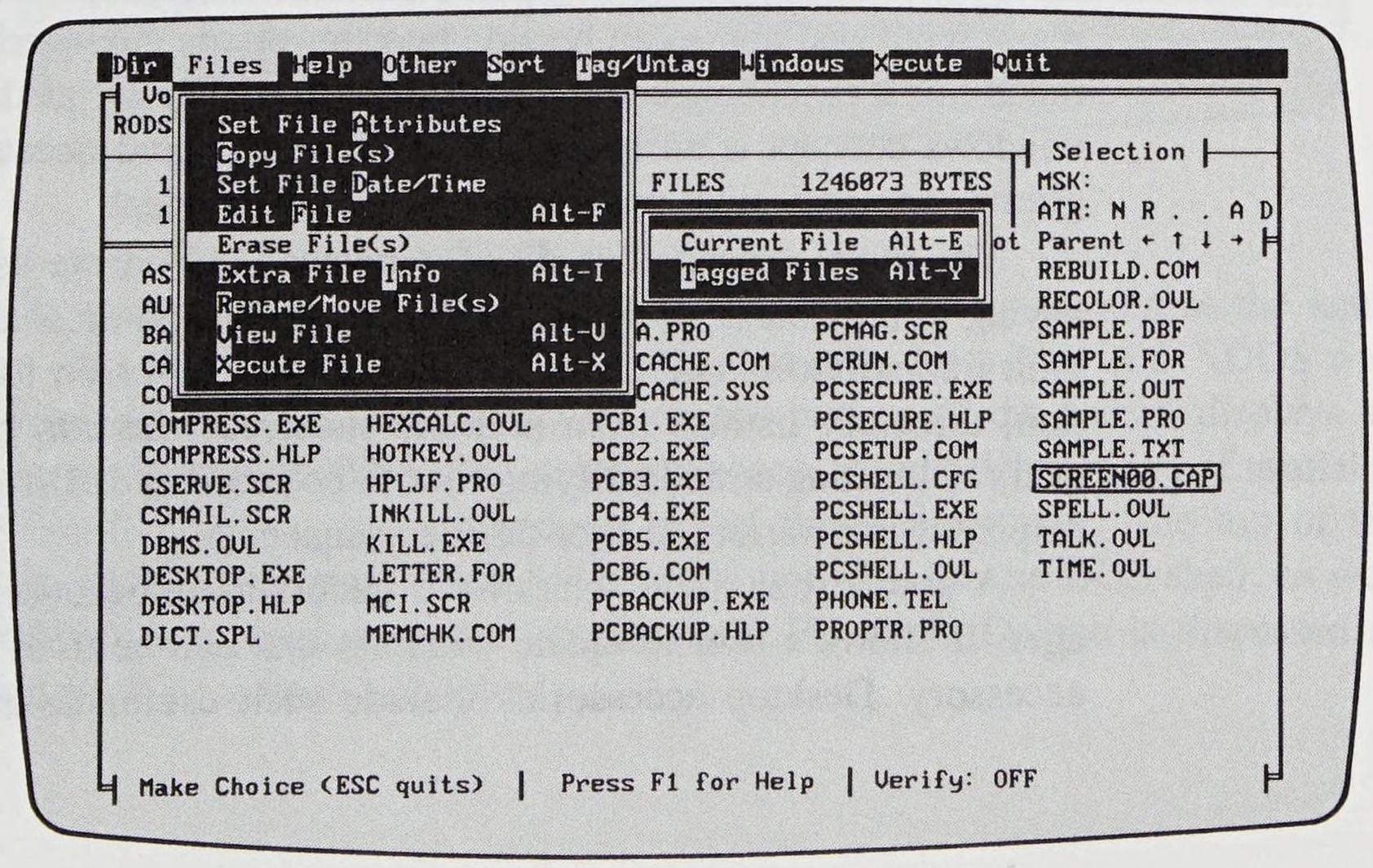
yourself, making complex calculations with a powerful calculator, and scheduling appointments on your calendar. Desktop windows are an example of a pop-up application program because they can be instantly "popped up" on the screen with a hot key when you are running another program. A **hot key** is a unique combination key, like Ctrl-Alt, that activates a TSR program.

Another major benefit of windows is that they provide the ability to run several different programs concurrently. For example, you can be entering text in a word processing program while a spreadsheet program is merging several spreadsheets together and doing all the required calculations. In addition, the windows approach to processing greatly facilitates the exchange of information between different programs. With windows you can pass the information from one program to another without having to create a temporary disk file. This feature dramatically simplifies consolidating data from many different programs. Microsoft's Windows is an outstanding example of a utility program that embodies all the benefits of windows discussed above.

As a graphical extension of DOS, the windows approach gives you a highly visible way to work and to organize your work. Several common approaches are typically available to assist users. These include pull-down menus, the tagging concept, and the point and shoot method of operation. A **pull-down menu** is a small window superimposed on the screen showing the related choices for a given major activity type. For example, the pull-down menu activities related to a file-based activity generally include copy, erase, edit, rename, view, and so on. Figure 12-2 illustrates a pull-down menu from Treeview, a friendly DOS shell and file management utility program.

Figure 12-2

Treeview Screen with Pull-Down Menu



The Treeview screen with the pull-down menu is asking if the user wants to erase just the current file or all tagged files. The use of the **tagging concept** allows a selected activity to be performed on more than one file at a time. Multiple files can be individually tagged and then executed as a group, much as DOS lets you use wildcard characters in commands. This process is often referred to as the **point and shoot approach**, where cursor movement keys (or a mouse) are used to point to a named item displayed on the screen. Users can point to a selected file and then press Enter to tag it. Then they can point to a selected activity and press a command key to execute (or "shoot") that activity for all tagged files. Most applications let the user identify a selected activity with a specific command key (like V for View). The point and shoot method is also used to easily navigate through the maze of subdirectories on a hard disk. Point to the desired directory displayed in a simple graphic form and press Enter (or click the mouse) to change directly to that directory.

COMMERCIAL UTILITY PROGRAMS

Many outstanding commercial utility support programs greatly extend the capabilities of DOS. Most of them employ menus, windows, and on-line help facilities that make them easy to use. Many are RAM-resident programs that allow their capabilities to be available at the touch of a hot key. The four packages discussed below are fairly representative of the kinds of utility support programs that you can purchase for less than $100.

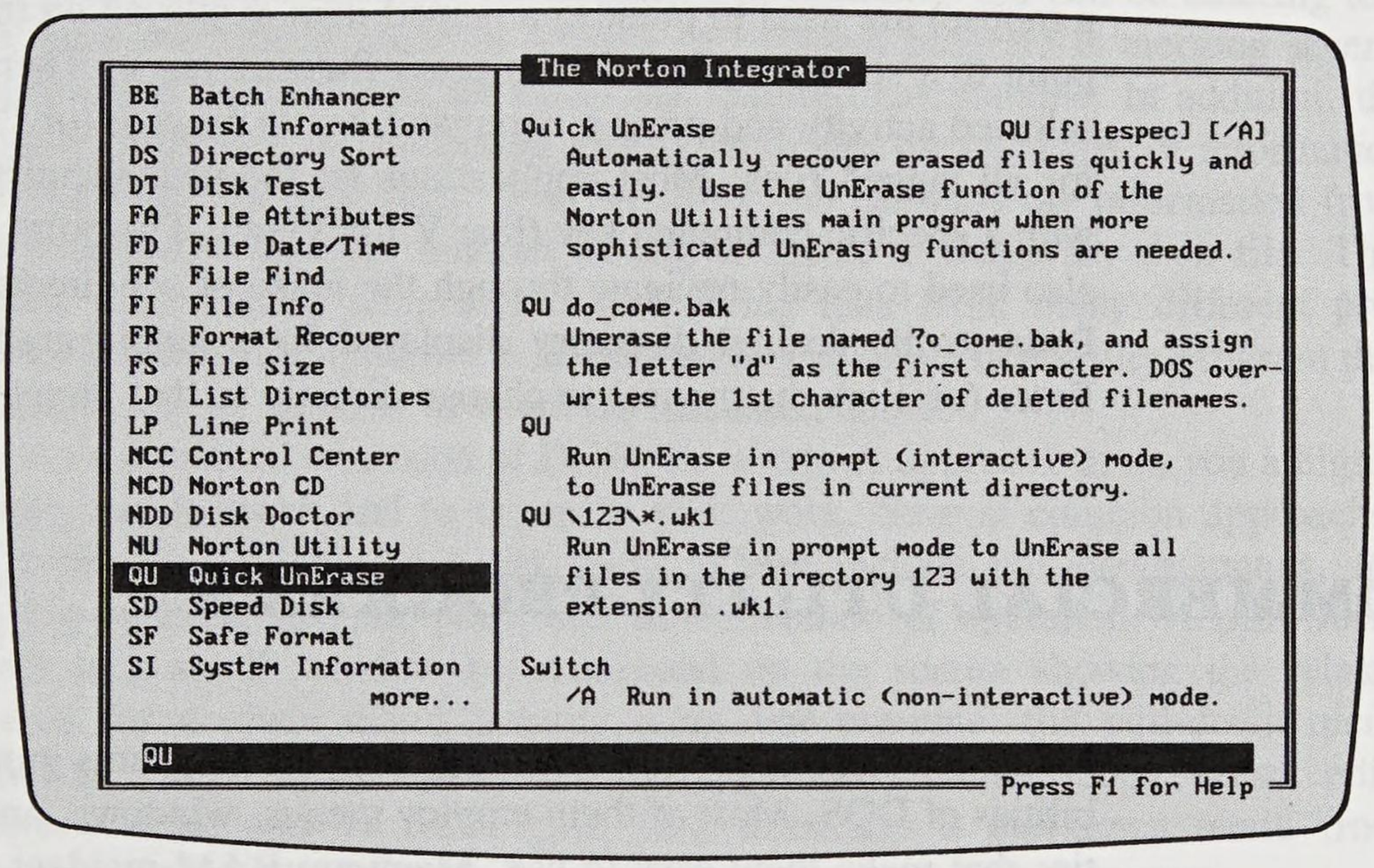
The Norton Utilities Advanced Edition

The Norton Utilities is undoubtedly the world's best-selling disk management package. It is renowned for its data recovery feature that lets you recover recently deleted files. Norton's Computing Index (CI) has become an industry standard for rating the speed of a microcomputer. It measures the computational speed of the CPU relative to the 8088-based IBM PC-XT. With the release of the Advanced Edition, this utility package is certain to continue in popularity.

Novice users will find this package easy to use, thanks to the Norton Integrator, which lets you control everything from a main menu. Figure 12-3 shows the Norton Integrator with the Quick UnErase (QU) program selected.

Figure 12-3

Norton Integrator



The menu of utility programs is displayed on the left side of the screen. The right side of the screen explains the various options for the utility program selected. Over two dozen utility programs can be selected and executed. Some of the new technical features and functions of the Advanced Edition include:

- Speed Disk for condensing disk space and optimizing disk access.
- Format Recovery for unformatting an accidentally reformatted hard disk.
- High-powered editors that allow access and modification of disk sectors.
- Control Center to set system defaults easily, like color and cursor size.
- Disk Doctor to diagnose and correct a variety of disk problems.

The Norton Troubleshooter, a nifty 158-page guide to finding and fixing disk problems, is a helpful bonus included free with the Advanced Edition. The

Norton Utilities are just one of the outstanding products developed by Peter Norton. You can contact the company at:

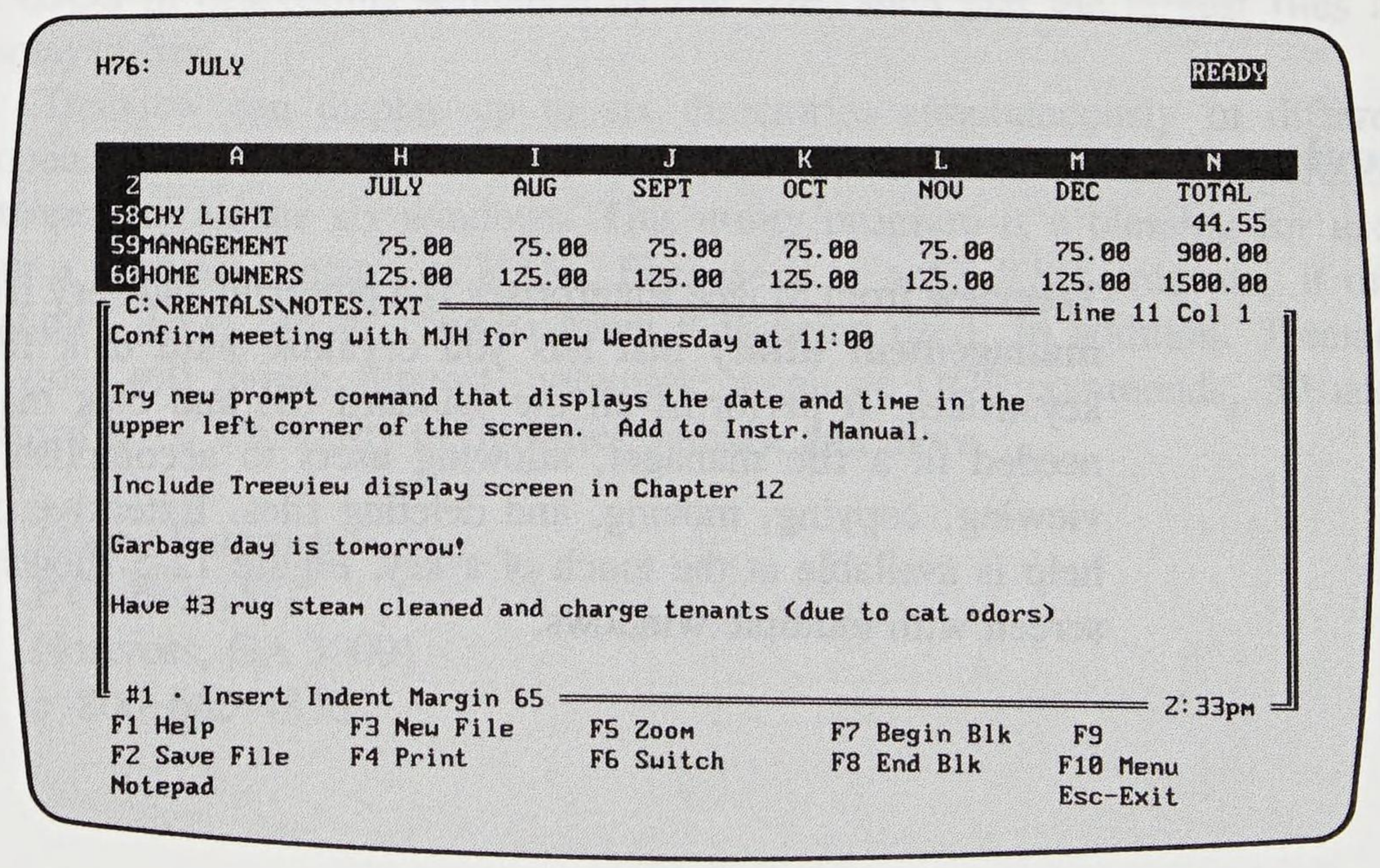
Peter Norton Computing, Inc.
100 Wilshire Blvd., 9th Floor
Santa Monica, CA 90401
1–800–365–1010

SideKick Plus

SideKick Plus is a powerful desktop manager. **Desktop managers** usually operate via windows and encompass the functions of tools normally found on executive desktops, like a phone directory, a financial calculator, and a variety of notepads. SideKick lets you do what you couldn't do in the past—it lets you stay at the keyboard instead of rummaging through your desk when you need something. With a single keystroke, SideKick opens a window to a notepad, calculator, appointment calendar, phone directory with autodial capability, and more. Because it is RAM-resident, the windows appear instantly. Figure 12-4 illustrates a SideKick screen with a notepad superimposed on an Enable spreadsheet.

Figure 12-4

SideKick Notepad Superimposed on a Spreadsheet



Having SideKick is like having your own personal assistant. Suppose you are using your word processing program and need to consult your appointment book or some note you wrote a few days ago. Simply press the SideKick hot key to activate the main menu of options and select the desktop tool you need. When you are finished using SideKick, you instantly return to the program you were previously executing. The capabilities of the calculators available with SideKick will probably surprise you—they have all the capabilities of expensive electronic hand-held calculators.

With the release of SideKick Plus, desktop managers reached a new level of operation for memory-resident utility programs. SideKick Plus has a file management facility to manipulate and catalog hard disk files. It has an outliner to help you organize your thoughts and a notepad facility that provides for basic word processing functions in up to nine active windows. SideKick's phone book is linked to a full-featured communications package. In addition, SideKick Plus contains a clipboard facility that allows data to be transferred between different facilities and application programs.

If you load all the capabilities of SideKick Plus into RAM you may find there is not enough room left to run your application programs effectively. Fortunately, SideKick Plus includes an install command that lets you load only the facilities that you need. SideKick Plus is developed and marketed by:

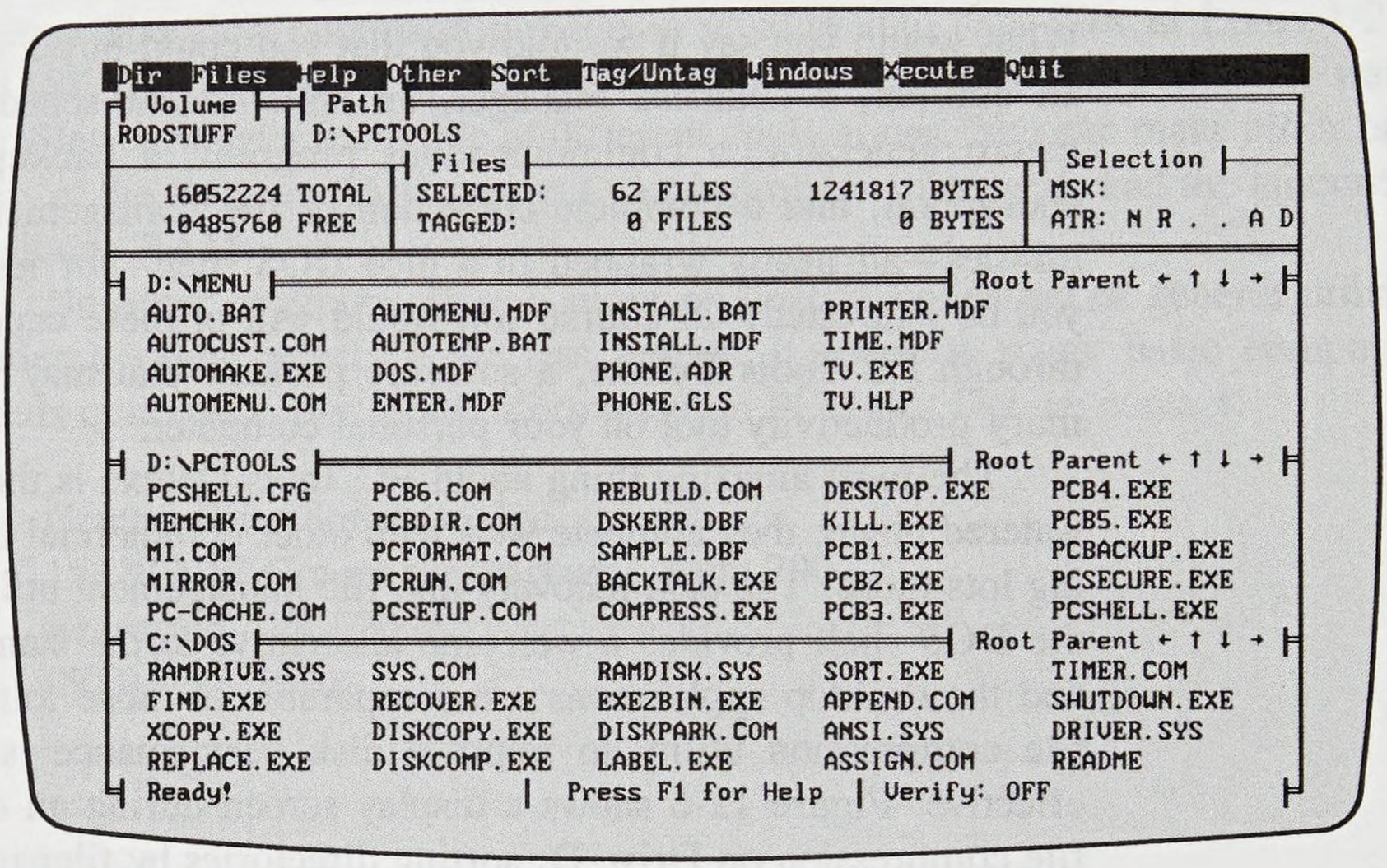
Borland International
4585 Scotts Valley Drive
Scotts Valley, CA 95066
1–800–543–7543

Treeview

Treeview from Magee Enterprises is a powerful and very friendly hard disk **file management utility** that lets you organize hard disk files with just a few keystrokes. It performs all the standard file and disk management functions needed in a file manager, allowing users to accomplish such operations as viewing, copying, moving, and deleting files. Extensive on-line menu-driven help is available at the touch of a key. Figure 12-5 shows a sample Treeview screen with multiple windows.

Figure 12-5

Multiple Treeview Windows



The top part of the screen displays the main menu selections that are available. The bottom three windows display files stored in different directories: The top window displays the files stored in D:\MENU in ascending sequence by filename; the middle window displays files in D:\PCTOOLS in ascending sequence by filename extension; and the last window displays files in C:\DOS in descending sequence by file size, such that the largest files are displayed first.

Treeview can display up to six directories simultaneously in different sequences. You can perform all the Treeview file management operations between any of the six windows. This utility program is a blessing for users with a minimal amount of RAM. Because it is not a TSR program, it only occupies RAM (about 156KB) when it is being used. In addition, Treeview provides full mouse support, automatic recall of DOS commands, 30 user-defined keys, the ability to find files quickly, and more.

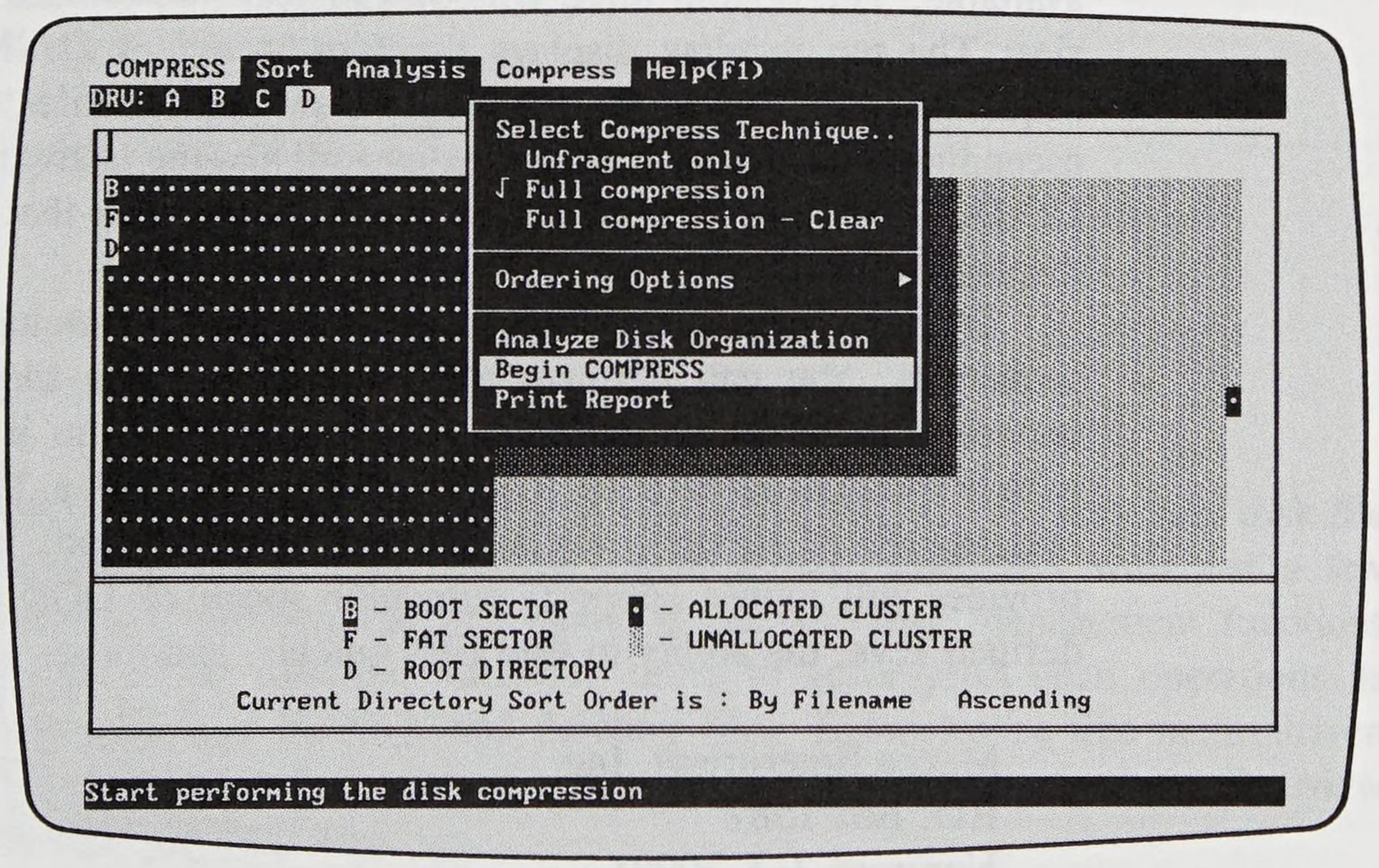
Magee Enterprises, Inc.
P.O. Box 1587
Norcross, GA 30091
1–800–662–4330

PC Tools Deluxe

What would you say if we told you that you could buy a mini word processor, an outliner, a database manager, an appointment scheduler, a calculator, a macro generator, a communications program, a backup program with file encryption, and a complete collection of file management and data recovery utilities—all neatly wrapped in a nice DOS shell—for less than $100? Would you be interested? Of course you would. All of these capabilities are available through **PC Tools Deluxe**, a software package that may well become the primary productivity tool on your personal computer.

The most amazing thing about PC Tools Deluxe is that its features are not watered down; they compete well with other commercial utility programs costing lots more. The data recovery and file management utilities are outstanding, the DOS shell provides a welcome alternative to the standard DOS interface, and the desktop applications are comparable to those found in SideKick. The file compression utility to improve disk performance is amazingly fast and effective. Figure 12-6 shows a display screen during an operation that does a file compression on Drive D, sorting directories by filename.

Figure 12-6
PC Tools File Compression



The PC Tools DOS shell includes a friendly method for viewing selected files or for running a program that you select from a customized pull-down menu. The view command provides a window into data files in Lotus 1-2-3, dBASE III, or ASCII text formats. A clipboard capability lets you easily transfer files between a multitude of different applications. You can make all or only part of the features RAM-resident, depending on your needs and the amount of available RAM.

No other program offers a more productive collection of general utilities under a single interface for less money. It supports many basic computing needs quite well. For more information on PC Tools Deluxe contact:

Central Point Software, Inc.
15220 NW Greenbrier Parkway, Suite 200
Beaverton, OR 97006
(503) 690–8090

Other Commercial Utility Programs

It is not always easy to find a utility program that best meets your specific needs. In addition to these four programs, this chapter lists a small sampling of 25 popular and useful utility programs. There are many outstanding programs other than the ones presented here. Because software prices vary significantly depending on the time and location of purchase, these utility programs are listed in two groups. Figure 12-7 on the next page contains those that typically cost under $50 and includes discount pricing. Figure 12-8 on page 217 includes utility support programs that typically cost between $50 and $100. Software developers are constantly improving the features offered and frequently change their prices. Therefore, this sampling of utility support programs is to be used only as a guide.

Figure 12-7

Sampling of Utility Programs (Under $50)

Program Name	Major Program Features
1 DIR Plus*	DOS shell and disk management system.
Automenu	Powerful menu system (see Chapter 13).
COPY II PC	Backs up protected software and allows it to be executed from hard disk.
Cruise Control*	Keyboard accelerator and screen dimmer.
DJ Ten Utilities	Disk management utilities to improve speed, security, and file management.
DoubleDOS*	Runs two DOS partitions concurrently.
GOfer*	Searches hard disk files for data.
Procomm Plus	Menu-driven communications facilities.
Q-DOS	Menu-driven file management tool.
Software Carousel*	Quickly switches to any of 12 preloaded programs.
Super PC Kwik*	Disk caching utility for significantly improving disk access.
Tornado Notes*	Quick and easy program to enter random notes and retrieve related entries on demand.

* Terminate and Stay Resident program (TSR)

Figure 12-8

Sampling of Utility Programs (from $50 to $100)

Program Name	Major Program Features
Back-it	Comprehensive and friendly disk backup.
DESQview*	Powerful multitasking DOS shell with windows.
FATCAT*	File management program with disk optimizer, file compression, recovery, and security.
Fastback Plus	Very fast and friendly disk backup.
Form Filer	Simplifies the completion of preprinted forms.
Form Tool	Powerful and friendly forms generator.
Instant Recall*	Personal information manager for entering and retrieving items of information.
Mace Utilities	Set of file management utility programs.
Microsoft Windows*	Mouse-directed DOS shell with multitasking and good integration capabilities.
Norton Commander*	DOS shell with ability to enter DOS commands, define menus, and view application files.
SuperKey*	Macro generator with ability to define the keys and lock the keyboard.
Take Charge*	Integrated system with menus, communications, file management, and a desktop manager.
XTreePro	DOS shell and disk management system.

* Terminate and Stay Resident program (TSR)

NONCOMMERCIAL PROGRAMS

Many utility support programs are not commercially developed. They are written by individuals who essentially donate them to the public, allowing for free distribution. However, few things are totally free; there is always a nominal charge levied for copying files and for shipping and handling of disks. Generally, the utility programs from these companies fall into three broad categories: freeware, public-domain programs, and shareware.

Freeware, Public-Domain, and Shareware Programs

Literally thousands of programs are distributed for little more than the cost of the disk and the shipping and handling fees. The number of programs contained on a disk can be 50 or more, depending on the complexity and size of the programs. Programs can be sent on 5 1/4 inch or 3 1/2 inch floppy disks. You can find a wide assortment of powerful word processors, electronic spreadsheets, database managers, communication programs, DOS utilities, and games.

Some people simply like to program and enjoy sharing their best efforts with the rest of us. Others write a program to solve a specific problem and offer to share it with others. Software that is basically free is called **freeware**. It is protected by copyright laws, but the author specifically allows it to be distributed without any fees. However, most freeware programs include a displayed message that asks you to send money to the author if you like the program. Most freeware authors are just trying to cover their time and expenses.

Sometimes these freeware programs are mistakenly called **public-domain** software. But for a program to be truly in the public domain, a programmer must include a notice in the program that says, in essence, "I wrote this program and am placing it in the public domain." Public-domain programs have no rights and they cannot be sold.

Shareware is very much like freeware in that the authors retain all rights to the software under copyright laws, and allow distribution of their systems with few restrictions. If you use shareware, however, you must register the use with the author, who is attempting to make a profit. In most cases, a registration fee that ranges from $5 to $99 and more is required. In return you typically get a bound manual, quick reference cards, free updates, telephone support, and other niceties.

The advantages of the shareware method of distribution are significant. You get a chance to test the software on your computer with your other programs. You can make sure it works to your satisfaction before you buy it with your registration. Because the authors do not have to advertise their products heavily, their prices are generally low. By registering you are supporting future software development of shareware.

Unless software has been placed in the public domain, it is protected by copyright law. Owners of a copyright hold exclusive rights to the reproduction and distribution of their work. Copyright law makes it illegal to duplicate or distribute software or its documentation without the permission of the copyright owner. When you purchase software you may make a backup copy for your use in case the original is destroyed or fails to work. We suggest you use the copy on a regular basis and store the original in a save place as a backup.

Sampling of Utility Support Programs

PC Magazine, a popular computer periodical, has been encouraging the distribution of powerful utility programs for several years. The set of utility disks available free with a subscription to *PC Magazine* is one of the best bargains in utility programs. Users receive many valuable programs with an annual subscription, for about $30. Here is the format and a short description of eight *PC Magazine* utilities that DOS neglected to provide. A word of caution: Due to internal design differences, these utility programs may not work with Version 4.0 of DOS.

Format: `ATTR [±A][±R][±S][±H] [d:][filename[.ext]]`

The DOS ATTRIB command only lets you work with archive and read-only attributes, but **ATTR** lets you display and modify system and hidden file attributes. A plus (+) sign is used to turn on an attribute and a minus sign (–) turns it off. The use of wildcard characters allows multiple files to be modified with a single command.

EXAMPLES OF USAGE:

```
A> ATTR *.*
```
(displays the attributes of all files in Drive A)

```
A> attr +h+r b:test5.doc
```
(converts TEST5.DOC on Drive B to hidden and read-only status)

```
A> ATTR -H B:T*.DOC
```
(all files on Drive B with a DOC extension and a filename that starts with T are no longer hidden)

Format: `BLANKINS`

BLANKINS causes the screen to go completely blank when executed, thereby minimizing screen burnout. Restore the screen by pressing the INS key.

Format: `BROWSE [d:]filename[.ext]`

BROWSE uses cursor movement keys to let you scroll forward and backward to view an ASCII file. The ESC key exits back to DOS.

EXAMPLES OF USAGE:

```
A> BROWSE B:TEST3.DOC
```
(lets you browse through the file named TEST3.DOC on Drive B)

```
A> browse automenu.mdf
```
(lets you browse through AUTOMENU.MDF on Drive A)

Format: `DDIR [d:][path][filename[.ext]]`

DDIR displays a complete directory listing in a two-column format, one screen at a time, sequenced by filename. Wildcard characters are permitted.

EXAMPLES OF USAGE:

```
A> DDIR *.DOC
```
(displays a sorted two-up listing of just the .DOC files on Drive A)

```
A> ddir \word\memo*.*
```
(displays a sorted two-up directory of all the files on \WORD that begin with MEMO)

Format: PRSWAP

PRSWAP converts graphic characters displayed on the screen into ASCII characters that can be printed by nongraphics printers. This TSR program is very helpful in printing graphic displays using the PrtSc key.

Format: SWEEP command [parameters]

In addition to executing a command in the current disk directory, **SWEEP** also executes it in all subdirectories below it. The current directory must be the root to execute a command on all subdirectories on a disk.

EXAMPLES OF USAGE:

```
C:\> A:SWEEP DEL *.BAK
```
(deletes all files with an extension of BAK on Drive C)
```
C:\WORD> a:sweep copy *.doc b:*.bak
```
(copies all DOC files on the WORD directory and its subdirectories on Drive C, backing them up on Drive B with new extensions of BAK)

Format: VTREE [d:]

VTREE displays a graphic tree representation of subdirectories for the specified disk. The output can be redirected to a graphics printer. If you have a nongraphics printer, execute PRSWAP first.

Format: WHERE [filename[.ext]]

WHERE searches all subdirectories on the default drive beginning at the root and lists the full pathname for any nonhidden entry that matches the specified filename. DOS wildcard characters can be used. If you omit a filename, all files are listed.

EXAMPLES OF USAGE:

```
A> WHERE *.BAK
```
(finds all nonhidden files on Drive A with an extension of BAK)

```
B> a:where t*.doc
```
(finds all nonhidden files on Drive B with an extension of DOC and a filename beginning with the letter T)

CONSIDERATIONS, CAUTIONS, AND CONCLUSIONS

With the vast number of low-cost utility support programs readily available, it is important that you be aware of ethical and legal considerations. Don't pirate software from others. It is not only unlawful, but pirated software may contain a computer virus that could adversely affect your system when executed. Instead, go to a legitimate source for your software. Several worthwhile suggestions for obtaining low-cost utility support programs are included below.

Many utility programs can be downloaded from electronic bulletin boards or on-line information services like CompuServe. You can also get them from local users' groups or from large shareware libraries like PC-SIG. Many companies providing this service have toll-free phone numbers for ordering software and for obtaining a free catalog. Several of these companies are listed below:

Gold Hill Software	Gold Hill, OR	1–800–234–6467
Micro Star	Encinitas, CA	1–800–444–1343
Public Brand Software	Indianapolis, IN	1–800–426–DISK
Reasonable Solutions	Medford, OR	1–800–876–3475
Software Excitement	Central Point, OR	1–800–444–5457
The Software Labs	Los Angeles, CA	1–800–359–9998

Some of these utility programs are in the public domain; others are either freeware or shareware. In all cases, you typically get a lot of capabilities for your money. For a flat fee of about $3–$5 per disk, you can get a large number of utility programs. Dozens of programs typically fit on a single disk. Files are often transmitted or copied to a disk in compressed format to save file transmission time and disk space. These files are called **archive files** and almost always have a filename extension of ARC. A special program, such as ARC.EXE and PKPAK.EXE, is normally provided to allow you to decompress archive files.

Over the last two years, there has been increasing concern about programs containing hidden viruses. A **computer virus** is a group of program instructions added by someone perverse enough to enjoy destroying files on other people's hard disks. If you purchase programs from recognized firms or if you download programs from a large information service like CompuServe, you have almost nothing to worry about. All these programs have been thoroughly checked.

Beware of utility programs that you download from an untested electronic bulletin board or that are given to you by a friend. They may contain a computer virus. Once a computer virus is loaded to your system and executed, there is a good chance that it could cause major problems such as destroying all your disk files. There are some good utility programs that can be used to detect and eradicate computer viruses on your system. If you think your system could contain a virus, using one of these programs could save you a great deal of trouble.

Review Questions

12

1. How can the power and capabilities of DOS be extended?
2. What is a DOS shell?
3. What is a macro generator?
4. What is a TSR program?
5. What are the pros and cons of using a TSR program?
6. Describe the windows concept.
7. What are the benefits of using windows?
8. What is a pull-down menu?
9. What is the point and shoot method of operations?

10. What should you do if you accidentally delete a file for which no backup copy exists?
11. What is a desktop manager?
12. What is the purpose of a hot key?
13. What is a file management utility program?
14. Why would you likely run a file compression utility program?
15. What is the difference between freeware and shareware?
16. What are the benefits of the shareware concept?
17. What are the typical benefits of registering for shareware?
18. What are archive files?
19. What is a computer virus?
20. What can be done if you think you have a computer virus?

Chapter

13

MENU-DRIVEN SYSTEMS

INTRODUCTION TO MENUS

☐

USING AUTOMENU

☐ The Elements of Automenu
Format of a Menu Display
Building and Modifying Menus
Technical Considerations

Chapter

13

MENU-DRIVEN SYSTEMS

INTRODUCTION TO MENUS

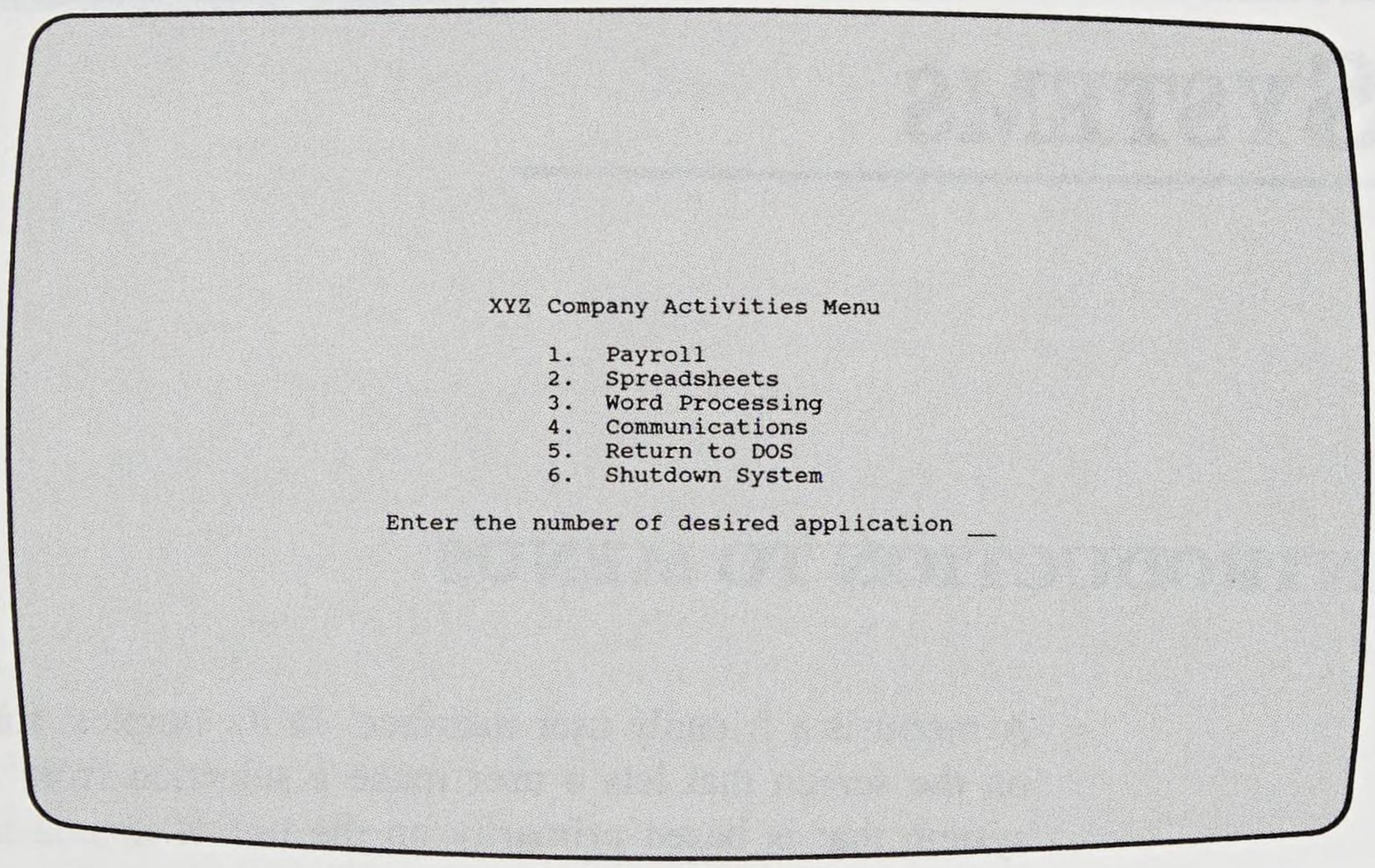
A **menu** is a friendly user interface. In its simplest form, a menu is a display on the screen that lets a user make a selection from a list of alternatives. A system that is based primarily on the use of menus is called a **menu-driven system**. With menu-driven systems, frequently used commands are incorporated into a menu. New or inexperienced users can easily select the activities they want from the menu. No longer do they have to remember complex DOS commands or the correct subdirectory for executing a program.

A sophisticated menu-driven system has a main menu that controls an organized set of related menus. This organized structure is analogous to the directory structure on a hard disk. A menu program is often utilized to assist the user in developing a menu-driven system. It offers an easy way to predefine the activities displayed on menus and navigate through a set of menus to make a selection. In addition, a menu program initiates the execution of a selected activity. When the selected activity is finished, control returns to the menu program and the main menu is displayed again.

Although menu-driven systems may appear slow or restrictive, they represent an extremely easy and friendly way to direct user activities on a computer. Users who don't understand DOS can be highly productive using a menu-driven system, limited only by the capabilities built into the menus. In addition, menus offer some protection by not allowing novice users access to restricted parts of the system. Figure 13-1 illustrates a relatively simple menu displayed on a screen.

Figure 13-1

Sample Menu Display



In most cases, menus are initially set up by an experienced DOS user. These menus can then be used by less experienced DOS users to improve their productivity. Novice users need only enter a single keystroke reflecting an activity they wish to accomplish, such as word processing or spreadsheets. Activities can be DOS commands, batch files or application programs. A menu system is especially helpful when the set of activities is fairly constant and the menu does not require constant changing to meet the needs.

There are many commercially developed menu systems on the market. However, you do not have to purchase one; the experienced DOS user can develop his or her own menu-driven computer system utilizing batch files. The first two exercises at the end of this chapter show you how to create and modify this type of manual menu system. Manually developed menu systems, however, are primitive compared with commercial systems. In addition, manual systems are time-consuming to create and modify. This chapter discusses the capabilities of a sophisticated, yet reasonably priced commercial menu system called **Automenu®**, the Software Management System™. We chose to introduce you to Automenu for two reasons: (1) It is the de facto standard for menu systems, and (2) its shareware approach lets you evaluate it free of charge.

Before we specifically discuss Automenu, however, it is important to understand the concepts and terminology common to commercial menu systems. In addition to displaying menu selections, commercial menu systems display menu titles, explanations of selections, and the system date and time. They also display the status of special keys such as Caps Lock and Num Lock. With commercial menu systems users can choose from several methods to select a desired activity. They may use a function key, a number key, or the up/down arrow keys to make a selection. A wraparound technique allows users to jump from the first selection on the list to the last item by pressing the up arrow key. Selected activities can be highlighted on the screen for visual verification before being executed. Use your choice of function keys, number and arrow keys, space bar, mouse, or voice controller to highlight and select menu options easily and quickly. Although the specific method for executing a selected menu activity varies, the following four steps are usually required:

1. Change to the appropriate directory using the DOS CD command.
2. Execute the appropriate DOS commands, batch files, or application programs stored on that directory.
3. Use the CD command to return to the original directory of menu programs.
4. Display the menu again, allowing the user to select another activity.

With a manual menu system you must create a unique set of commands for executing these four steps for each menu selection. Each set of commands is saved in a separate batch file. The appropriate batch file can then be executed when an activity is selected. With commercial systems, commands for all menu selections are stored together in a common **Menu Definition File (MDF)**. The MDF is a critical element; it defines all the capabilities of a menu-driven system. A menu building program is provided to assist users in building and maintaining the MDF. A sophisticated menu system like Automenu can have unlimited MDFs, allowing for an unlimited number of menu selections.

Automenu is an excellent menu program for DOS-based microcomputers. Developed by Marshall W. Magee in 1983, it is now widely distributed through user groups, electronic bulletin boards, and retail outlets. Automenu has been so successful that Magee Enterprises is offering other outstanding products like Treeview. Treeview is a powerful and easy-to-use file management utility program we reviewed in the previous chapter.

Automenu has all the features typically found in other commercial menu systems, plus a significant number of additional ones. Figure 13-2 highlights some of the outstanding features of Automenu.

Figure 13-2
Features of Automenu

- Easy installation and operation on a single user system or on a local area network (LAN).
- A comprehensive program for building customized menu selections using a powerful command language.
- Ability to prompt the user for input and act accordingly via replaceable parameters.
- Unlimited number of menu selections by linking multiple submenus.
- Menu selections can be entered in a variety of ways including the use of function keys, number keys, arrow keys, space bar, mouse, and a voice controller.
- Easy-to-use password protection for execution of restricted activities on a menu, as well as protection for the entire menu.
- Preset timed execution of menu selections allows them to be run at any predetermined time, such as at night or on the weekend.
- An automatic and customized screen black out facility to prevent the screen from burning out.
- Non-RAM resident program to minimize the amount of memory required to make a selection.
- Non-copy protected program with the mouse and color monitor support.
- Complete and easy to follow user's manual, with on-line help capability.

Automenu is a shareware software product that encourages you to distribute copies to friends, but only for a 30-day evaluation period. After evaluation, if they want to continue using Automenu they must become registered users by purchasing it from Magee Enterprises. The current purchase price is less than $70. That is a reasonable price to pay for such a helpful program. Registered users benefit from receiving a comprehensive user's manual, enhanced technical support, and notification of new products. Here is how to contact the company.

Mailing address:
Magee Enterprises, Inc.
P.O. Box 1587
Norcross, GA 30091

Orders (USA): 1–800–662–4330
Main office: (404) 446–6611
FAX number: (404) 368–0719
CompuServe number: 76004,1541
or 70167,2200

USING AUTOMENU

The Elements of Automenu

The Automenu package consists of a dozen or so programs and files. Many of these assist you in getting started, including installation programs, customization programs, documentation files, and sample menu files. The critical elements of Automenu include a main menu definition file, two menu programs, and two required batch files. Figure 13-3 on the next page describes the major components of Automenu.

Figure 13-3

Primary Automenu Programs and Files

AUTOMENU.MDF--This skeleton Menu Definition File (MDF) is very helpful in getting a new user started. It is one of several MDFs supplied with the package. Using the menu building facility of Automenu, an MDF can be easily modified to meet individual user requirements. Multiple MDFs may be created and used with Automenu.

AUTOMAKE.EXE--This program is used to build and modify MDF files. It is very easy to use and is normally executed via a menu selection in the skeleton MDF file supplied with Automenu. Some working knowledge of DOS is required to define the specific commands required to execute each activity.

AUTOMENU.COM--This is the main operating program used to display menus, capture user selections, and then initiate execution of predefined commands required to complete the menu selection. Normally this program creates a batch file called AUTOTEMP.BAT.

AUTOTEMP.BAT--This temporary batch file contains the set of commands that relate to a selected menu item. This batch file is automatically executed by Automenu (via AUTO.BAT) after AUTOMENU.COM creates it.

AUTO.BAT--This simple batch file is used to run Automenu. Specifically, it executes AUTOMENU.COM to select an activity from a menu, and then AUTOTEMP.BAT to execute that selection.

Once the MDF files are built, the process of running Automenu is straight forward. AUTO.BAT executes AUTOMENU.COM to allow you to enter a menu selection. The commands related to a particular selection are stored in AUTOTEMP.BAT, which is then executed as the next statement in AUTO.BAT. After selecting an activity to run Lotus 1-2-3, the contents of AUTOTEMP.BAT might be as follows:

```
CD\SS
LOTUS
CD\AUTOMENU
AUTO
```

Whenever the last command in the AUTOTEMP.BAT file is AUTO, AUTOMENU.COM is executed. This allows the cycle to continue by displaying the menu selections again. Using this example as a guide, you can easily create your own set of commands to execute a wide variety of applications.

Format of a Menu Display

As a new user of Automenu, when you first enter the command AUTO, your monitor should display the main menu, defined by AUTOMENU.MDF. It should look similar to the one in Figure 13-4 on the next page, with the featured areas numbered from 1 to 7. The numbered areas are explained in Figure 13-5 on page 235.

Figure 13-4

Main Menu Display

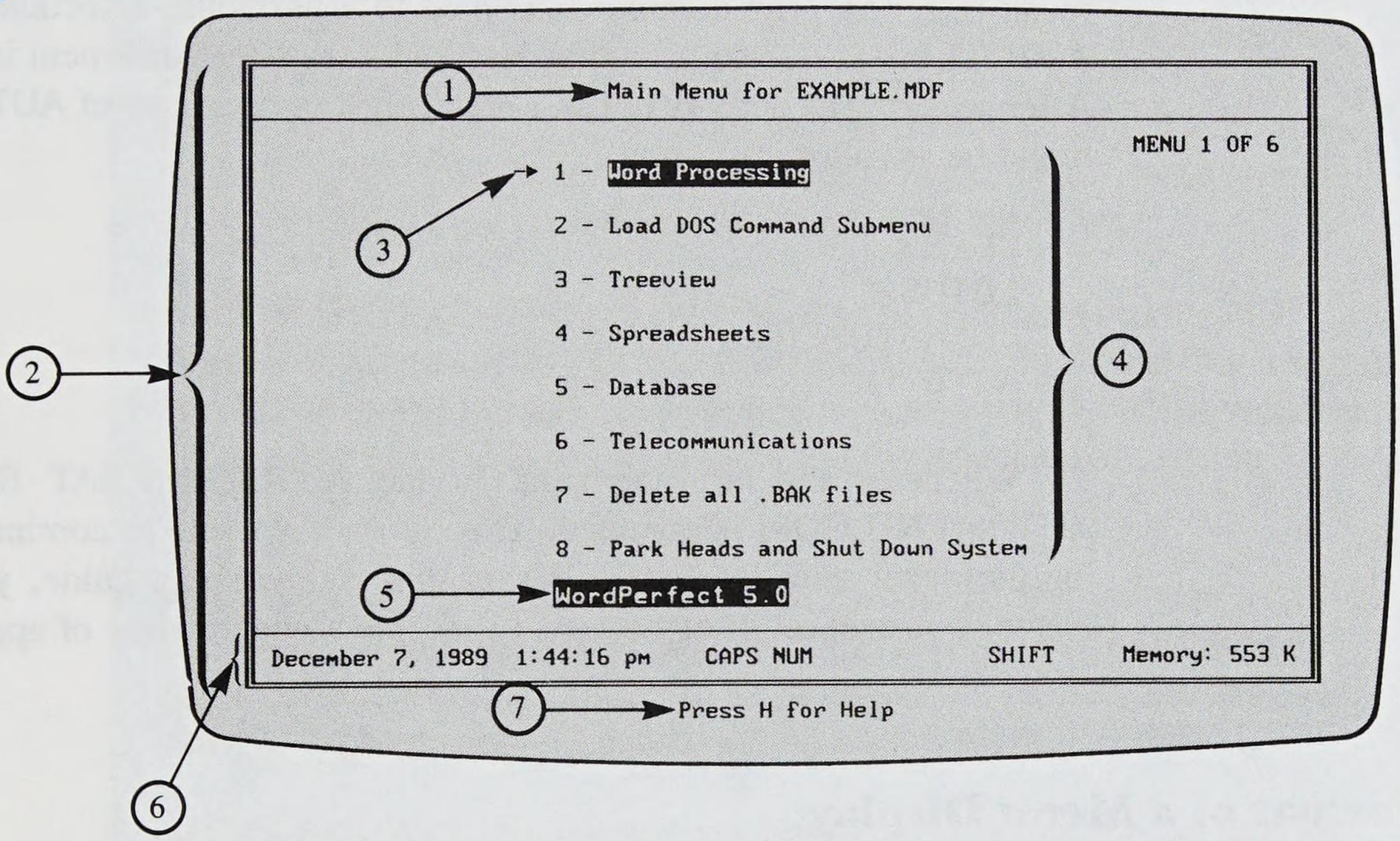


Figure 13-5

Explanation of Menu Features

1. Title--Each menu page has its own title which is automatically centered on the top line of the menu page.

2. Menu Page--Each MDF can have up to 8 menu pages with each one being specified with a new title (you can use the right/left arrow keys to display next or previous menu page).

3. Menu Selector--The selection pointer can be moved using the space bar or the arrow keys to highlight menu items. Highlighted selections are executed by pressing the Enter key. The function keys or the number keys reduce this process to a single keystroke. They can be used to highlight and execute a menu selection without pressing the Enter key.

4. Menu Selections--Predefined activities are displayed on a menu and specified in an MDF file.

5. Explanation--Text used to clarify a highlighted menu selection can be displayed in reverse video at the bottom of the menu.

6. Status Line--This portion of the menu page displays the date, time, status of special keys in use, and available memory.

7. On-line Help--Press H to access to the Automenu help facility.

Building and Modifying Menus

AUTOMAKE is the menu building facility for Automenu that is used to create and modify a menu definition file (MDF). This menu editor is easy to use and features a wealth of capabilities to let you build sophisticated MDFs. It even lets you password-protect an MDF so that unauthorized users cannot view or modify it. The AUTOMAKE facility provides for three processing options, shown in Figure 13-6.

Figure 13-6

Screen Used to Access the AUTOMAKE Program

```
AUTOMAKE  4.5 - The Automenu Menu-Building Facility

1 - Modify an Old or Create a New Menu Definition File

2 - Print the Menu Definition File to Printer

3 - Leave the AUTOMAKE Facility

      Please Make Your Selection (1-3) => __
```

AUTOMAKE rejects any selection that is not 1, 2, or 3. The last two options of AUTOMAKE are self-explanatory, but the first option may need some clarification. This option lets you edit an MDF by first asking you to identify the MDF you wish to edit. If you do not specify any name, then AUTOMAKE defaults to AUTOMENU.MDF. If you specify an existing file and you know the password (assuming one was previously specified), you will be allowed to view and/or modify it. Otherwise, AUTOMAKE lets you create a new MDF using a skeleton MDF to get you started. Figure 13-7 illustrates a sample MDF that can be used to explain some of the menu building capabilities of Automenu.

Figure 13-7

Sample MDF File Displayed by AUTOMAKE

```
 Function Description    AUTOMAKE - The Menu Building Facility for AUTOMENU
                                    Now Editing Row #:  1

 Comment      .  EXAMPLE AUTOMENU MENU DEFINITION FILE
 Comment      .  By Rod B. Southworth 11/23/89 ..............................
 Title        %  Main Menu for EXAMPLE.MDF
 Selection    *  Word Processing
 Password     ^  broncos
 Batch-NRes   +  cd\word
 Batch-NRes   +  word
 Batch-NRes   +  cd\automenu
 Batch-NRes   +  auto
 Selection    *  Load DOS Command Submenu
 Descript     ?  Transfers to the DOS MDF
 Load a MDF   @  DOS.MDF
 Selection    *  Treeview

                          MDF Functions
 . = Comment          ^ = Password         + = Batch-NRes       > = Output Msg
 % = Title            [ = Time Start       ! = Batch-Res        < = Input
 * = Selection        @ = Load a MDF       - = Direct
 ? = Description                           = = Direct/P         # = End of MDF

 F1 = TOGGLE HELP ON/OFF   F3 = INSERT LINE   F4 = DELETE LINE   ESC = QUIT
```

The bottom portion of the screen identifies the command symbols that can be used to edit an MDF. Most of these functions are discussed below. The command symbols not discussed here are advanced functions. Registered owners with a user's manual can learn these on their own.

The top portion of the screen contains entries in the MDF, up to 13 entries at a time. The Function Description area (on the left edge) is a protected area of the screen containing a brief explanation of each command entry. Command symbols you enter on each line of the MDF determine the contents of the protected area. Each item in the MDF example shown in Figure 13-7 is discussed below.

The first two entries are comment lines, denoted by a period (.) as the command symbol. Comments can contain anything. They are much like REM statements in a batch file and are ignored by Automenu during execution.

The next entry (%) identifies a menu title that must be defined for each individual menu page. Each MDF can have a maximum of eight pages. The title for each page is centered automatically.

Following the title is the first menu selection. Menu selections are identified with an asterisk (*). There can be up to eight selections per page.

Between each set of asterisks are the entries that are used to execute the selected activity. The first entry shown here is a caret (^), the symbol used by Automenu for entering a password. In this example the password is broncos. During execution of this sample menu a user may select this activity. Users are not allowed to continue past this point unless they supply the password.

The next four entries are the specific commands used by Automenu to create AUTOTEMP.BAT. There is no limit to the number of entries that can be defined for each menu selection. Entries can be DOS commands, batch files, or application programs. These four entries demonstrate the steps we discussed earlier, those used to execute a selected activity. Remember, AUTOTEMP.BAT is part of AUTO.BAT. It is automatically executed after being created by Automenu. AUTOTEMP.BAT contains a set of commands extracted from the MDF for a given menu selection.

The next menu selection (identified by an asterisk, *) involves two additional command symbols. The question mark (?) causes the specified text to be displayed as an explanation or description for the selection that is highlighted. The text is automatically centered and printed on a line above the status portion at the bottom of the menu.

The at sign (@) is used by Automenu to load and execute another MDF. It must be a stand-alone entry; it cannot be used together with a symbol like the plus sign (+) that executes a command, batch file, or application program.

The last three MDF functions shown at the bottom right side of the screen in Figure 13-7 are also explained here. The output command symbol (>) is used to display a message to the user on the monitor, typically to inform the user what is happening or to prompt him or her for additional parameters. Each output message is limited to 80 characters, but you can enter as many message lines as you wish.

When entries in an MDF require additional information from the user, the input command symbol (<) can be used. A highlighted row of dots is automatically included when the entry executes to identify the place to enter a response. These two output and input commands can best be explained with an example. Suppose you wanted the ability to selectively execute your word processing program from a given subdirectory. You could add the following set of entries to your MDF.

```
*Word Processing
?Executes WORD from a given subdirectory
>Enter the subdirectory on WP containing the documents you
>want to process on the DOTTED line below and press Enter.
<
+CD\WP\%1
+\WP\WORD
+CD\AUTOMENU
+AUTO
#End of Menu Definition File
```

This set of entries lets the user input a replaceable parameter during execution. The entries used to create AUTOTEMP.BAT reflect the subdirectory entered by the user. When the word processing program (WORD) is finished, AUTO is executed and control is passed back to Automenu to display the menu again. The last entry shown in this example is required to inform Automenu that it has reached the end of the MDF. It must be included as the last entry in any MDF.

Technical Considerations

The primary purpose of this chapter is to provide an overview of menu-driven systems and to introduce you to Automenu. The chapter does not cover the many aspects of Automenu you need to know to be truly effective using it. Here is a summary of some of the technical considerations:

- All Automenu functions are menu-driven, making it easy to install and use.
- Automenu requires only 32KB of memory because it is written entirely in assembler. It is compatible with DOS, Microsoft Windows and several network operating systems.
- You can use INSTALL.BAT to simplify the process of installing Automenu on a hard disk system.
- Most entries in an MDF can be up to 127 characters long, but cannot be blank.
- When modifying entries in an MDF, you can use the Ins and Del keys to edit them just as you use the DOS editing keys to modify batch file entries or the last DOS command from the buffer.

- You can never predefine all the activities you want to accomplish in a menu. Therefore, you should make sure that each MDF has an entry that returns you to the DOS prompt. Otherwise, you can be stuck in the menu with no way to exit.
- If you password-protect an MDF, be sure to remember the password. There is no way to view or modify a protected MDF unless you know the password.
- When you execute another MDF (via the @ command), be sure to provide a means of returning to the original MDF. This is easily done by a selection in the second MDF that contains @AUTOMENU.MDF.
- AUTOCUST is a program that lets you customize the execution of Automenu. You can modify such things as the color, the time delay to automatically black out the screen, and the way items display on the status line at the bottom of the screen.

You will need to refer to the Automenu user's manual for additional information on Automenu. The command symbols not discussed in this chapter provide for much more capability and can be used to significantly improve the efficiency of operations.

Review Questions

13

1. What is a menu-driven system?
2. What is a menu program (menu building facility)?
3. Why use a commercial menu system?
4. Why is a working knowledge of batch files required to build menus?
5. What is the purpose of a menu definition file?
6. What types of activities can be executed from a menu selection?
7. What four operations are typically used to execute something from a menu and then display the menu again?

8. What is the purpose of the AUTOTEMP.BAT file in Automenu?
9. What is the purpose of AUTOMENU.COM in Automenu?
10. How does Automenu provide for an unlimited number of menu selections?
11. How can you restrict an unauthorized user from executing a menu selection?
12. How can you protect an Automenu MDF from being viewed or modified?
13. What is the purpose of a screen blackout facility?

Chapter 13 Lab Exercises

1. In order to develop a manual menu system you must create two types of files. The first type of file is a text file containing the menu display. The second type of file is a set of batch files, one for each activity on the menu and one to execute the menu. To simplify the process, both types of files should be stored on the same subdirectory. Complete the following steps to create a manual menu system:

   - Make a subdirectory on Drive A called \MENU and make it the current directory. Then use EDLIN to create the following text file called **MENU.TXT**:

     SAMPLE MENU
     1. Directory of Drive A
     2. CHKDSK of Drive A
     3. Return to Root Directory

     Enter Number of Desired Selection

   - Create a batch file (**GO.BAT**) to execute the menu as follows:

```
ECHO OFF
REM  GO.BAT
CLS
TYPE MENU.TXT
```

- Now create a separate batch file for each of the three activities in the sample menu. The first batch file (**1.BAT**) should look like this:

```
ECHO OFF
REM  1.BAT
DIR  A:/P
PAUSE
CLS
TYPE MENU.TXT
```

- The last batch file (**3.BAT**) should not display the menu again. When all three batch files are created, test each one with the menu. Enter GO to execute the menu. When you are satisfied it works correctly, make sure the current directory is \MENU before continuing to the second exercise.

2. Modifying a manual menu system can sometimes be cumbersome. Modify the menu created in the first exercise to have menu selection 3 create a backup copy of the menu system (the TXT and BAT menu files) on Drive B. Label this menu selection "Menu System Backup to Drive B:". The selection previously numbered 3 now becomes the fourth activity, requiring you to make further modifications. Be sure to test the revised menu completely. When you are finished and the menu files are backed up on Drive B, delete all the menu files you created above from Drive A.

3. Format a blank disk using your name as the volume label. Then copy all the files from the Automenu disk provided by your instructor to this disk. Label the new disk "Automenu Working Copy" and use it for the remaining exercises.

4. With the Automenu disk in Drive A, enter `AUTO ENTER.MDF`. This executes a sample MDF that requires a password to continue. The only way out of this operation is to enter the correct password. Enter `IBM` for the password when you are prompted. It then executes another MDF (AUTOMENU.MDF) that can be used as a starting point to complete the next exercise.

5. Use the working copy of Automenu to create a menu system duplicating the activities in the first exercise. Use the menu building facility within Automenu to create a new menu definition file called **EXERCISE.MDF**.

6. Bonus exercise:

   Use Automenu to change EXERCISE.MDF. First, make the modifications specified in the second exercise. Then add a password to selection 1 and a descriptive comment to selection 3. Make up a password and explanation to use in the MDF. Finally, add another menu selection (as selection 1) to execute another MDF called SAMPLE.MDF. Test all these modifications carefully to make sure they work correctly. You have probably thought of other modifications you would like to make as well, so go ahead!

This concludes Chapter 13 Lab Exercises. When you are finished, remove your disks. We hope you enjoy working with Automenu!

A

Summary of DOS Commands

The term filespec in the command formats below refers to the complete filename `[d:][path]filename[.ext]`. Chapter references are shown in parentheses for each command.

DOS Command	*Brief Description of Command*
APPEND	`[d:][path][;[d:][path]...]` Sets a search path for data files. (8)
ASSIGN	`d=d [d=d]` Assigns a disk drive letter to another drive. (8)
ATTRIB	`[±R][±A] filespec[/S]` Sets/displays attributes of a file. (4)
BACKUP	`d:[path][filespec] d:[/S][/M]` Backs up files from disk. (7)
BREAK	`[ON|OFF]` Sets Ctrl-Break check. (3)
CD (CHDIR)	`[d:][path]` Changes directories. (7)
CHKDSK	`[d:][filespec] [/F][/V]` Checks status of a disk, files, and RAM. (4)

DOS Command	*Brief Description of Command*
CLS	Clears the screen. (3)
COMMAND	Supplies internal DOS commands. (3)
COMP	`[filespec] [filespec]` Compares two files for differences. (4)
COPY	filespec `[d:][filespec] [/V]` Copies specified files. (3)
DATE	`[mm-dd-yy]` Displays and sets the system date. (3)
DEBUG	`[filespec]` Machine-level utility program for DOS. (10)
DEL	`filespec` Deletes (erases) specified files. (3)
DIR	`[d:][filespec] [/P][/W]` Displays directory entries. (3)
DISKCOMP	`[d:[d:]]` Compares two disks. (4)
DISKCOPY	`[d:[d:]]` Makes an exact copy of a disk. (4)
EDLIN	`filespec` Executes the DOS line editor. (6)
ERASE	`filespec` Identical to the DEL command. (3)
FASTOPEN	`d:[=size] [d:[=size]...]` Improves hard disk performance. (10)
FDISK	Partitions a hard disk for DOS. (7)
FIND	`"text" filespec` Searches for a given string of text. (4)
FORMAT	`[d:] [/S][/V]` Formats a disk to receive DOS files. (4)
LABEL	`[d:] [text]` Labels a disk with volume label. (4)
MD (MKDIR)	`[d:]path` Makes a new directory. (7)

DOS Command	*Brief Description of Command*
MODE	Modifies system parameters. (9)
MORE	Displays output one screen at a time. (5)
PATH	`[d:][path][;path...]` Sets a search path for executable files. (7)
PRINT	`[/B:buffer] [/S:time] [/D:device] filespec` Queues files for printing concurrently. (10)
PROMPT	`[text]` Assigns the system prompt. (3)
RECOVER	`d:` –or– `filespec` Recovers a bad disk's FAT or a damaged file. (4)
RENAME	`filespec filename[.ext]` Renames a file. (3)
REPLACE	`filespec [d:][path] [/A][/P]` Adds or replaces files to another directory. (10)
RESTORE	`d: d:filespec [/S]` Restores previously backed up files. (7)
RD (RMDIR)	`[d:]path` Removes a directory. (7)
SORT	Sorts data in ascending or descending order. (5)
SUBST	`d: [d:]path [/D]` Substitutes a disk drive letter for a path. (10)
SYS	`[d:]` Transfers DOS hidden files to a disk. (4)
TIME	`[hh:mm[:ss]]` Displays and sets the system time. (3)
TREE	`[d:] [/F]` Graphically displays directories and filenames. (7)
TYPE	`filespec` Displays the contents of an ASCII file. (3)
VER	Displays the DOS version number. (3)
VERIFY	`[ON|OFF]` Verifies all writes to a disk for accuracy. (3)
VOL	`[d:]` Displays the disk volume label. (3)
XCOPY	`filespec [filespec] [/M][/P][/S]` Expanded version of the COPY command. (8)

DOS CONFIG.SYS Commands (Chapter 9)

Command	*Brief Description of Command*
BREAK = ON	Checks for a Ctrl-Break during disk I/O.
BUFFERS = nn	Increases default buffer size.
FILES = nn	Increases default number of files open.
DEVICE = ANSI.SYS	Activates driver for monitor and keyboard.
DEVICE = VDISK.SYS	Activates RAM disk (RAMDRIVE.SYS in MS-DOS).

DOS EDLIN Commands (Chapter 6)

Command	*Brief Description of Command*
#	Edit a given line.
C	Copy a range of lines prior to a line.
D	Delete a range of lines.
E	End editing and save the file.
I	Insert lines prior to a line.
L	List a block of lines.
M	Move a range of lines prior to a line.
Q	Quit editing without saving the file.
R	Replace one text string with another in a range of lines.
S	Search for a text string in a range of lines.

DOS Batch File Commands (Chapter 5, Chapter 11)

Command	*Brief Description of Command*
CALL	Executes another batch file.
ECHO	Sets the echo feature on/off.
FOR	Command for repetitive looping.
GOTO	Command for branching.
IF	Command for conditional branching.
PAUSE	Pauses for input in a batch file.
REM	Provides for remarks in a batch file.

B

DOS 4.0 AND OS/2 HIGHLIGHTS

Both DOS 4.0 and OS/2 give the user a lot more power and capabilities than DOS 3.3. With its ability to perform multitasking, OS/2 is a very powerful operating system, requiring a substantial amount of RAM to execute. If you aren't quite ready for OS/2, DOS 4.0 is a potential alternative. Because DOS 4.0 can also be memory intensive, you should be willing to pay the RAM price for upgrading. Here is a summary of DOS 4.0 features.

DOS 4.0 HIGHLIGHTS

☐

The most obvious change from DOS 3.3 is the inclusion of a user shell which makes DOS easier to learn and use. The shell, which requires 256K bytes of RAM, contains a command selector, a menu facility, a file management program, and help screens. The shell's pull-down menus, pop-up windows, and point and shoot method of operation can be controlled by a mouse or by the keyboard. Other significant additions found in DOS 4.0 follow.

- An install program that assists you with installation of DOS 4.0 by simplifying the modification of configuration defaults.
- A modified FDISK program that lets users create partitions greater than 32MB, up to a maximum of 1 gigabyte.
- Support for applications that use the Lotus/Intel/Microsoft Expanded Memory Specification (LIM EMS) 4.0 and an option to use expanded memory with BUFFERS, FASTOPEN, and VDISK.

This upgrade of DOS (from 3.3 to 4.0), however, won't have nearly the effect that the major upgrades in the past had. To use it effectively, you should have either extended or expanded memory. On 80286-based machines, extended memory is between 1MB and 16MB of additional RAM. This memory is largely untapped by DOS, except for disk caching or print spooling purposes.

Expanded memory goes beyond extended memory and requires a special driver to work with DOS. It swaps data in the extended memory area with the first megabyte of memory, fooling DOS into addressing more memory than it could otherwise handle. A growing number of application programs support LIM EMS 4.0, allowing them to use data stored in expanded memory.

There are other potential problems with upgrading to DOS 4.0. Some TSR programs may not work due to changes in the internal design. These programs must be upgraded to work with DOS 4.0. In addition, if you have trouble creating batch files with replaceable parameters, you won't find the menu facility very easy to use.

If you have less than one megabyte of RAM you should seriously consider expanding the capabilities of DOS 3.3 with a variety of utility support programs. There are several excellent DOS shells like DESQview or The Norton Commander, as well as some excellent file managers like Treeview or Q-DOS that can give you comparable capabilities.

In summary, while DOS 4.0 contains some improvements in usability and functionality, it requires extra memory and has some drawbacks. If you want effective multitasking, virtual memory, and high performance, you should consider upgrading to a more powerful operating system like OS/2.

OS/2 HIGHLIGHTS

OS/2 is a complex multitasking operating system designed for use in a sophisticated multi-user environment. It has enough significant new hardware requirements, like a 80386 processor, 4–5MB of RAM, and high-resolution graphics, to demand a vastly more expensive computer system. The most significant features of OS/2 are as follows:

- It breaks the 640KB RAM barrier that DOS imposes, giving users access to huge chunks of contiguous memory.
- It allows multitasking, which means that you can switch between several applications, each in its own window.
- OS/2 lets programs efficiently pipe data back and forth between windows, invisibly and in the background.

As nice as OS/2 is, it appears that it will not significantly replace DOS for several more years. The upgrade costs for such a user-friendly and powerful computing environment are excessive. One company estimated that it would cost well over $1000 per machine to upgrade their existing PC ATs to run OS/2.

Many users are choosing DOS rather than OS/2, thanks in part to powerful utility programs that run multitasking with a graphic interface. These programs include Microsoft Windows, DESQview, and Concurrent DOS. Although OS/2 is slow to gain acceptance, new applications for it will continue to develop as microcomputers get bigger. For the time being, however, it appears that DOS will continue to be the dominant operating system for several more years.

C

Common DOS Error Messages

Abort, Ignore, Retry, Fail?

A disk error has occurred. Enter A to end the process that requested the disk read or write operation. Enter R to retry the operation or F to end the operation, but continue the process that requested the disk read or write. Do not enter I because it can cause a loss of data.

Abort, Retry, Fail?

A floppy disk device error has occurred. Normally you should enter R to try again after correcting the problem, like closing the latch on the drive or reinserting the disk. If the second attempt fails, enter A to terminate the process.

Access denied

You tried to access a write-protected or a read-only file. Modify the file attributes to allow access.

Bad command or filename

(1) The command entered is not a valid DOS command; (2) DOS cannot find the command; or (3) DOS cannot find the specified file. Often the correct path is not provided for DOS to locate the command or file.

Bad or missing Command Interpreter

COMMAND.COM is not on the root directory of the boot disk.

Disk error reading (or writing) drive x:

The disk in the drive specified has a bad sector. If entering R (retry) does not resolve the problem, enter A to end the process.

Disk unsuitable for system disk

FORMAT detected a bad track on the disk where system files reside. The disk can only be formatted as a non-system disk.

File allocation table bad

The disk may be defective. Run CHKDSK /F to fix the problem. If this message is displayed again, you must reformat the disk.

File cannot be copied onto itself

The source file name is the same as the target file name. Make sure the target file name (drive, path, etc.) is complete.

File creation error

You tried to add a filename that already exists in the directory, or there was not enough space for the file.

Format failure

A fatal disk error prevented DOS from formatting the disk.

Incorrect DOS version

You attempted to run a DOS command from a different version of DOS than that used to boot the system. Reenter the DOS command from the DOS version used to boot the system.

Insufficient disk space

The disk does not contain enough room to perform the operation.

Invalid number of parameters

The command line entered did not contain the correct number of parameters for the command you invoked.

Invalid parameter(s)

One of the command options is wrong or does not exist.

Invalid path, not directory, or directory not empty

You could not remove a directory for one of the specified reasons. Correct the problem and reenter command.

Non-system disk or disk error

The disk is not a bootable DOS disk.

Not ready error reading drive x: Abort, Retry, Fail?

DOS cannot read or write to the specified drive. Make sure that the drive latch is closed and that it contains a disk.

Program too big to fit in memory

DOS cannot load an executable program because of insufficient free space. Remove any unnecessary buffers and TSR programs.

Target diskette may be unusable

The target disk in a DISKCOPY operation has an unrecognizable format or it is defective.

Write-protect error writing drive x:

You tried to write data to a disk with a write-protect tab covering the notch, or it does not have a write-protect notch.

D

ASCII Codes

ASCII Value	Hexidecimal Form	Character	Control Character
000	00	(null)	NUL
001	01	☺	SOH
002	02	●	STX
003	03	♥	ETX
004	04	♦	EOT
005	05	♣	ENQ
006	06	♠	ACK
007	07	(beep)	BEL
008	08	◘	BS
009	09	(tab)	HT
010	0A	(line feed)	LF
011	0B	(home)	VT
012	0C	(form feed)	FF
013	0D	(carriage return)	CR
014	0E	♫	SO
015	0F	☼	SI
016	10	►	DLE
017	11	◄	DC1
018	12	↕	DC2
019	13	!!	DC3
020	14	¶	DC4
021	15	§	NAK
022	16	▬	SYN
023	17	↨	ETB
024	18	↑	CAN

ASCII Value	***Hexidecimal Form***	***Character***	***Control Character***
025	19	↓	EM
026	1A	→	SUB
027	1B	←	ESC
028	1C	(cursor right)	FS
029	1D	(cursor left)	GS
030	1E	(cursor up)	RS
031	1F	(cursor down)	US
032	20	(space)	SP
033	21	!	
034	22	"	
035	23	#	
036	24	$	
037	25	%	
038	26	&	
039	27	'	
040	28	(	
041	29	)	
042	2A	*	
043	2B	+	
044	2C	,	
045	2D	-	
046	2E	.	
047	2F	/	
048	30	0	
049	31	1	
050	32	2	
051	33	3	
052	34	4	
053	35	5	
054	36	6	
055	37	7	
056	38	8	
057	39	9	
058	3A	:	
059	3B	;	
060	3C	<	
061	3D	=	
062	3E	>	
063	3F	?	
064	40	@	
065	41	A	
066	42	B	
067	43	C	
068	44	D	
069	45	E	
070	46	F	

ASCII Value	***Hexidecimal Form***	***Character***	***Control Character***
071	47	G	
072	48	H	
073	49	I	
074	4A	J	
075	4B	K	
076	4C	L	
077	4D	M	
078	4E	N	
079	4F	O	
080	50	P	
081	51	Q	
082	52	R	
083	53	S	
084	54	T	
085	55	U	
086	56	V	
087	57	W	
088	58	X	
089	59	Y	
090	5A	Z	
091	5B	[	
092	5C	\	
093	5D	]	
094	5E	^	
095	5F	_	
096	60	`	
097	61	a	
098	62	b	
099	63	c	
100	64	d	
101	65	e	
102	66	f	
103	67	g	
104	68	h	
105	69	i	
106	6A	j	
107	6B	k	
108	6C	l	
109	6D	m	
110	6E	n	
111	6F	o	
112	70	p	
113	71	q	
114	72	r	
115	73	s	
116	74	t	
117	75	u	

ASCII Value	***Hexidecimal Form***	***Character***	***Control Character***
118	76	v	
119	77	w	
120	78	x	
121	79	y	
122	7A	z	
123	7B	{	
124	7C	|	
125	7D	}	
126	7E	~	
127	7F	⌂	
128	80	Ç	
129	81	ü	
130	82	é	
131	83	â	
132	84	ä	
133	85	à	
134	86	å	
135	87	ç	
136	88	ê	
137	89	ë	
138	8A	è	
139	8B	ï	
140	8C	î	
141	8D	ì	
142	8E	Ä	
143	8F	Å	
144	90	É	
145	91	æ	
146	92	Æ	
147	93	ô	
148	94	ö	
149	95	ò	
150	96	û	
151	97	ù	
152	98	ÿ	
153	99	Ö	
154	9A	Ü	
155	9B	¢	
156	9C	£	
157	9D	¥	
158	9E	Pt	
159	9F	*f*	
160	A0	á	
161	A1	í	
162	A2	ó	
163	A3	ú	
164	A4	ñ	

ASCII Value	***Hexidecimal Form***	***Character***	***Control Character***
165	A5	Ñ	
166	A6	ª	
167	A7	º	
168	A8	¿	
169	A9	⌐	
170	AA	¬	
171	AB	½	
172	AC	¼	
173	AD	¡	
174	AE	«	
175	AF	»	
176	B0	▒	
177	B1	▒	
178	B2	▓	
179	B3	│	
180	B4	┤	
181	B5	╡	
182	B6	╢	
183	B7	╖	
184	B8	╕	
185	B9	╣	
186	BA	║	
187	BB	╗	
188	BC	╝	
189	BD	╜	
190	BE	╛	
191	BF	┐	
192	C0	└	
193	C1	┴	
194	C2	┬	
195	C3	├	
196	C4	─	
197	C5	┼	
198	C6	╞	
199	C7	╟	
200	C8	╚	
201	C9	╔	
202	CA	╩	
203	CB	╦	
204	CC	╠	
205	CD	═	
206	CE	╬	
207	CF	╧	
208	D0	╨	
209	D1	╤	
210	D2	╥	
211	D3	╙	

ASCII Value	***Hexidecimal Form***	***Character***	***Control Character***
212	D4	╘	
213	D5	╒	
214	D6	╓	
215	D7	╫	
216	D8	╪	
217	D9	┘	
218	DA	┌	
219	DB	■	
220	DC	▄	
221	DD	▌	
222	DE	▐	
223	DF	▀	
224	E0	α	
225	E1	β	
226	E2	Γ	
227	E3	π	
228	E4	Σ	
229	E5	σ	
230	E6	µ	
231	E7	τ	
232	E8	Φ	
233	E9	Θ	
234	EA	Ω	
235	EB	δ	
236	EC	∞	
237	ED	ϕ	
238	EE	∈	
239	EF	∩	
240	F0	≡	
241	F1	±	
242	F2	≥	
243	F3	≤	
244	F4	⌠	
245	F5	⌡	
246	F6	÷	
247	F7	≈	
248	F8	°	
249	F9	•	
250	FA	·	
251	FB	$\sqrt{\ }^{n}$	
252	FC	^{2}	
253	FD		
254	FE	■	
255	FF		(blank 'FF')

INDEX

A

Alt (Alternate) key, 9
ANSI color codes, 156
ANSI.SYS, 153
APPEND command, 141
Application programs (*See* Software)
Archive bit, 72, 143
Archive files, 223
ASSIGN command, 142, 167
ASCII characters, Appendix D
ATTRIB command, 39, 71
AUTOEXEC.BAT file, 28, 90, 151, 167
Automenu, 216, 229–240
AUX1, 32

B

.BAK files, 33
.BAS files, 33
.BAT files, 33
Backslash, 121–123
Backspace key, 9
BACKUP command, 124, 129
Batch files, 89
  CALL command, 187
  creating with COPY CON, 90
  creating with EDLIN, 181–193
  ECHO command, 92
  FOR command, 185–187
  IF command, 182–185
  GOTO command, 182
  Looping in, 182
  PAUSE command, 92
  REM statement, 92
  replaceable parameters in, 91, 183
Batch processing, 17
Baud rates, 12
Binary digit (bit), 5
Booting the system, 27–29
Brackets, 7, 32
BREAK command, 53, 152
Break key, 8
Buffer, 10, 168
BUFFERS option in CONFIG.SYS, 152
Bus, 13, 20
Byte, 5

C

CALL command, 187
Caps Lock key, 8, 9
CD command (CHDIR), 125
Central processing unit (CPU), 4, 13
CGA (Color Graphics Adapter) card, 10
CHKDSK command, 71, 73
Clock rate, 5
CLS command, 53
Cluster, 26
Cold boot, 127
COM files, 32, 141
COM1, 32
COM2, 32
Combination keys, 8, 28
COMMAND.COM file, 28
Commands
  batch file, 89–92, 181–193
  external, 38
  general format of, 51
  internal, 38
  hard disk, 119–130, 139–143
  options in, 51
  parameters in, 51
  using DOS editing keys, 110
COMP command, 71, 75
Computer system
  booting, 27–29
  components of, 3–18
  turning on, 27–30
Concatenating files, 55
CON device, 31, 55
CONFIG.SYS file, 28, 151–160, 171
Configuring your system, 151–160
COPY command, 53
  COPY CON, 55
Copy in EDLIN, 109
Ctrl (control) key, 9
Ctrl-Alt-Del, 28
Ctrl-Break, 8, 52
Ctrl-C, 52
Ctrl-Prt-Sc, 8
Ctrl-S, 8, 52
Ctrl-Z, 55
Current directory, 122
Current line in EDLIN, 108
Cursor, 29

D

.DAT files, 33
.DBF files, 33
.DEF files, 33
.DOC files, 33
DATE command, 30, 53, 57
DEBUG command, 171, 177, 192
Default disk drive, 29, 31
Default prompt, 29
DEL command, 53, 57, 58
Del (delete) key, 9, 110
Delete in EDLIN, 109
Delimiter, 52
Desktop manager, 211
Device driver, 153
DEVICE option in CONFIG.SYS, 153
DIR command, 35, 53, 58
  DIR /P, 37, 59
  DIR /W, 37, 59
Directory
  current, 122
  root, 120
  subdirectory, 120
DISKCOMP command, 71, 76
DISKCOPY command, 71, 76
Disks
  backup, 53, 123, 129 140
  data, 26
  default, 29
  drive designator, 31
  floppy, 14
  formatting, 39
  hard, 15, 119–130, 140
  naming convention, 32
DOS
  definition of, 24
  editing keys, 110
  error messages, Appendix C
  functions of, 25
  MS-DOS, 24
  PC-DOS, 24
  shell, 18, 205
  Version 4.0, 249
  versions, 38
DOS commands
  summary, Appendix A
  APPEND, 141
  ASSIGN, 142, 167
  ATTRIB, 39, 71
  BACKUP, 124, 129
  BREAK, 54, 152
  CD (CHDIR), 124
  CHKDSK, 73
  CLS, 54
  COMP, 71, 75

COPY, 54–56
DATE, 30, 57
DEBUG, 171, 177, 192
DEL, 53, 57
DIR, 35, 53, 58
DISKCOMP, 76
DISKCOPY, 76
EDLIN, 107–111
ERASE (*See* DEL)
FASTOPEN, 166
FDISK, 124
FIND, 77
FORMAT, 78, 125
LABEL, 79
MD (MKDIR), 124, 126
MODE, 158–160
MORE, 96–98
PATH, 124, 127
PRINT, 167
PROMPT, 60, 141, 155
RD (RMDIR), 124, 125
RECOVER, 79
RENAME (REN), 61
REPLACE, 169
RESTORE, 124, 130
SORT, 96
SUBST, 170
SYS, 80
TIME, 30, 62
TREE, 124, 128
TYPE, 53, 63
VER, 53, 63
VERIFY, 64
VOL, 64
XCOPY, 143
DOS Error Messages, Appendix C
Dot matrix printer (*See* Printers)
Drive designator, 31

E

.EXE files, 33
ECHO (*See* Batch files)
Editing keys, 110
EDLIN commands, 107, 111
EGA (Enhanced Graphics Adapter)
  card, 11
End command in EDLIN, 109
End Of File (EOF), 55
Enter key, 9
ERASE command (*See* DEL)
ERRORLEVEL option in IF, 184, 192
Esc (Escape) key, 9, 110
EXIST option in IF, 183
Expansion slots, 13
Extensions
  general, 32
  naming, 32
  standard, 33
External commands, 38

F

FASTOPEN command, 166
FAT (*See* File allocation table)
FDISK command, 124
Files
  attributes of, 39
  backing up, 53, 124, 129, 143, 217
  batch, 89
  copying, 53
  data, 26
  definition, 26
  deleting, 53, 57
  displaying contents of, 53, 63
  extensions, 32
  fragmented, 26, 73
  hidden, 72, 143
  management of, 24, 205–219
  naming, 32
  organizing, 33
  program, 26
  recovery, 29, 210
  saving, 26
  tagging, 209
  transferring system, 40, 125
  types of filename extensions for, 33
File allocation table (FAT), 27
File management utility program,
  203–223
FILES option in CONFIG.SYS, 153
Filters, 96–98
  FIND, 97
  MORE, 98
  SORT, 96
FIND command, 71, 77
Firmware, 18
Fixed disks (*See* Hard disks)
Floppy disks, 14
FORMAT command, 40, 71, 78, 125
  FORMAT /S, 40, 125
  FORMAT /V, 40, 125
Formatting, 39, 71, 78, 125
Fragmented files, 26, 73
Freeware, 218
Full filename, 32
  drive designator, 32
  filename, 32
  filename extension, 32
  path, 32, 122
Function keys, 8, 110, 154

G

GOTO command (*See* Batch files)
Greater-than sign (>)
  prompts, 29
  redirection, 89, 95

H

.HLP files, 33
Hard disks, 15, 119–130, 139–143
Hardware, 4
Head crash, 119
Hidden files, 27, 39, 71
Hierarchical organization, 120
Hot key, 208

I

IBMBIO.COM file, 25
IBMDOS.COM file, 25
IF statement (*See also* Batch files)
  branch locations, 182
  ERRORLEVEL, 184, 192
  EXIST, 182
  GOTO, 182
  strings, 184
Input/output devices, 6–16
Input buffer, 110
Insert in EDLIN, 109
Ins (Insert) key, 9
Interactive processing, 17
Interface boards, 13
Internal DOS commands, 38, 53–65
I/O (*See* Input/output devices)
I/O interface, 13
IO.SYS file (*See* SYS files)

K

Keyboards, 7
Keys
  combination, 8
  function, 8
  hot, 208
  redefinition of, 141, 155
  standard keyboard, 7–10
Kilobyte (KB), 6

L

LABEL command, 39, 71, 79
Laser printer (*See* Printers)
LASTDRIVE command, 171
Letter quality printer (*See* Printers)
LIM EMS 4.0, 250

Line editor (*See* EDLIN)
List Lines in EDLIN, 109
Lost allocation clusters, 74
LPT1, 31

M

.MSG files, 33
Macro generator, 204
Magnetic tape, 16
Memory
 expanded, 250
 extended, 250
 main memory, 6
 read-only, 18
MD command (MKDIR), 124, 126
Megabyte (MB), 14
Megahertz (MHz), 5
Menu-driven systems, 227
Menu Definition File (MDF), 229, 232
Menus, 204, 227
 Automenu, 227–240
 pull-down, 206, 208
Microcomputer, 14
 software, 15
Microprocessor, 4
Microsoft's Windows, 208
MODE command, 158–160
Modem, 12
Monitor
 color, 10
 monochrome, 10
MORE command (*See* Filters)
Mouse, 12
Move in EDLIN, 109
MS-DOS, 23
MSDOS.SYS file (*See* System files)
Multifunction boards, 13
Multitasking, 251

N

Norton Utilities, 209–211
NUL device, 31, 141, 189
Numeric keypad, 8
Num Lock key, 8

O

.OVL files, 33
Operating system, 23
Options for commands, 51
Organizing files, 33, 120
OS/2, 250

P

Parameters
 in commands, 51
 replaceable, 91, 183
Parent directory, 122
Parking the heads, 120
Partition, 124
Path, 32, 122
PATH command, 141
PAUSE command (*See* Batch files)
Pause option, 37, 59
PC Tools Deluxe, 214
PC-Magazine, 215
 ATTR, 219
 BLANKINS, 220
 BROWSE, 220
 DDIR, 220
 PRSWAP, 221
 SWEEP, 221
 VTREE, 221
 WHERE, 222
Piping, 89, 95
Pixels, 10
Point and shoot, 209
Port
 parallel, 12
 RS-232, 12
 serial, 12
Primary storage, 4
Print buffer, 168
PRINT command, 167
Print spooler, 168
Printers
 dot matrix, 11
 draft-quality, 11
 laser, 11
 letter-quality, 11
PRN, 31, 168
 files, 168
Programs (*See* Software)
PROMPT command, 53, 60, 141, 155
PrtSc (print screen) key, 9
Public domain, 218

Q

Quit in EDLIN, 109

R

RAM (Random Access Memory), 6, 14, 152, 166
RAM disk, 157
RAMDRIVE.SYS file (*See* System files)
RAM-resident programs, 204, 207
RD command (RMDIR), 124–127
Read-only files, 72
Record, 26
RECOVER command, 71, 79
Redirection, 89, 95
REM statement (*See* Batch files)
Removing subdirectories, 124, 127
RENAME command (REN), 53, 61
Replaceable parameter, 91, 183
Replace in EDLIN, 109
REPLACE command, 169
Reserved names, 31
RESTORE command, 124, 130
Return key (*See* Enter key)
ROM (Read-only Memory), 13, 18, 28
Root directory, 120

S

.SYS, 25, 33, 157
Saving files, 26
Screen resolution, 10
Scrolling, 9, 35, 53, 63
Scroll Lock key, 9
Search in EDLIN, 109
Secondary storage, 14
Sector, 26
Shareware, 218
Shell (*See* DOS shell)
Shift key, 8
Shift-PrtSc key, 8
SideKick Plus, 211
Slash, 52
Software
 application programs, 17
 definition of, 16
 freeware, 218
 integrated, 17
 public domain, 218
 shareware, 218
 system, 17
 utility support programs, 203–223
SORT command (*See* Filters)
Spooling, 167

Standard device names, 32
Standby power supply (SPS), 119
Static electricity, 6
Storage
  primary, 4–6
  secondary, 14–16
String = = String in IF, 184
Strings, 77, 182–185
Subdirectory (*See* Directory)
SUBST command, 142, 167, 170
SYS command, 71, 80
System (SYS) files
  IO.SYS, 25
  MSDOS.SYS, 25
  RAMDRIVE.SYS, 57
  VDISK.SYS, 39, 157
System prompt, 29
System software (*See* Software)

T

.TXT files, 33
Tagging files, 209
Tape (*See* Magnetic tape)
Terminate and stay resident (TSR), 204, 207
The Norton Utilities Advanced Edition, 209
TIME command, 30, 53, 62
Toggle key, 8
Track, 26
TREE command, 124, 128
Treeview, 212–213
TYPE command, 53,63

U

Utility support programs (*See* Software)

V

VDISK.SYS file (*See* SYS file)
VER (Version) command, 53, 63
VERIFY command, 53, 64
Versions of DOS, 38
VGA (Video Display Adapter) card, 10
Virus, 223
VOL command, 53, 64
Volume labels, 53, 64, 190

W

.WKS files, 33
Warm boot, 27
Wide option, 37, 59
Wildcards, 30, 34
Windows, 204, 207
Word, 5

X

XCOPY command, 143

DOS: Complete and Simplified offers comprehensive coverage of the commands and uses of the IBM Disk Operating System for students using IBM and compatible computers for the first time. Beginning with coverage of the fundamental concepts of microcomputers and operating systems in general, this text covers PC-DOS (and MS-DOS) topics including hardware and software, internal and external DOS commands, DOS editing capabilities, batch files, hard disk concepts and management, beginning and advanced DOS commands, customization of DOS, and utility support programs.

DISTINGUISHING FEATURES:

- Easy-to-follow presentation of commands and associated options
- Up-to-date coverage of DOS 4.0 and OS/2 highlights, including the pros and cons of upgrading
- Common DOS error messages, what they mean, and corrective courses of action
- An appendix on AUTOMENU and TREEVIEW, hard disk management software from Magee Enterprises, with free software and software support for registered adopters of the text
- Coverage of utility support programs, including shareware and public domain software, as well as commercial utility support programs

BF4921

ISBN 0-87835-492-1

9780878354924

10/17/2016 13:02-3

Boyd & Fraser Publishing Company
A Division of South-Western Publishing Company
20 Park Plaza, Boston, MA 02116