Microsoft® MS-DOS®

Operating System version 5.0

User's Guide and Reference

An important note to MS-DOS users

The hologram on the spine of this manual is your assurance of the genuine Microsoft MS-DOS operating system. If it is missing or appears altered, this may not be the official version from Microsoft—please notify your dealer immediately. In this way, we can continue to provide you with the quality you expect from the world's leader in personal computer software.

Microsoft®

Microsoft® MS-DOS®

User's Guide and Reference

for the MS-DOS® Operating System

VERSION 5.0

Microsoft Corporation

Information in this document is subject to change without notice and does not represent a commitment on the part of Microsoft Corporation. The software and/or databases described in this document is furnished under a license agreement or nondisclosure agreement. The software and/or databases may be used or copied only in accordance with the terms of the agreement. It is against the law to copy the software on any medium except as specifically allowed in the license or nondisclosure agreement. The purchaser may make one copy of the software for backup purposes. No part of this manual and/or databases may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording, or information recording and retrieval systems, for any purpose other than the purchaser's personal use, without the express written permission of Microsoft.

© 1991 Microsoft Corporation. All rights reserved.
Printed in the United States of America.

Mirror, Undelete, and Unformat © 1987-1991 Central Point Software, Inc.

Unless otherwise noted, all names of companies, products, street addresses, and persons contained herein are part of a completely fictitious scenario or scenarios, and are designed solely to document the use of a Microsoft product.

Microsoft, the Microsoft logo, MS, MS-DOS, Microsoft Excel, GW-BASIC, and XENIX are registered trademarks and *Making it all make sense*, QBasic, Windows, and Windows/386 are trademarks of Microsoft Corporation.

U.S. Patent No. 4955066

Acer is a registered trademark of Acer Technologies Corporation.
PostScript is a registered trademark of Adobe Systems, Inc.
AT&T is a registered trademark of American Telephone and Telegraph Company.
AST is a registered trademark of AST Research, Inc.
COMPAQ is a registered trademark of Compaq Computer Corporation.
CompuServe is a registered trademark of CompuServe, Inc.
Dow Jones News/Retrieval is a registered trademark of Dow Jones & Company, Inc.
Epson is a registered trademark of Epson America, Inc.
Hewlett-Packard, HP, LaserJet, PaintJet, ThinkJet, and Vectra are registered trademarks and DeskJet is a trademark of Hewlett-Packard Company.
Intel is a registered trademark and 386 is a trademark of Intel Corporation.
AT, IBM, Personal System/2, PS/2, and Quietwriter are registered trademarks and PC/XT and Proprinter are trademarks of International Business Machines Corporation.
Lotus and 1-2-3 are registered trademarks of Lotus Development Corporation.
Philips is a registered trademark of Philips International B.V.
Toshiba is a registered trademark of Kabushiki Kaisha Toshiba.
Tulip is a registered trademark of Tulip Computers International B.V.
Weitek is a registered trademark of Weitek Corporation.
WordPerfect is a registered trademark of WordPerfect Corporation.
Wyse is a registered trademark of Wyse Technology.
Zenith is a registered trademark of Zenith Radio Corporation.

Document No. SY07661/20885-0391

Contents

Welcome ..... **ix**

New Features of MS-DOS Version 5.0 x
Conventions xii

PART 1 MS-DOS Fundamentals

Chapter 1 Learning About Your Computer **3**

Looking at Hardware 3
Using Disks and Disk Drives 10
Using Software 13
Looking at MS-DOS 14
Using Files and Directories 15

Chapter 2 Command-Line Basics **19**

Parts of a Command 19
Typing a Command 20
Shortcuts to Typing a Command 21
How MS-DOS Responds to a Command 22
Stopping or Canceling a Command 23
Designating a Disk Drive 23
Internal and External Commands 24
Getting Help with a Command 24

Chapter 3 MS-DOS Shell Basics **27**

Starting MS-DOS Shell 27
The MS-DOS Shell Window 28
Working with Menus 32
Working with Dialog Boxes 34
Changing Views 41
Working with Files and Directories 42
Working with Programs 53
Associating Files with a Program 58
Suppressing Confirmation Messages 60

Repainting and Updating the Screen 61
Getting Help 62
Leaving MS-DOS Shell 65

PART 2 Working with MS-DOS

Chapter 4 Working with Files 69

Filenames 69
Types of Files 70
File Size, Date, and Time 72
Using Wildcards 73
Viewing Text Files 75
Copying Files 77
Renaming a File 83
Printing Text Files 83
Deleting Files 86
Moving Files 92
Comparing Files 93
Viewing and Changing File Attributes 94
Finding Text Within a File 97
Searching for Files by Using MS-DOS Shell 98
Getting Information by Using MS-DOS Shell 99

Chapter 5 Working with Directories 101

Understanding Directories 101
Viewing Directories 107
Creating Directories 113
Moving Between Directories 114
Deleting Directories 116
Copying Directories 118
Updating Directories 122
Specifying a Search Path 123

Chapter 6 Managing Disks 125

Types of Disks 125
Bytes, Kilobytes, and Megabytes 126
Types of Disk Drives 126
Formatting Disks 127
Labeling a Disk 134
Making a Backup Disk 136
Restoring Directories and Files 141
Recovering Files from Defective Disks 145
Substituting a Drive Letter with a Directory 146
Partitioning Your Hard Disk 147
Using Fdisk 149

Chapter 7 Advanced Command Techniques 161

Redirecting Command Input and Output 161
Passing Information Through Filter Commands 163
Using Editing Keys 166
Using Doskey to Work with Commands 169
Using Doskey to Work with Macros 175

Chapter 8 Customizing MS-DOS Shell 183

Changing Screen Colors 183
Switching Between Text and Graphics Mode 184
Organizing Programs 184
Working with Properties 190

Chapter 9 Working with MS-DOS Editor 203

Getting Started with MS-DOS Editor 204
Using Help 209
Quitting MS-DOS Editor 211
Creating a Text File 211
Editing Text 214
Managing Files 217
Customizing MS-DOS Editor 221

PART 3 Customizing MS-DOS

Chapter 10 Working with Batch Programs 225

Understanding Batch Programs 225
Creating a Small Batch Program 228
Displaying Messages with a Batch Program 229
Using the Pause Command 231
Including Remarks in a Batch Program 232
Running One Batch Program from Another 233
Using Replaceable Parameters 234
Controlling Program Flow 235
Creating a Menu System 238

Chapter 11 Customizing Your System 245

Creating a Startup Procedure 246
Configuring MS-DOS for Your System 249
Configuring Your Ports 255
Adding Disk Drives 258
Modifying Your Screen and Keyboard 261

Chapter 12 Optimizing Your System 273

Understanding System Resources 274
Making More Memory Available 279
Speeding Up Your System 291
Running Programs in the Upper Memory Area 313
Optimization Summary 331

Chapter 13 Customizing for International Use 333

Changing Conventions and Keyboards 335
Using Code Pages 339
Sample Language Changes 350

PART 4 MS-DOS Reference

Chapter 14 Commands 357

Types of Commands 357
Syntax Conventions 360
Online Help with Commands 362
Alphabetic Listing of Commands 363

Chapter 15 Device Drivers 591

Appendix

Keyboard Layouts and Code Pages 623

Keyboard Layouts 623
Code-Page Tables 631

Index 639

Welcome

Welcome to Microsoft® MS-DOS®, the most widely used operating system for personal computers. MS-DOS version 5.0 includes many new features that turn your computer into a powerful tool for your business or personal use. These features are described in detail later in this introduction.

If MS-DOS version 5.0 is not yet set up on your computer system, you will need to use the Setup program. For information about how to set up MS-DOS, see *Microsoft MS-DOS Getting Started.*

This guide is organized as a comprehensive reference and is arranged by task. It includes information for users who have little experience with computer systems, in addition to advanced topics for those who are familiar with MS-DOS. It is not necessary to read this guide straight through. Consult the table of contents or index to learn how to use the commands you work with most often.

This guide is divided into the following parts:

- *Part 1: MS-DOS Fundamentals.* Explains the essentials of a computer system and MS-DOS. This section also describes how to use the MS-DOS command line and MS-DOS Shell.

- *Part 2: Working with MS-DOS.* Explains how to use MS-DOS to manage files, directories, and disks. Also included are descriptions of how to use advanced command techniques, how to customize MS-DOS Shell, and how to use the new MS-DOS full-screen editor.

  "In Brief" cues at the beginning of each section serve as quick reminders of which command and option you need to accomplish a task. Detailed procedures following the command examples outline tasks that can be performed from MS-DOS Shell.

- *Part 3: Customizing MS-DOS.* Provides information about advanced topics, such as using batch programs, customizing your computer system, and using MS-DOS internationally.

- *Part 4: MS-DOS Reference.* Provides an alphabetical reference, with complete descriptions of MS-DOS commands and installable device drivers. Intermediate and advanced users can use this reference to learn about the full capabilities of MS-DOS.

- *Appendix.* The appendix contains information about keyboard configurations and code pages.

New Features of MS-DOS Version 5.0

If you have used earlier versions of MS-DOS, you will find many improvements and new features in MS-DOS version 5.0. These features include changes to the way MS-DOS uses memory and new commands and programs. If you run MS-DOS programs with Microsoft Windows™ graphical environment version 3.0, you will want to use MS-DOS version 5.0 because its memory features make your MS-DOS programs run faster and more efficiently.

MS-DOS version 5.0 has the following new features:

- The ability to run MS-DOS in the high memory area (HMA). If your system has extended memory, you can now run MS-DOS in the high memory area (HMA) instead of in conventional memory. This makes more conventional memory available to your programs, enabling them to run faster and more efficiently.

  If you use Microsoft Windows version 3.0, this feature is of added benefit. By running MS-DOS in HMA, you can run several MS-DOS programs at the same time with improved speed and performance. For more information about using this feature, see Chapter 12, "Optimizing Your System."

- The ability to run certain device drivers and programs in the upper memory area if you have an 80386 or higher system. This ability also makes more conventional memory available to your programs, enabling them to run faster and more efficiently.

As with running MS-DOS in HMA, running device drivers in the upper memory area improves the speed and performance of your MS-DOS programs when you run them with Microsoft Windows version 3.0. For details about using this feature, see Chapter 12, "Optimizing Your System."

- MS-DOS Shell, an improved graphical interface that you can use to manage programs and switch between them. With MS-DOS Shell, you can view the directory structure of any disk, view the contents of several directories, and navigate through your files and directories quickly. For details about using MS-DOS Shell, see Chapter 3, "MS-DOS Shell Basics," and Chapter 8, "Customizing MS-DOS Shell."

- Added data security. MS-DOS version 5.0 includes two new commands, **unformat** and **undelete**. These commands allow you to undo a format or delete operation, restoring your disk or file to its original state. For details about using the **undelete** command, see Chapter 4, "Working with Files." For details about using the **unformat** command, see Chapter 6, "Managing Disks."

- Online Help for all MS-DOS commands and MS-DOS Shell. You can get help for a particular MS-DOS command by typing the command name followed by **/?**, or by typing **help** followed by the command name. For more information about getting help for commands, see Chapter 2, "Command-Line Basics." For information about MS-DOS Shell Help, see Chapter 3, "MS-DOS Shell Basics."

- MS-DOS Editor, a new full-screen text editor that you can use to easily create and modify text files. MS-DOS Editor also includes online Help. For information about how to use MS-DOS Editor, see Chapter 9, "Working with MS-DOS Editor."

- The ability to create large disk partitions easily. With MS-DOS version 5.0, you can create disk partitions of up to 2 gigabytes. If you are upgrading from MS-DOS version 4.0, you will no longer need to use the Share program to take full advantage of this feature. For more information about partitioning your hard disk, see Chapter 6, "Managing Disks."

- The ability to search for files through multiple levels of directories.

- Added functionality to the **dir** command that enables you to sort directory listings by filename, type of file, the date and time that files were created, and file size. You can also specify a default setting for sorting directory listings by using an environment variable. For more information about using the **dir** command, see Chapter 5, "Working with Directories." For more information about specifying a default setting for sorting directory listings, see the **dir** command in Chapter 14, "Commands."

- Doskey, a program you can use to recall, edit, and carry out commands that you have already used. If you perform a series of tasks frequently, you can also use Doskey to create macros. Macros make it easy for you to perform a series of tasks without having to type several commands. For more information about using Doskey, see Chapter 7, "Advanced Command Techniques" or the **doskey** command in Chapter 14, "Commands."

- MS-DOS QBasic, an improved Basic programming environment that includes extensive online Help. To use QBasic, type **qbasic** at the command prompt or run it from MS-DOS Shell. (Throughout this guide, "QBasic" refers to MS-DOS QBasic.)

- The ability to access more than two hard disk drives. For more information about this feature, see Chapter 6, "Managing Disks."

- Support for 2.88-megabyte floppy disks. For more information about working with floppy disks, see Chapter 6, "Managing Disks."

- A new Setup program with online Help, which guides you through each step of the setup process and provides help whenever you need it. For more information about using the Setup program, see *Microsoft MS-DOS Getting Started.*

Conventions

This guide uses particular document and keyboard conventions to help you locate and identify information.

For more information about conventions used with MS-DOS version 5.0, see the "Syntax Conventions" section in Chapter 14, "Commands."

Document Conventions

To help you locate and identify information easily, this guide uses visual cues and standard text formats. The following typographical conventions are used in this guide:

Type style	Used for
bold	Command names, switches, and any text you must type exactly as it appears. To carry out a command, type the command name and then press ENTER.
italic	Parameters. You can supply the text for any item shown in italic. For example, if you want to use a parameter that calls for a *filename*, you must type the name of the specific file. New terms appear in italic type. The term is explained when it first occurs. Italic type is also used for emphasis in certain examples.
ALL CAPITALS	Directory names, filenames, and acronyms. When you type directory names and filenames, you can use lowercase letters.

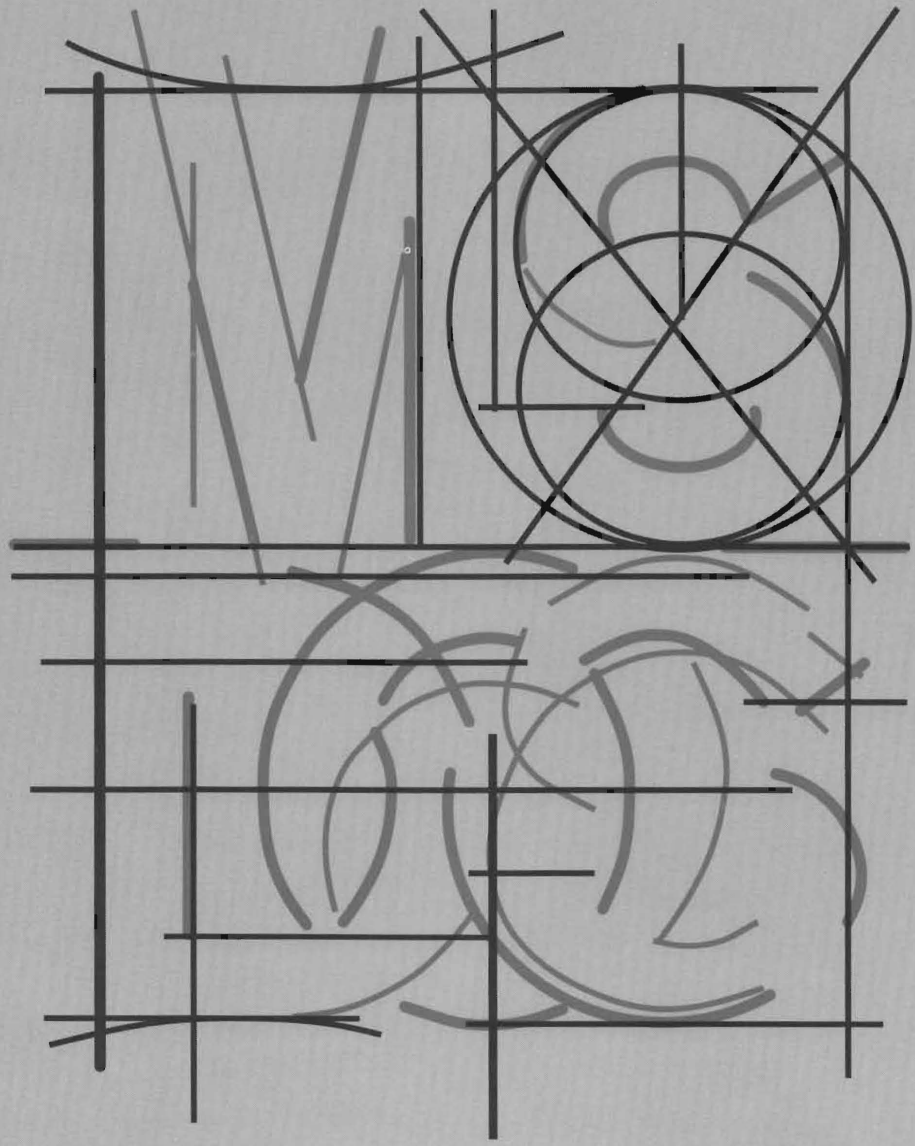
Microsoft documentation uses the term "MS-DOS" to refer to the MS-DOS and IBM Personal Computer DOS operating systems.

Keyboard Conventions

Key combinations and key sequences appear in the following format:

Notation	Meaning
KEY1+KEY2	A plus sign (+) between key names means you must press the keys at the same time. For example, "Press CTRL+C" means that you press CTRL and hold it down while you press C.
KEY1, KEY2	A comma (,) between key names means you must press the keys in sequence—for example, "Press ALT, F10" means that you press the ALT key and release it, and then press the F10 key and release it.

Part 1
MS-DOS Fundamentals



Chapter 1

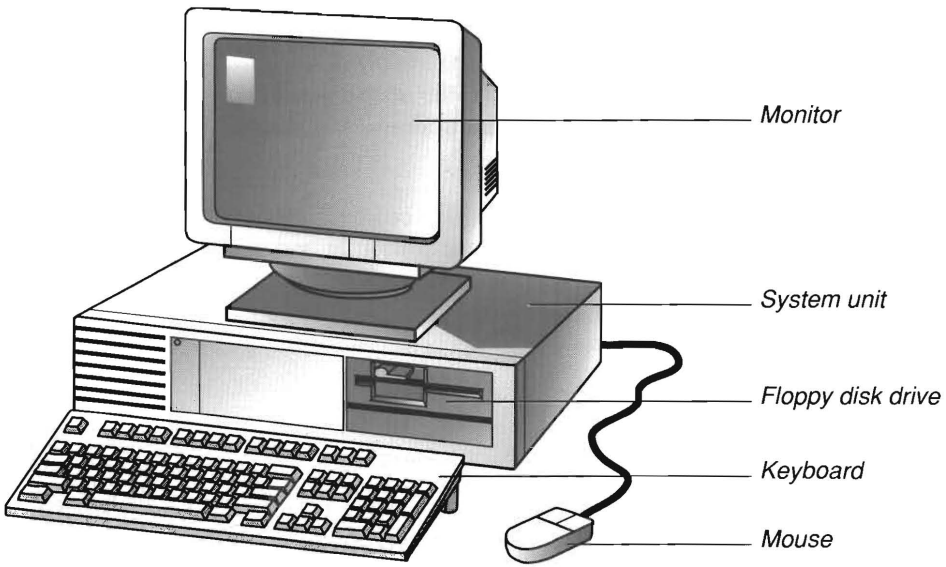
Learning About Your Computer

1

Your computer is made up of many parts called *hardware*. Your hardware runs *software*—programs that translate the instructions you send to your computer into a language it can understand. If you are not familiar with the terms *hardware*, *floppy disk*, and *operating system*, you should read this chapter before setting up or using MS-DOS.

Looking at Hardware

The hardware that makes up the most basic computer system includes a monitor, a keyboard, and a system unit. The system unit holds your computer's processor, memory, disk drives, ports, and video card.



Processor and Memory

The *central processing unit* (CPU) and memory are located on chips inside the system unit. The CPU is the brain of your computer. This is the place where your computer interprets and processes information.

You may have heard computer memory referred to as RAM. The term RAM stands for *random-access memory*. The instructions that your computer gets and the information your computer processes are kept in RAM during your work session.

Your computer's RAM is not a permanent storage place for information; it is active only when your computer is on. When you turn off your computer, information is deleted from memory. To avoid losing your work, remember to save it on a disk, a permanent storage device, before turning off your computer.

Computer memory is measured in kilobytes or megabytes of information. (A byte is the amount of storage needed to hold one character.) One kilobyte equals 1024 bytes, and one megabyte equals 1,048,576 bytes. Therefore, if your system has 640K of memory, it can hold 655,360 bytes at one time. Software requires a minimum amount of random-access memory to work properly. You can usually find memory requirements on software packaging, or you can ask your software dealer.

Monitor

The monitor has a screen that displays information, such as the instructions you send to your computer and the information and results your computer sends back after interpreting your instructions. The screen may display information in one color or in several colors.

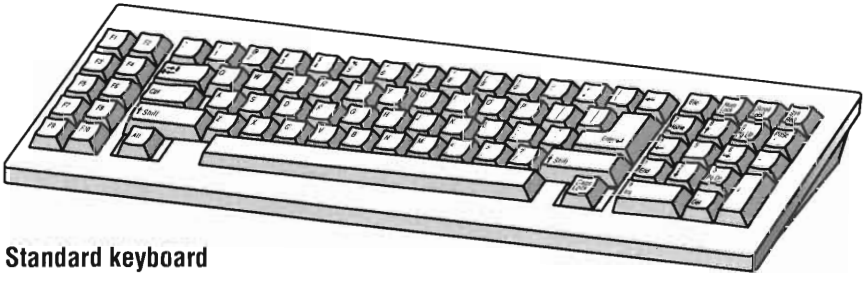
A color monitor won't display color unless you have the appropriate video card. A video card fits inside your computer and determines the screen resolution and number of colors your monitor can display. Some video cards enable your monitor to display graphical information, such as geometric designs, in addition to text and numbers.

Keyboard

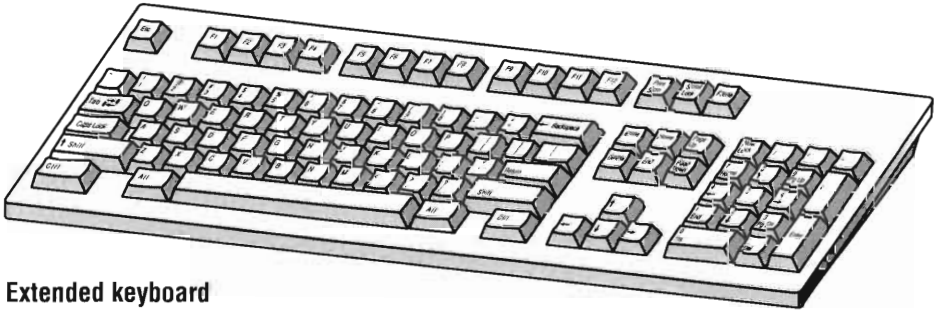
You use the keyboard to type instructions for your computer, and to type information you want your computer to process. All keyboards have letter keys, punctuation keys, and a spacebar, which resemble the keys on a typewriter. Most keyboards also have function, numeric, and arrow keys, in addition to ALT, CTRL, DEL, and ENTER or RETURN keys. Their placement on the keyboard is determined by the computer manufacturer; how they are used is determined by the software you are using.

Standard Keys

You use the letter keys, punctuation keys, and spacebar the same way you use them on a typewriter.



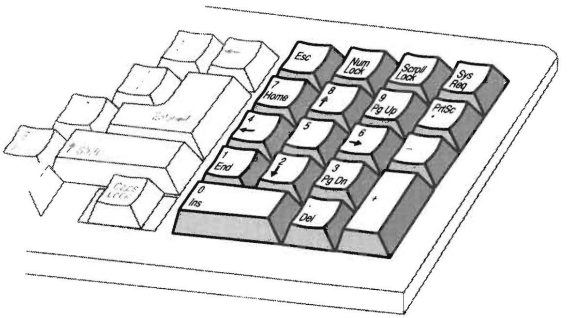
Standard keyboard



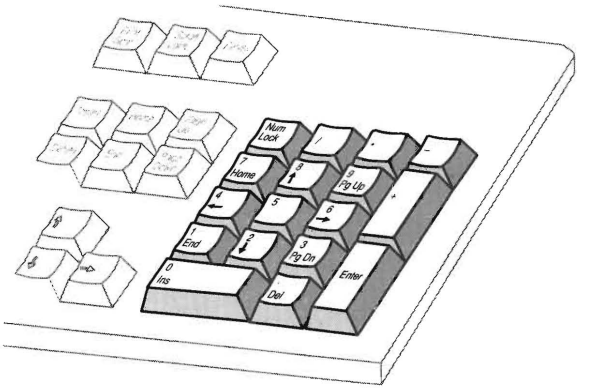
Extended keyboard

Numeric Keypad

Because the number keys are grouped together on the numeric keypad, the keypad is a quick and easy way to type numbers. With many software products, you must press the NUM LOCK key before using the numeric keypad to type numbers. The NUM LOCK key works somewhat like the CAPS LOCK key on your typewriter. When you press the NUM LOCK key on the numeric keypad, the numbers you press on the keypad appear on your screen.



Standard keyboard

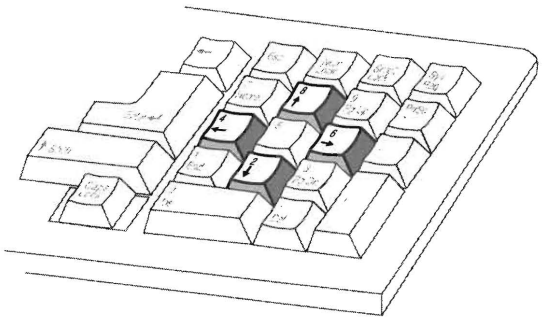


Extended keyboard

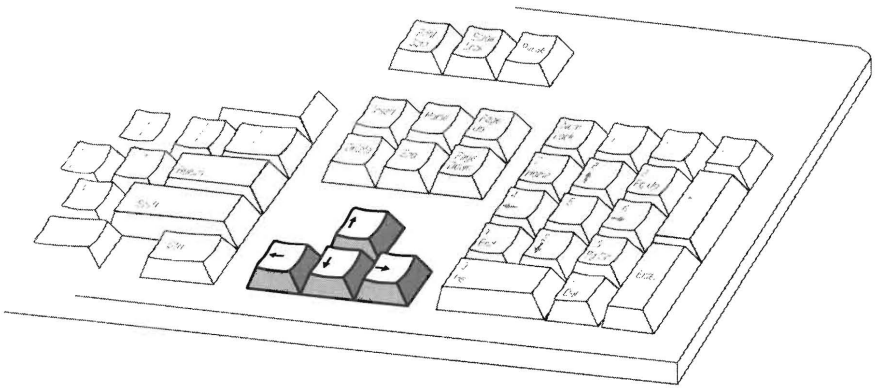
Arrow Keys

In most cases, the arrow keys help you move around the screen. Like other keys, however, their exact function is determined by the software you are using. On standard keyboards, the arrow keys are on the numeric keypad, along with the numbers. To use these keys properly, the NUM LOCK key must not be active. If NUM LOCK is active, you will type a number instead of changing your position on the screen when you use an arrow key.

On extended keyboards, arrow keys appear to the lower left of the numeric keypad. You can use these arrow keys anytime, regardless of whether the NUM LOCK key is active.



Standard keyboard



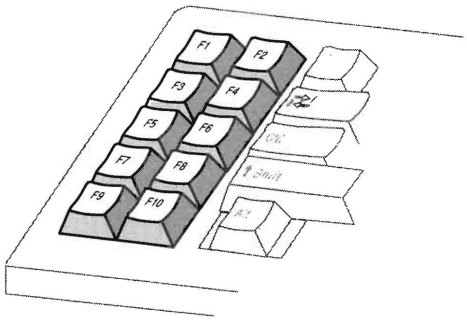
Extended keyboard

Function Keys

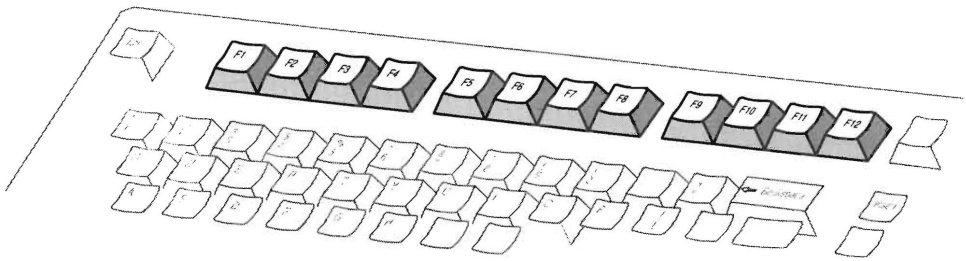
Function keys send instructions to the software you are using. For example, suppose you are writing a memo and you have trouble moving a paragraph. By pressing a function key, you may be able to get Help information on your screen.

Like other keys, what a function key does depends on the software you are using. For example, some software uses F1 to display Help information, whereas other software uses the F1 key for a different function (or may not assign a function to that key at all).

The function keys are located across the top on extended keyboards or on the left side of standard keyboard.



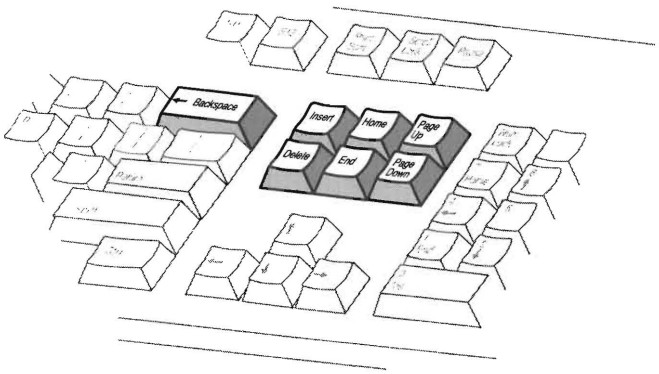
Standard keyboard



Extended keyboard

Additional Keys

In addition to the standard, numeric, arrow, and function keys, extended keyboards include BACKSPACE, INS, DEL, PAGE UP, PAGE DOWN, HOME, and END keys. Typically, you can use the PAGE UP, PAGE DOWN, HOME, and END keys to move around or scroll through information on your screen, and you can use the BACKSPACE, INS, and DEL keys to edit text. As with other keys on your keyboard, what these keys do depends on the software you are using.



The additional keys on an extended keyboard

Ports

Located on the back or front of the system unit are sockets called ports. You use ports to plug in your keyboard, monitor, and any additional hardware, such as a printer or mouse, that you want to add to your system.

Additional Hardware

In addition to the basic components, most computer systems include other hardware, such as a printer, a mouse, and possibly a modem.

- A *printer* prints the information processed by your computer. Printers vary in printing speed and output capabilities. For example, some letter-quality printers can print high-quality text quickly; some dot matrix and laser printers can print both text and graphics.

- By using a *mouse*, you can move a pointer on your screen. You change the position of the pointer by moving the mouse across your table or desktop. You can also use a mouse to select an item by pointing to it and then clicking a button.

- A *modem* connects your system to a telephone line so that you can communicate with another computer in another building, or even across the country. You can use a modem to receive business information from services like the Dow Jones News Retrieval and CompuServe Information Service.

Using Disks and Disk Drives

A disk, like a cassette tape, is a reusable storage device that holds information, such as software and data, in files. As with random-access memory, the amount of space on a disk is measured in bytes. Unlike information stored in random-access memory, however, information stored on a disk is not deleted when you turn off your computer. If you choose to, you can delete the information on a disk, and with proper care, you can use the disk over and over again.

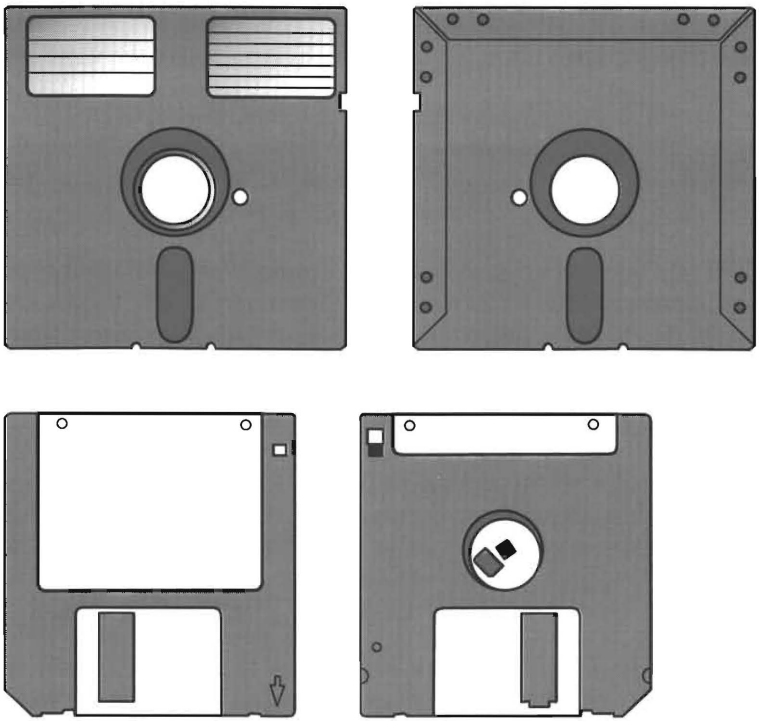
Your computer's disk drives move information stored on the disks into and out of random-access memory. For example, your computer can read software instructions from a disk into random-access memory, and write your data to a disk for safekeeping.

There are two kinds of disk drives: a *hard disk drive* and a *floppy disk drive*. A hard disk drive contains a non-removable disk that is built into your system. With a hard disk drive, you can store large amounts of information in one convenient place, instead of storing it on many floppy disks.

A floppy disk drive holds a removable floppy disk, which has less storage capacity than a hard disk. The hard disk drive can also write and read information to and from your system much faster than a floppy disk drive can write and read information to and from a disk. When your system writes or reads information to or from a disk, the indicator light on the drive goes on.

Each disk drive has a letter assigned to it so you can tell your system where to find instructions and information. For example, on many systems the floppy disk drive is called the A drive, and the hard disk drive is called the C drive.

Floppy disks are removable and come in various sizes. Many systems use 5.25-inch floppy disks. This floppy disk is thin and flexible, and therefore somewhat fragile. Some systems use 3.5-inch floppy disks. This kind of floppy disk is protected by a hard plastic cover.



Front and back view of a 5.25-inch and 3.5-inch floppy disk

Labeling and Caring for a Floppy Disk

Store floppy disks in a safe place, away from dust, moisture, magnetic fields (such as televisions, speakers, and computer monitors), and extreme temperatures. Label each floppy disk so that you can identify the information stored on it. Place the label on the front of the disk, at the top, so that the label does not stick to any exposed areas on the disk.

If you are using a 5.25-inch floppy disk, use a soft, felt-tip pen to label it. Using a pencil or ballpoint pen can damage a floppy disk.

Protecting Information on a Floppy Disk

A 5.25-inch floppy disk has a write-protect notch for protecting information. By placing a small piece of tape, called a write-protect tab, over the notch, you can

protect the information on the disk from being changed. A 3.5-inch disk can be write-protected by sliding a built-in tab over to reveal the write-protect hole.

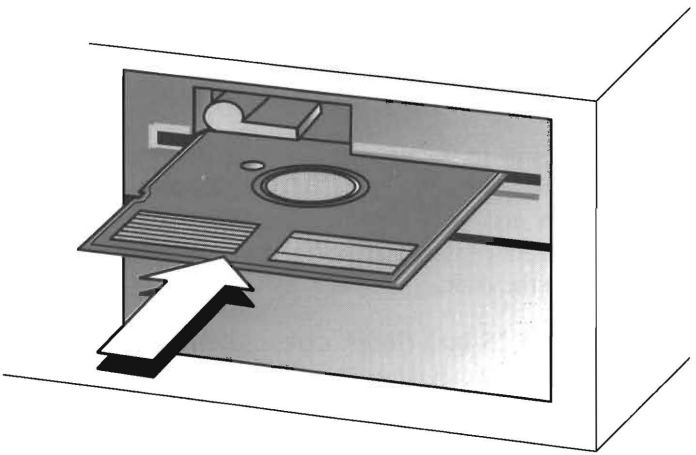
To store information on a write-protected disk, you must first remove the write-protect tab on 5.25-inch disks, or slide the tab back over the write-protect hole on 3.5-inch disks. When you are finished working with a write-protected disk, it is recommended that you replace the write-protect tab.

If a floppy disk does not have a write-protect notch or hole, it is permanently write-protected. Many software manufacturers put their products on such disks to prevent the information on the disks from being changed or deleted accidentally.

Inserting and Removing a Floppy Disk

The following illustration shows you how to insert a 5.25-inch floppy disk into a horizontal disk drive. Some floppy disk drives that use 5.25-inch floppy disks have a lever that you need to push down or to the side after inserting the floppy disk. When you are finished working with the disk, lift up the drive lever and take out the disk.

To insert a 3.5-inch floppy disk, slide it into the disk drive until you hear a click. When you are finished using the disk, press the button on the front of the drive. The disk pops out so that you can remove it easily.



Inserting a 5.25-inch disk into a drive

Preparing Disks to Hold Information

If you have a new disk that has never been used, you must prepare it for storing information. You do this by running a program that *formats* the disk so that MS-DOS can find information on it. When you format a disk, MS-DOS also checks the disk for defects.

Be very careful when formatting disks. Formatting a disk deletes any information that was stored on the disk, and this information may not be recoverable. For more information about formatting disks with MS-DOS, see Chapter 6, "Managing Disks."

Using Software

Software is the set of programs, procedures, and related documentation associated with a computer system. A program is a coded set of instructions that interprets the information you give to the computer with the keyboard or a mouse, and then directs your computer to carry out a task.

Different kinds of software perform different kinds of tasks. Examples of two kinds of software are operating systems, such as MS-DOS, and programs, such as Microsoft Excel and Lotus 1–2–3. There is a wide variety of software programs, including word processors, accounting packages, computer-aided design systems, and games.

The operating system gets your computer running and controls the operation of your computer's activities. It manages the flow, entry, and display of software and data to and from each part of your computer system. To run a program, you first need to run the operating system. The Microsoft disk operating system (MS-DOS) or a computer manufacturer's version of MS-DOS is the operating system most widely used with personal computers.

When you start a program, you see information on your screen, such as menus and commands. While you are using a program, the operating system is managing your computer's activities. It moves instructions and files from one part of your computer system to another as you choose commands and type information.

Looking at MS-DOS

MS-DOS, like other operating systems, manages the flow of information to and from the various parts of your computer system. You work with MS-DOS by typing or choosing commands, which direct your system to perform tasks. MS-DOS includes commands that you can use to perform these tasks:

- Manage files and directories
- Maintain disks
- Configure hardware
- Optimize the use of memory
- Speed up programs
- Customize MS-DOS

There are two ways you can work with MS-DOS—by using MS-DOS Shell or by typing commands at the command prompt.

MS-DOS Command Line

The MS-DOS command line is where you type commands. The command prompt indicates that you are at the command line. The prompt may be a drive letter followed by a backslash (C:\ or A:\, for example) or a backslash and the name of a directory (C:\STATUS, for example).

The letter indicates which disk drive is the current drive. MS-DOS searches the current drive for the information it needs to process the commands you type.

To direct MS-DOS to perform a task, you type a command and then press ENTER. Characters appear to the right of the command prompt as you type. For more information about the MS-DOS command line and basic techniques for typing commands, see Chapter 2, "Command-Line Basics."

MS-DOS Shell

In addition to the command line, you can use MS-DOS Shell to work with many of the MS-DOS commands. MS-DOS Shell offers a visual way of working with MS-DOS. It displays drives, directories, files, and programs available for your use.

The commands in MS-DOS Shell are listed on *menus*; the names of these menus are located across the top of the screen. You use a command in MS-DOS Shell by choosing it from a menu with the keyboard or a mouse. Keep in mind that not all MS-DOS commands can be used in MS-DOS Shell; you must type some commands at the command prompt.

To learn how to use MS-DOS Shell, see Chapter 3, "MS-DOS Shell Basics."

Using Files and Directories

The information your computer uses is stored in *files*. The instructions used to run a program are stored in *program files*, and the information you create by using a program is stored in *data files*.

As you work with a program, MS-DOS processes the information stored in program files and passes it along to your system when it is needed. When you are finished using the program, your data files are stored on a disk.

You assign each file a name so that you can identify its contents. For example, suppose you use a word-processing program to write the notes for a meeting you attended. You might assign the name MINUTES to the data file that contains these notes.

Organizing your files into directories and subdirectories on a disk is like organizing your documents into file folders and then storing them in the drawers of a file cabinet.

MS-DOS has rules for naming files. MS-DOS also includes several commands that you can use to work with files. For information about using MS-DOS to name and work with files, see Chapter 4, "Working with Files."

Organizing Files into Directories

A disk can hold several hundred or even thousands of files, depending on its size. The more files you have, the more difficult it is to keep track of them. To help you keep track of your files, you can use MS-DOS commands to group your files into *directories*. Just as file folders in a file cabinet contain groups of related documents, directories contain groups of related files, such as the minutes and expense reports you create by using a word-processing program. You assign each directory a unique name so that you can identify it.

For example, suppose you had three files: one for meeting minutes, one for monthly expense reports, and one for weekly status reports. You could use the MS-DOS **md** command to create three directories: MEETING, EXPENSE, and STATUS. You could keep your meeting minutes in the MEETING directory, your expense files in the EXPENSE directory, and your status files in the STATUS directory.

Using Subdirectories

When your directories become too large, you can use MS-DOS to create additional directories to further organize your files. A directory within another directory is called a *subdirectory*. For example, within your STATUS directory, you could organize your status reports by month if you create a subdirectory for each month. The subdirectory STATUS\JAN would hold the status reports you wrote in January, STATUS\FEB would hold the ones you wrote in February, and so on.

The organization of directories, subdirectories, and files is called the *directory tree*. When you format a disk, MS-DOS creates one large directory, called the *root directory*, on the disk. All other directories you create branch out from the root directory, as shown in the following illustration:

```
[C:\] tree
Directory PATH listing for Volume CHRIS
Volume Serial Number is 1575-6935
C:.
├──DOS
├──MEETING
├──EXPENSE
└──STATUS
    ├──JAN
    └──FEB
```

With the MS-DOS **dir** command, you can view a list of files and subdirectories in a directory. For example, suppose you use the **dir** command to view the list of files in the C:\STATUS\JAN subdirectory. Your screen would display a list similar to the following:

```
[C:\STATUS\JAN] dir

 Volume in drive C is CHRIS
 Volume Serial Number is 157A-6C23
 Directory of C:\STATUS\JAN

.            <DIR>      11-28-90   2:55p
..           <DIR>      11-28-90   2:55p
WK1STAT  TXT         47 01-03-91   3:10p
WK2STAT  TXT        377 01-03-91   3:10p
WK3STAT  TXT       1236 01-03-91   3:10p
WK4STAT  TXT       2487 01-03-91   3:10p
        6 file(s)       4147 bytes
                     37267456 bytes free
```

For more information about creating, listing, and using directories and subdirectories in MS-DOS, see Chapter 5, "Working with Directories."

Chapter 2

Command-Line Basics

2

MS-DOS indicates the command line by displaying the command prompt, for example:

```
C:\>
```

You type commands at the prompt to specify tasks you want MS-DOS to perform. For example, if you want the MS-DOS version number displayed, do the following:

1. Type **ver**
2. Press ENTER.

   The software version number appears.

Each command contains a set of instructions. For example, when you use the **ver** command, you instruct MS-DOS to display information about the MS-DOS version number. A command can be a word (**time**) or an abbreviation (**dir**). To carry out a command, you type the command and then press ENTER.

Parts of a Command

An MS-DOS command has up to three parts. Every command has a *command name.* Some commands require one or more *parameters* that identify the object you want MS-DOS to act on. Some commands also include one or more *switches,* which modify the action being performed.

The Command Name

The command name, which you type first, states the action you want MS-DOS to carry out. Some commands (such as the **cls** command, which clears your screen) consist only of a command name. Most MS-DOS commands, however, require more than a name.

Parameters

MS-DOS sometimes requires additional information, which you specify in one or more parameters after the command name. A parameter defines the object you want MS-DOS to act on. For example, the **del** (**erase**) command requires a parameter that names the file you want to delete. Suppose, for example, you want to delete a file named NOTES.TXT. Here is what you would type:

```
del notes.txt
```

Some commands require more than one parameter. For example, to rename a file by using the **rename** (**ren**) command, you must include the original name of the file in addition to the new name. The following command changes LETTER.TXT to MEMO.TXT:

```
ren letter.txt memo.txt
```

With some commands, parameters are optional. For example, you can use the **dir** command without a parameter to list files in the directory you are currently using. Or you can include a parameter (a different drive, for example) to list files in a different directory.

Switches

A switch is a forward slash (/) usually followed by a single letter or number. You use switches to modify the way a command performs a task. For example, suppose you want to use the **dir** command to view a listing of a directory that contains a large number of files. When you type the **dir** command by itself, the filenames scroll by so rapidly on the screen that you cannot read them all. If you add the **/p** switch, you can view the list of files one screen at a time.

Some MS-DOS commands do not have any switches, whereas others have several. If a command has more than one switch, you type them one after the other. You can separate switches with a space, but the space is optional.

Typing a Command

The flashing underscore on the command line is the *cursor*. The cursor shows you where to type the command. When you type a character, the cursor moves one space to the right. If you make a mistake, press BACKSPACE to delete a character to the left of the cursor. You can type a command in uppercase or lowercase letters. Unless otherwise specified, you must use a space to separate a command from its parameters.

If you want to retype a command, press ESC. The cursor moves to the beginning of the next line, and you can start over. Anything that you typed before you pressed ESC is ignored.

Shortcuts to Typing a Command

MS-DOS has editing keys that change or repeat a command you've typed. Two of the most commonly used keys are F1 and F3:

F1	Displays the previous command one character at a time
F3	Displays all of the previous command

For information about other editing keys, see Chapter 7, "Advanced Command Techniques."

Suppose you type the following:

```
dir a: #
```

Because there is an extra character at the end of this command, MS-DOS displays an error message. If you press F3, the command reappears. Press BACKSPACE to delete the number sign (#), and then press ENTER to see the directory listing.

Suppose you want to retype the same command, substituting drive B for drive A. By pressing F1 four times, the first part of the command, **dir**, appears. You then type **b**, and press F1 once. The colon (:) appears. If your system has a B drive, press ENTER to see a directory listing. Otherwise, press ESC.

MS-DOS has a Doskey program that retrieves, modifies, and reuses commands.

To install Doskey, type **doskey** at the command prompt. If Doskey has not already been installed, the following message appears when you type **doskey**:

```
DOSKey installed.
```

You can now retrieve and edit commands you type. For example, suppose you type the following three commands:

```
type tuesday
date
time
```

The first command displays the contents of a file named TUESDAY; the second displays the current date; the third displays the current time. All of these commands are stored in your system's temporary memory.

You can use several methods to retrieve these commands when Doskey is installed. The easiest method is to press the UP ARROW key. If you press the UP ARROW key once, the most recent command (**time**) appears at the command prompt. Press the UP ARROW key two more times to view the first command:

```
type tuesday
```

You can press ENTER if you want the command to be carried out again, or you can edit the command. For example, press HOME to move the cursor back to the beginning of the line, and type the **ren** command over the **type** command. Press the DEL key to erase the **e** left from the **type** command. Then press END to move the cursor to the end of the line, press the SPACEBAR, and type **monday** as your new name for the file. The edited command looks like this:

```
ren tuesday monday
```

To carry out the revised command, press ENTER.

For more information about using Doskey, see Chapter 7, "Advanced Command Techniques."

How MS-DOS Responds to a Command

MS-DOS responds to commands in various ways. MS-DOS might display a message indicating that the command has been successfully completed or that you didn't type the command correctly.

When you type some commands, MS-DOS prompts you for more information. For example, if you type the **time** command, MS-DOS displays the following prompt:

```
Current time is: 9:52:18:34a
Enter new time:
```

In response, you specify the new time.

Sometimes MS-DOS prompts you to verify a command. For example, suppose you use the following **del** command with wildcards (described in Chapter 4, "Working with Files") to delete all files in the C:\TMP directory:

```
del c:\tmp\*.*
```

MS-DOS displays the following message:

```
All files in directory will be deleted!
Are you sure (Y/N)?
```

If you don't want to delete all the files, type **n**. If you do want to delete all files, type **y**.

Sometimes MS-DOS displays the results of a command. For example, when you use the **copy** command to instruct MS-DOS to copy a particular file, MS-DOS displays the following information:

```
1 file(s) copied
```

Sometimes you receive an error message indicating that MS-DOS doesn't recognize the command you typed. If you misspelled the command, type it again and press ENTER. If the command exists and you typed it correctly, you may have to change directories or specify the directory where the program file is located. For more information, see Chapter 5, "Working with Directories."

Stopping or Canceling a Command

You can temporarily stop the output of a command by pressing CTRL+S or PAUSE. Press any key except PAUSE to restart the output of the command. You can stop and restart the output of a command as many times as you want.

If you want to stop MS-DOS from completing a command, press CTRL+BREAK or CTRL+C. Your command is canceled, and the command prompt appears.

NOTE Any action MS-DOS carries out before you press CTRL+BREAK or CTRL+C cannot be undone.

Designating a Disk Drive

The *current drive* appears as the first letter of the command prompt. On most systems, if the letter is A or B, one of the floppy disk drives is the current drive. If the letter is C, the hard disk drive is current. Some systems have additional drives as well.

If the files or directories you want to work with are on a disk in the current drive, you do not need to specify the drive. If the files or directories are not located on the current drive, you can either specify the drive as part of a command or change the current drive.

To change the current drive, type the letter of the drive followed by a colon. For example, to change the current drive from C to A, type the following:

```
a:
```

To specify another drive, include the drive letter with the command. For example, suppose the current drive is C. To view a list of files on a disk in drive A, type **a** followed by a colon as a parameter in the **dir** command:

```
dir a:
```

Internal and External Commands

MS-DOS loads some commands into memory when you start your system. These *internal* commands are in a file called COMMAND.COM. Some internal commands are **dir**, **del**, **date**, and **time**.

MS-DOS stores *external* commands in files on a disk and transfers them from disk to memory as you use them. Two external commands are **chkdsk** and **format**. If you set up MS-DOS version 5.0 on a system with a hard disk and you use the default directory, the external-command files are placed in the \DOS directory. If you move the files, indicate their new location by using the **path** command. For information about this command, see Chapter 5, "Working with Directories" or Chapter 14, "Commands."

To determine whether a specific command is internal or external, see Chapter 14, "Commands."

Getting Help with a Command

Online Help is available for all MS-DOS commands. Help describes the purpose of the command you specify and provides a summary of its parameters and switches. To use Help, type the command name followed by the **/?** switch, or type **help** followed by the command name.

For example, to use Help for the **del** command, you could type this:

```
del /?
```

Or you could type the following, with the same result:

```
help del
```

MS-DOS displays the following Help on the **del** command:

```
Deletes one or more files.

DEL pathname [/P]
ERASE pathname [/P]

  pathname  File(s) to delete. Specify multiple files by using
wildcards.
  /P        Prompt before deleting each file.
```

If you type **help** without a command name, MS-DOS displays a list of all MS-DOS commands and their purpose.

Chapter 3

MS-DOS Shell Basics

3

Through the use of color and graphics, MS-DOS Shell offers a visual way of working with MS-DOS. Information is set up in different areas on your screen, making it easy to find. For example, when you first run MS-DOS Shell, the following information is displayed on your screen:

- The disk drives available on your system
- The directory structure or tree for the current disk drive
- A list of files in the current directory
- A list of programs that you can run

You can use MS-DOS Shell to perform many of the same file-management and disk-maintenance tasks that you perform from the command line. For example, you can use the commands on the File menu to create directories, copy files, and view the contents of a file. You can use the Disk Utilities group to perform disk-maintenance tasks, such as formatting and copying disks. You can also use MS-DOS Shell to organize and start programs, and to switch between them.

Starting MS-DOS Shell

If MS-DOS Shell was set up to run whenever you start your system, MS-DOS Shell appears on your screen when you start MS-DOS version 5.0. If the command prompt appears instead, you can start MS-DOS Shell from there.

▶ To start MS-DOS Shell from a hard disk:

- At the command prompt, type **dosshell**, and then press ENTER.

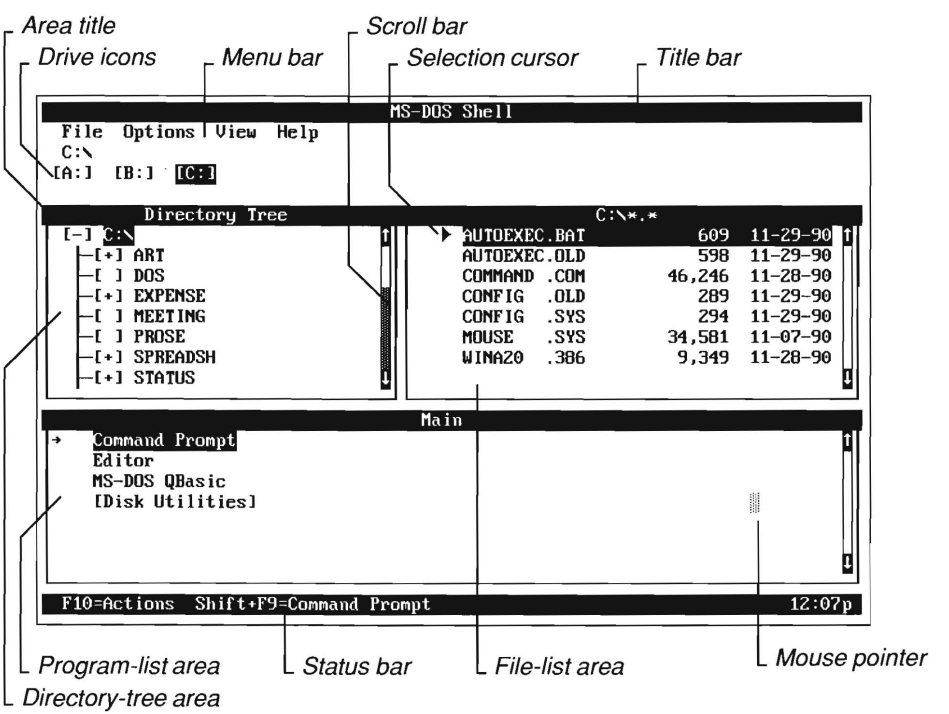
▶ To start MS-DOS Shell from a floppy disk:

1. Start MS-DOS from your startup disk.
2. When the command prompt appears, insert in drive A the floppy disk that contains the MS-DOS Shell program file.
3. Type **dosshell**, and then press ENTER.

WARNING If you start a terminate-and-stay-resident (TSR) program before starting MS-DOS Shell, do not quit the TSR program by using its exit procedure while you are still in MS-DOS Shell. Instead, you must first leave MS-DOS Shell and then quit the TSR program.

The MS-DOS Shell Window

When you first start MS-DOS Shell, your screen is divided into different areas, as follows:



The following elements are part of the MS-DOS Shell window:

- The *title bar* displays MS-DOS Shell.
- The *menu bar* lists the names of the available menus. When you select a menu, it displays a list of commands you can choose from.
- There is an *area* for each of the following: drive icons, a directory tree, a program list, a file list, and an Active Task List. (The Active Task List is illustrated and described in more detail later in this section.)
- The *selection cursor* shows you which item has been selected.
- *Scroll bars* move part of a list into view when the entire list won't fit in an area.
- The *status bar* displays shortcut keys, messages from MS-DOS Shell, and the current time.
- The *mouse pointer* appears if you have a mouse installed.

Areas in the MS-DOS Shell Window

The MS-DOS Shell window is divided into areas. Each area displays different information. By using the commands on the View menu, you can choose which areas you want displayed. When you start MS-DOS Shell, the following areas appear on your screen by default: drive icons, the directory tree for the current drive, a list of files in the current directory, and the program list displaying the Main group. You can also display the Active Task List, a list of programs that you have started.

Drive Icons

The drive icons represent each available disk drive on your computer. By selecting a drive icon, you make that drive the current drive.

The Directory Tree

The directory-tree area shows the structure of the directories on the current disk drive. When you select a drive icon, the information in the directory-tree area changes to reflect the directory structure on that drive.

Whenever the directory-tree area, file-list area, or a drive icon is selected, the Tree menu appears in the menu bar. By using the commands on the Tree menu, you can choose the level of the directory structure that you want to view. For example, you could choose to view only the directories under the root directory, or all the directories and subdirectories on the current drive.

The File List

The area next to the directory tree shows a list of files in the current directory. The current directory is the one that is selected in the directory tree. When you select another directory from the directory tree, the title bar of the file-list area changes to reflect the path of the selected directory, and the files in the selected directory appear in the file-list area.

By using commands on the Options menu, you can specify how you want the files in the file list displayed. For example, you can use the File Display Options command to choose which files you want displayed and the order in which you want them to appear.

The Program List

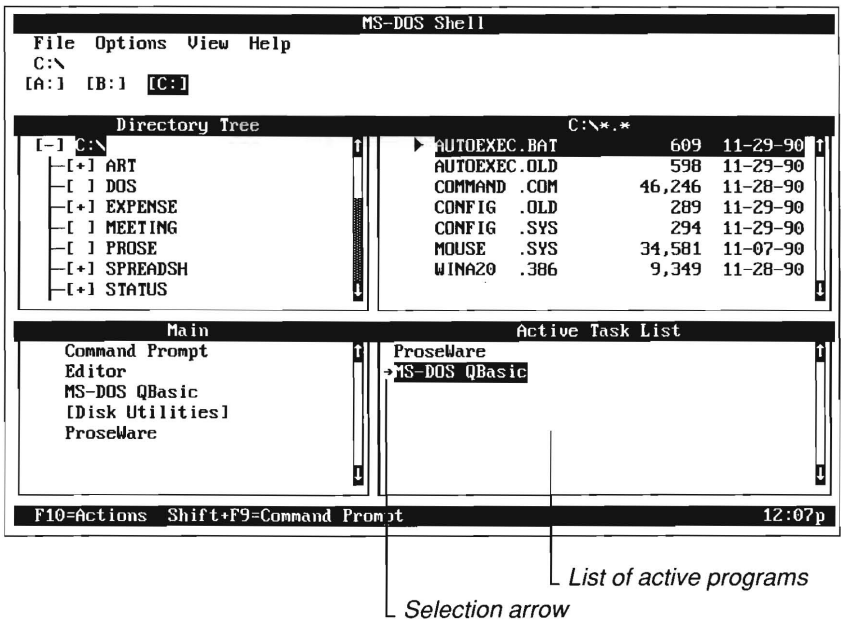
If Program/File Lists is selected on the View menu, you will see the program list below the directory tree and the file list. By default, the Main group is displayed in the program list. The Main group includes two programs that you can start directly from MS-DOS Shell: Editor starts MS-DOS Editor, a text editor for creating text files (such as batch programs) or for modifying existing files (such as your CONFIG.SYS file); MS-DOS QBasic is an interpreter that you can use to write Basic programs.

The Main group also includes Command Prompt. When you choose Command Prompt, you leave MS-DOS Shell and go to the MS-DOS command line. For more information about using the command line, see Chapter 2, "Command-Line Basics."

The Disk Utilities group also is in the Main group. The Disk Utilities group includes several programs you can use to maintain your disks. To view these programs, open the Disk Utilities group. For more information about starting programs and opening groups, see "Working with Programs" later in this chapter.

The Active Task List

The Active Task List area appears in the MS-DOS Shell window when you enable Task Swapper.



To enable Task Swapper, you use the Enable Task Swapper command on the Options menu. Any programs you start after enabling Task Swapper appear in the Active Task List. You switch to a program by selecting it from the list. For more information about enabling Task Swapper and switching between programs, see "Switching Between Programs" later in this chapter.

Selecting an Area

Before you can work in an area in MS-DOS Shell, you need to select it. If you have a color monitor, the title bar of the selected area changes color. If you have a monochrome monitor, the selected area contains a small arrow to the left of an item in the area.

To select an area:

Mouse

- Click the area you want to select.

  The selection cursor or arrow marks the currently selected drive, directory, file, group, or program.

Keyboard

- Press TAB to move among areas until you reach the area you want to select.

  To reverse direction, press SHIFT+TAB.

  After you move to an area, the selection cursor or arrow marks the currently selected drive, directory, file, group, or program.

Working with Menus

Menus are lists of commands. Menu names appear in the menu bar along the top of the MS-DOS Shell window.

Selecting and Canceling a Menu

MS-DOS Shell has the following menus to choose from: File, Options, View, Tree, and Help.

To select a menu:

Mouse

- Point to the name of the menu on the menu bar, and click the name to open the menu. (You can drag the selection cursor down the menu if you want to move to a command immediately.)

Keyboard

1. To select the menu bar, press ALT or F10.
2. To select and open the menu you want, press the LEFT ARROW or RIGHT ARROW key, and then press ENTER.

   Or type the highlighted letter of the menu name.

   After you select a menu, you can select another by using the RIGHT ARROW or LEFT ARROW key.

▶ **To cancel a menu:**

Mouse ■ Click the menu name or anywhere outside the menu.

Keyboard ■ To cancel the menu, press ESC.

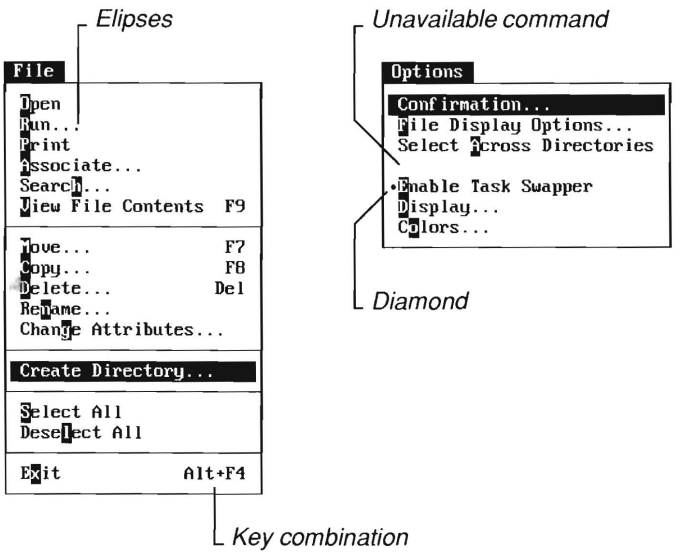
Or if you want to select another menu, you can move to it by using the RIGHT ARROW or LEFT ARROW key.

Choosing Commands

You use the commands on menus to specify various tasks you want MS-DOS to carry out. For example, you use some commands to assign characteristics to programs. Some commands perform tasks directly; others require you to make additional choices.

The following conventions are used in MS-DOS Shell:

Convention	Description
Dimmed (or not visible) command name	The command is unavailable at this time. You might need to select an item before you can use this command; or it might not be available for the task you are currently performing. If you are using the Monochrome–2 Colors or Reverse color scheme, these commands are not visible.
Ellipsis (...) after a command name	A dialog box appears when the command is chosen, asking for information needed to carry out the command.
Diamond (◆) next to a command name	The command is active. This convention is used for commands that switch between one state and another.
Key combination after a command name	A key combination is a *shortcut* for the command. Use the key combination to choose the menu command without first opening the menu.



In the File menu just shown, note that some commands have keynames or key combinations next to their names and several have ellipses (...). In the Options menu just shown, note that one command has a diamond (◆) next to it, and one command is not visible.

▶ To choose a command from a selected menu:

Mouse

- Click the name of the command.

Keyboard

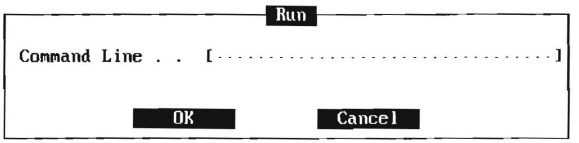
- Type the highlighted letter in the command name.

  Or use the UP ARROW or DOWN ARROW key to move the selection cursor to the command you want, and then press ENTER.

Working with Dialog Boxes

MS-DOS Shell uses *dialog boxes* to request information it needs to carry out a command.

When you choose a command that is followed by an ellipsis, a dialog box appears. For example, if you choose Run from the File menu, MS-DOS Shell displays the Run dialog box. In the dialog box, you supply the path and filename of the program you want to run. The Run dialog box is shown here:



Most dialog boxes contain check boxes, list boxes, option buttons, text boxes, or a combination of these. Each one provides MS-DOS with information necessary to carry out a task or to configure the MS-DOS Shell according to your specifications. After you supply the information, you choose a command button to carry out the command. To close a dialog box without carrying out the command, press ESC or choose the Cancel button.

The following list shows the types of options found in a dialog box:

Check box	Select as many of the boxes as you like. A selected check box contains an X.
List box	Select one from a list of choices.
Option buttons	Select one from a list of options. You can only select one option at a time. The selected option button contains a dot.
Text box	Type text in the box that has a cursor in it.

Moving Within Dialog Boxes

You may need to move within a dialog box to provide MS-DOS with the information it needs.

▶ To move within a dialog box:

Mouse

- Click the area that you want to move to.

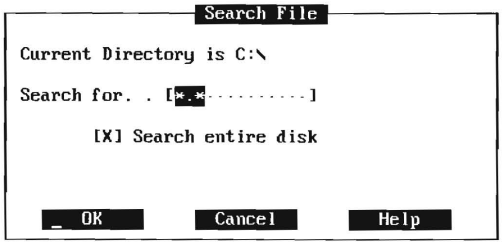
Keyboard

1. To move forward (generally from left to right or from top to bottom), press TAB; to move in the opposite direction, press SHIFT+TAB.
2. Within a list box, check box, or group of option buttons, use the arrow keys to move among the options.

Choosing a Command Button

A *command button* carries out an immediate action. In the Search File dialog box, for example, the OK button carries out the search command, and the Cancel button

cancels the search command. Some dialog boxes have a button labeled Advanced, which, when chosen, opens a dialog box used for advanced features. Sometimes there is a Help command button that you can use to get additional information about the dialog box. The Help button in the Search File dialog box is an example. For more information about Help, see "Getting Help" later in this chapter.



▶ To choose a command button:

Mouse

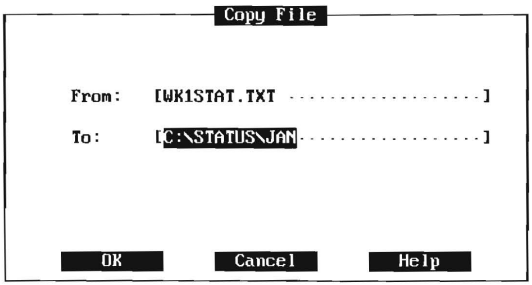
- Click the command button.

Keyboard

1. To move to the command button you want, press TAB.
   The currently selected command contains an underscore.
2. To carry out the command, press the SPACEBAR or ENTER.

Typing Text in a Text Box

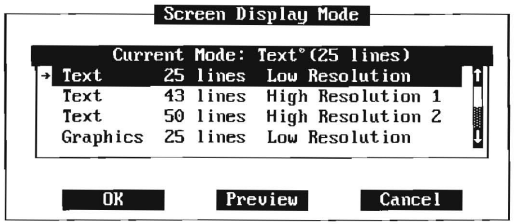
Sometimes you are required to type text in a box within a dialog box, as in the following Copy File dialog box:



The selection cursor indicates where you can start typing. If the box already contains text, any text you type replaces it. However, you can also use the LEFT ARROW and RIGHT ARROW keys to place the cursor anywhere in the text, and insert or delete text at that point.

Selecting an Item from a List Box

Some dialog boxes display a list box. For example, if you choose the Display command from the Options menu, the following Screen Display Mode dialog box appears. If there are more choices than can fit in the dialog box at one time, you can use the scroll bars or the UP ARROW and DOWN ARROW keys to move through the list. For information about scrolling, see "Using Scroll Bars" later in this chapter.



▶ To select an item from a list box:

Mouse

1. Click the scroll arrows until the item you want appears in the box.
2. Click the item you want, and then choose a command button.

   Or to choose the item and complete the command, double-click the item.

Keyboard

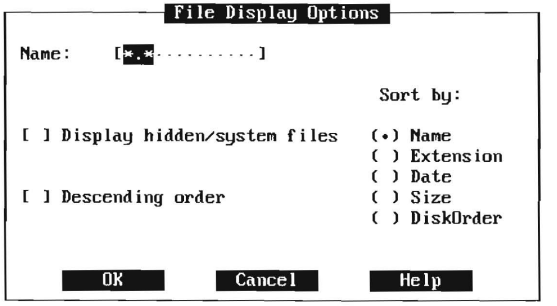
1. Use the UP ARROW or DOWN ARROW key to scroll to the item you want.

   Or type the first letter of the item you want. The selection cursor moves to the first item that starts with that letter. If an item is preceded by a number, you can type the number to move the cursor to that item.
2. To choose the item and complete the command, press ENTER.

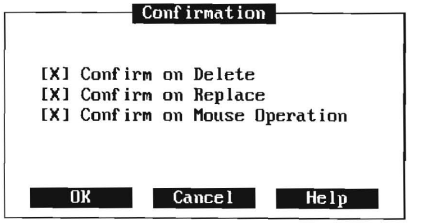
To change your selection, select a different item before choosing the command button.

Selecting an Option Button or a Check Box

If a dialog box contains a list of option buttons, you can select only one of them. The selected item contains a dot. In the following example, the option button Name is selected.



If a dialog box contains a list of check boxes, you can select more than one of them at a time. The selected boxes contain an X. In the following dialog box, all three check boxes are selected.



▶ To select or cancel an option button:

Mouse ■ Click the option button you want. To cancel a selection, click a different button.

Keyboard

1. To move to the area you want, press TAB.
2. Use the arrow keys to select an option button. A dot appears when the option button is selected. To cancel a selection, select a different button.

To select or cancel a check box:

Mouse

- Click the check box you want. To cancel a selection, click it again.

Keyboard

1. To move to the area you want, press TAB.
2. Use the arrow keys to move to the check box you want to select.
3. Press the SPACEBAR to select the box. To cancel the selection, press the SPACEBAR again.

Closing a Dialog Box

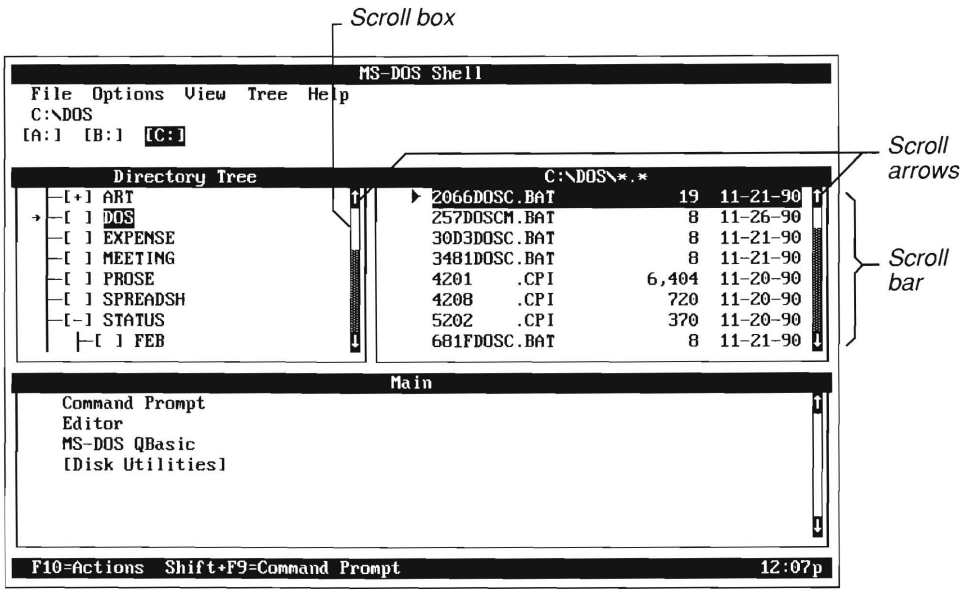
When you choose the appropriate command button (or press ENTER), the dialog box closes and the command takes effect.

To close a dialog box without completing a command:

- Choose the Cancel button, or press ESC.

Using Scroll Bars

Some areas of the MS-DOS Shell window and some dialog boxes contain scroll bars. If there is more information than can fit in an area or list, a *scroll box* appears within the scroll bar. The size of the scroll box tells you how much of the available information you can currently see. A small scroll box tells you that you are seeing only a small portion of what is available. A large scroll box tells you that you are seeing most of what is available. When there is no scroll box, there is no additional information to be seen. You can drag the scroll box up or down to scroll through all the information in a list or area. You can also use the *scroll arrows* to scroll through the information.



To scroll through information displayed in a selected area or dialog box:

Mouse

- Drag the scroll box up or down, until the area you want to work with comes into view.

  You can also use the following techniques to scroll:

To scroll one line	Click one of the scroll arrows.
To scroll continuously	Point to one of the scroll arrows and hold down the mouse button until the information you want comes into view.

Keyboard

- Press the arrow key that points in the direction you want to scroll.

  You can also use the following keys to scroll:

PAGE UP or PAGE DOWN	Scrolls up or down one window
HOME or CTRL+HOME	Scrolls to the beginning of the list
END or CTRL+END	Scrolls to the end of the list

Changing Views

When you first run MS-DOS Shell, directories, filenames, and *program groups* are displayed. A program group is a collection of programs that can have information associated with them, such as startup commands or passwords. Programs listed in a program group are referred to as *program items*. For more information about program groups and items, see "Working with Programs" later in this chapter.

You can view your directories, files, and programs in several ways. You can configure MS-DOS Shell to show:

- Directories and files on a disk.
- Directories and files on two disks.
- Files in a selected directory and information about the files.
- Directories and files on a disk, in addition to program groups and items. (MS-DOS Shell presents information this way by default.)
- Program groups and items only.

▶ To view lists of directories and files on a single disk:

1. Select the drive that contains the disk you want to view.
2. From the View menu, choose Single File List.

   MS-DOS Shell displays a directory tree in the left part of the window. The directory tree shows you the overall organization of directories and subdirectories on the selected disk drive. Directories branch out from the *root directory*. The root directory on a hard disk is generally C:\. In the right part of the window, MS-DOS Shell displays a list of files in the currently selected directory.

▶ To view lists of directories and files on two disks:

1. From the View menu, choose Dual File Lists.

   The window splits into two areas. The directory tree of the disk in the currently selected drive and the filenames in the currently selected directory are displayed in both areas. In other words, the two areas contain the same information at this point.

2. From one of the two groups of drive icons, select the drive that contains the second disk you want to view.

   In one area, MS-DOS Shell displays a list of directories and files on the second disk, and in the other area, it displays a list of directories and files on the first disk you selected.

▶ To view a list of all files on a drive:

1. Select the drive that contains the filenames you want to view.
2. From the View menu, choose All Files.

   MS-DOS Shell displays a list of all filenames on the drive and supplies information about the currently selected file. For more information about files, see Chapter 4, "Working with Files."

▶ To view both programs and filenames:

1. Select the drive containing the filenames you want to view.
2. From the View menu, choose Program/File Lists.

▶ To view only program groups and items:

- From the View menu, choose Program List.

Working with Files and Directories

MS-DOS Shell is an efficient tool for organizing and working with files and directories. It displays a list of your directories, subdirectories, and files.

This section introduces the basic procedures for working with files. For more information about using MS-DOS Shell to work with files and directories, see Part 2, "Working with MS-DOS."

Selecting a Disk Drive

When you start MS-DOS Shell, it displays a list of the directories and files on the current drive. To work with directories and files on a different drive, you need to change the current drive. MS-DOS Shell displays the message "Reading Disk Information" while it is reading the contents of the disk.

▶ To change the current drive:

Mouse

- Click the appropriate drive icon.

Keyboard

1. Press TAB until one of the drive icons is selected.
2. Use the LEFT ARROW or RIGHT ARROW key to move the selection cursor to the drive icon you want.

   Or press and hold down the CTRL key while you type the letter of the drive.
3. Press the SPACEBAR if you are selecting the drive for the first time.

▶ To update disk information:

Mouse

- Double-click the appropriate drive icon.

  Or from the View menu, choose the Refresh command.

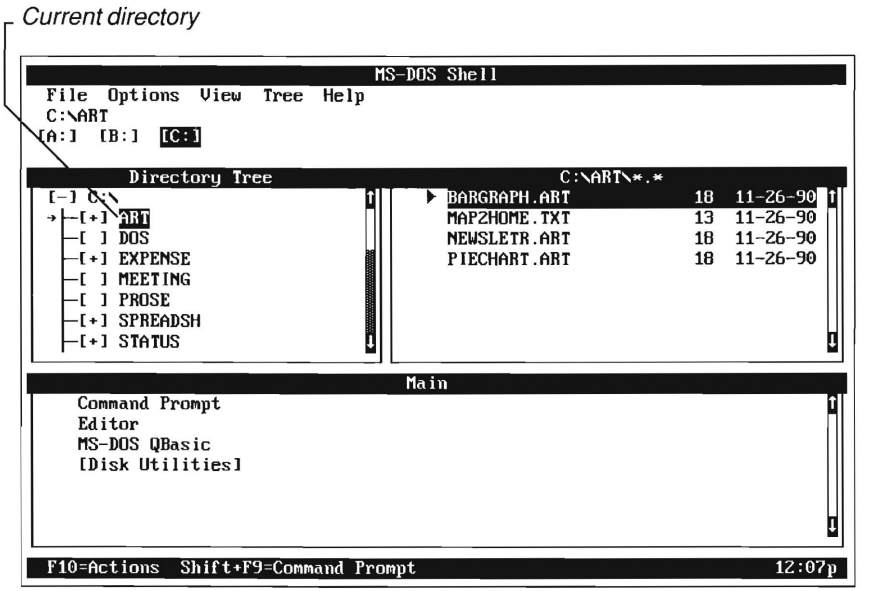
Keyboard

1. Select the drive containing the disk you want MS-DOS Shell to read again.
2. Press ENTER.

   Or press F5.

Changing Directories

The current directory appears highlighted in the directory tree. Only one directory can be current at a time.



To change the current directory:

Mouse

- Click the name of the directory you want to make current.

Keyboard

- Use the following keys to select a directory:

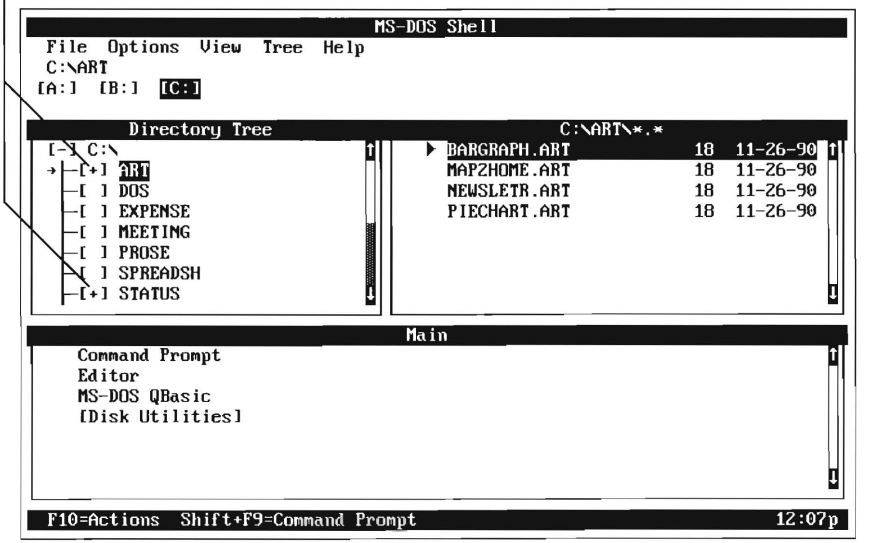
PAGE UP or PAGE DOWN	Scrolls up or down one window
UP ARROW or DOWN ARROW	Scrolls up or down, one directory at a time
HOME	Selects the root directory
END	Selects the last directory in the list
The first letter of the directory name	Selects a directory name beginning with the letter you typed

Expanding a Directory

When you start MS-DOS Shell, first-level directories on the current disk are displayed. You can use commands in the Tree menu to control the amount of directory information that is displayed. The Tree menu appears when you have selected a directory tree, file list, or drive icon.

In the following illustration, two directory names have a plus sign (+). A plus sign indicates that the directory has one or more subdirectories.

Expandable directories



You can click the plus sign to view subdirectories. This is called *expanding* a directory. You can expand an entire branch, a single level of the branch, or all branches. When you expand a directory, the plus sign changes to a minus sign (–).

To expand one level of a directory:

Mouse

- Click the plus sign (+) next to the name of the directory you want to expand.

Keyboard

1. Use the UP ARROW or DOWN ARROW key to select the directory you want to expand.
2. From the Tree menu, choose Expand One Level.
   Or press the PLUS (+) key.

To view all branches of a directory:

1. Select the directory you want to expand.
2. From the Tree menu, choose Expand Branch.
   Or press the ASTERISK (*) key.

To view all directory levels in a directory tree:

- From the Tree menu, choose Expand All.
  Or press CTRL+ASTERISK (*).

Collapsing a Directory

When you finish viewing subdirectories, you can *collapse* the directory so only the first-level directory name is displayed. A minus sign (–) beside a directory name indicates that you can collapse the directory tree. All directories except the root directory are collapsed when you start MS-DOS Shell.

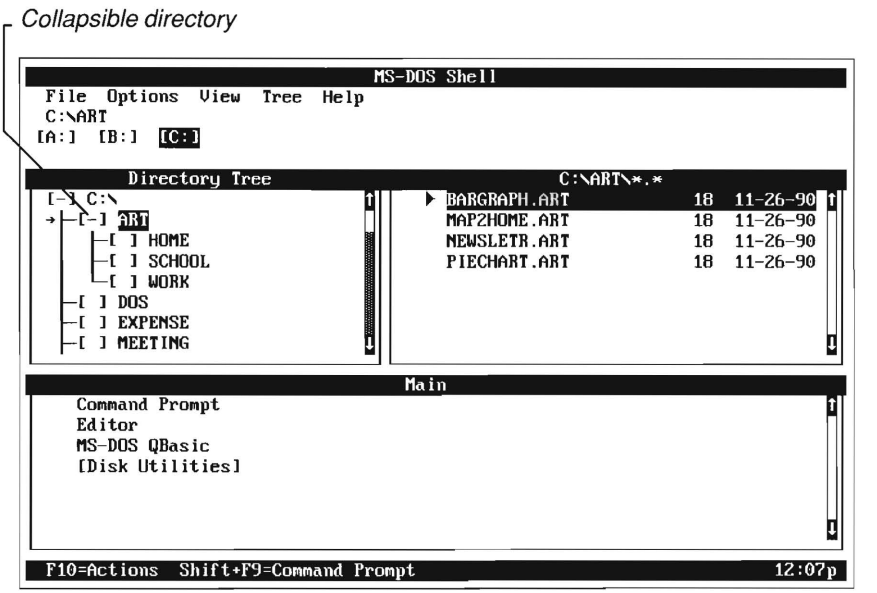
To collapse a directory:

Mouse

- Click the minus sign (–) next to the name of the directory you want to collapse.

Keyboard

1. Use the UP ARROW or DOWN ARROW key to select the directory you want to collapse.
2. From the Tree menu, choose Collapse Branch.
   Or press the MINUS (–) key.



Updating a Directory

If you leave MS-DOS Shell temporarily and make changes to files in a directory, the changes will not be displayed in the MS-DOS Shell file-list area until you update the directory.

▶ To update a directory:

1. Select the directory that you want to update.
2. Press CTRL+F5.

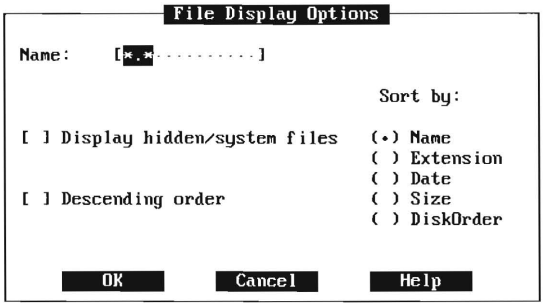
For information about leaving MS-DOS Shell, see "Leaving MS-DOS Shell" later in this chapter.

Changing How File Information Is Displayed

You can control how MS-DOS Shell displays information about files by using the File Display Options command on the Options menu. When you first run MS-DOS Shell, all files in the current directory (except hidden and system files) are listed. Files are listed alphabetically.

▶ To view a specific file:

1. From the Options menu, choose File Display Options.
   The File Display Options dialog box appears.



2. In the Name box, specify the filename or extension of the type of file you want to view. Use MS-DOS file-naming conventions, including wildcards. For information about specifying filenames, see Chapter 4, "Working with Files."
3. Choose the OK button.

▶ To view hidden and system files:

1. From the Options menu, choose File Display Options.
   The File Display Options dialog box appears.
2. Select Display Hidden/System Files.
3. Choose the OK button.

▶ To change the way a file list is sorted:

1. From the Options menu, choose File Display Options.
   The File Display Options dialog box appears.
2. Select one of the Sort By options.

You can select one of the following options:

Name	Sorts by filename, in alphabetic order
Extension	Sorts by extension, then by filename prefix, in alphabetic order
Date	Sorts by the date the file was last modified, with the most recently modified file listed last
Size	Sorts by file size, from the smallest to largest
DiskOrder	Sorts by the order that files are put on the disk

3. Choose the OK button.

▶ To reverse the order of sorting:

1. From the Options menu, choose File Display Options.
   The File Display Options dialog box appears.
2. Select Descending Order.
3. Choose the OK button.

Selecting Files

Before you can work with a file, you must select it.

▶ To select a file in a directory:

Mouse

- Click the name of the file.

Keyboard

- Use the following keys to select files:

UP ARROW or DOWN ARROW	Selects a file listed above or below the current file
HOME	Selects the first file in the list
END	Selects the last file in the list
The first letter of a filename	Selects the first file that starts with that letter

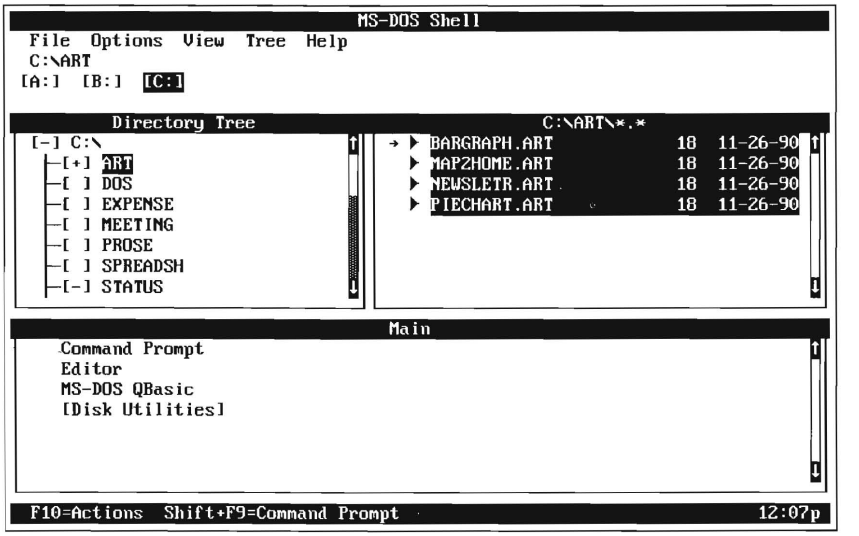
Extending a Selection

Within a directory, you can select more than one file at a time. Selecting more than one file is called *extending* a selection. You can select several files and copy them to another directory. The files can be in sequence or scattered throughout the file list.

▶ To select two or more files that are in sequence:

Mouse

1. Click the name of the first file you want to select.
2. Press and hold down SHIFT while you click the name of the last file you want to select.



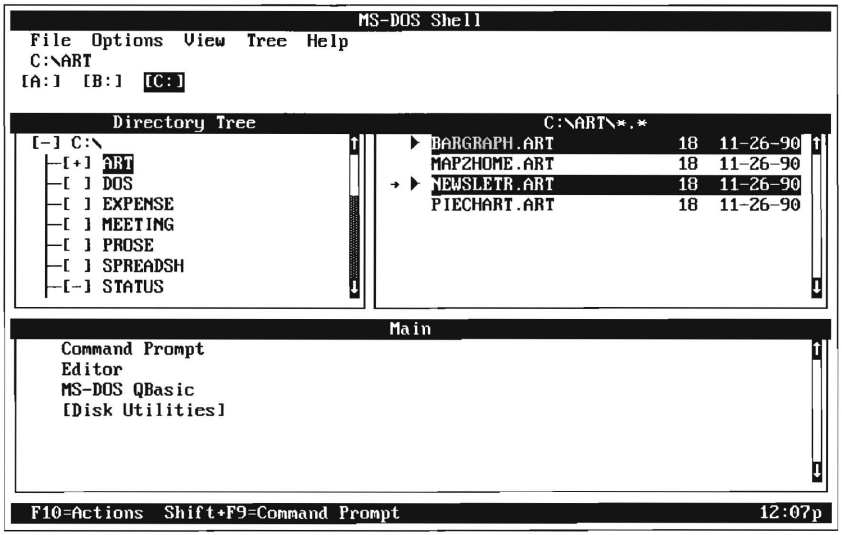
Keyboard

1. Use the UP ARROW or DOWN ARROW key to move to the name of the first file you want to select.
2. Press and hold down SHIFT while you use the arrow keys to select the remaining filenames in the group.

▶ To select two or more files that are not in sequence:

Mouse

- Press and hold down CTRL while you click the name of each file you want to select.



Keyboard

1. Select the file list.
2. Select the name of the first file in the group.
3. Press SHIFT+F8.

   The word "Add" appears in the status bar.
4. Use the UP ARROW or DOWN ARROW key to move to the name of the second file you want to select.
5. Press the SPACEBAR to select the name of the second file you want to select.
6. Repeat steps 4 and 5 for each file you want to select.
7. Press SHIFT+F8 when you have finished extending your selection.

   "Add" disappears from the status bar.

▶ To select more than one group of files:

Mouse

1. To select the first group of files, click the name of the first file, and then press and hold down SHIFT while you click the last filename in the group.
2. To select the next group of files, press and hold down CTRL while you click the first filename in the next group. Then press and hold down CTRL+SHIFT while you click the last filename in the group.

Keyboard

1. To select the first group of files, press and hold down SHIFT while you use the UP ARROW or DOWN ARROW key to move from the first to the last filename in the group.
2. Press SHIFT+F8.

   The word "Add" appears in the status bar.
3. Press the arrow keys to move to the first filename in the next group.
4. Press the SPACEBAR to select the name of the first file.
5. Press and hold down SHIFT while you use the arrow keys to select the names of the remaining files in the group.
6. Press SHIFT+F8 when you have finished extending your selection.

   "Add" disappears from the status bar.

▶ To select files in different directories:

1. From the Options menu, choose Select Across Directories.

   A diamond (◆) appears next to the command name.
2. Follow the instructions earlier in this chapter describing how to select two or more files that are not in sequence.

WARNING Because MS-DOS displays only the names of files in the current directory, you won't be able to see all of the filenames you have selected. Use caution when Select Across Directories is in effect, because you may inadvertently delete identically named files in different directories.

▶ To select all files:

- From the File menu, choose Select All.

  Or press CTRL+SLASH (/).

Canceling a Selection

You can cancel one or all of the selections you have made.

▶ To cancel a selection:

- Select a different item.

▶ To cancel a single selection from an extended selection:

Mouse

- Press and hold down CTRL while you click the selected item.

Keyboard

1. Press SHIFT+F8.
   The word "Add" appears in the status bar.
2. Use the UP ARROW or DOWN ARROW key to move to the selected item.
3. Press the SPACEBAR.
4. Press SHIFT+F8 to cancel the selection mode.
   "Add" disappears from the status bar.

▶ To cancel all selections except the first selected file:

- From the File menu, choose Deselect All.
  Or press CTRL+BACKSLASH (\).

Working with Programs

One of the major features of MS-DOS Shell is its ability to run two or more programs at one time. When you run programs at the same time, they use your system resources in different ways, depending on how your system is set up. For information about how you can set up your system to run programs most efficiently, see Chapter 12, "Optimizing Your System."

MS-DOS Shell is also a tool for organizing your programs into groups of program items. When you add a program to a group, you can specify information that the program will use each time you start the program. For example, the program file EDIT.COM has a name (MS-DOS Editor) and a startup command associated with it. For information about creating program items, see Chapter 8, "Customizing MS-DOS Shell."

Viewing a Program Group

The Main group is the program group displayed in the program list the first time you start MS-DOS Shell. For information about the program list, see "Areas in the MS-DOS Shell Window" earlier in this chapter.

The Main group contains the following program items: Command Prompt, Editor, and MS-DOS QBasic. The Main group also contains a program group called Disk Utilities.

For you to view the items in a group, the group must be open. For example, to view the items in the Disk Utilities group, you must first open it.

▶ To open a program group:

Mouse

- Double-click the group name.

Keyboard

- Use the UP ARROW or DOWN ARROW key to select the group you want to open, and then press ENTER.

You close one group by opening another group.

Starting a Program

There are four ways to start a program:

- From a program group, choose a program item.
- From a file list, choose a program file or a file associated with that program.
- From the File menu, choose the Run command and type the name of the program file.
- From the Main group, choose Command Prompt and type the name of the program file. For information about working at the command prompt, see Chapter 2, "Command-Line Basics."

Starting a Program from a Program Group

If a program is listed in a group displayed in the program list, the easiest way to run the program is to choose it from the list.

▶ To start a program from a program group:

Mouse

1. Open the group that contains the program you want to start.
2. Double-click the name of the program.

Keyboard

1. Open the group that contains the program you want to start.
2. Use the UP ARROW or DOWN ARROW key to select the program.
3. From the File menu, choose Open.
   Or press ENTER.

Starting a Program from the File List

You can start a program from the file list by choosing the program file from the file list. Program files have a .COM, .EXE, or .BAT extension.

▶ To start a program from the file list:

Mouse

1. Select the directory that contains the program you want to start.
2. In the file list, double-click the name of the program file.

Keyboard

1. Press TAB to select the directory-tree area.
2. Use the UP ARROW or DOWN ARROW key to select the directory that contains the name of the program you want to start.
3. Press TAB to select the file-list area.
4. Use the UP ARROW or DOWN ARROW key to select the name of the program file.
5. From the File menu, choose Open.
   Or press ENTER.

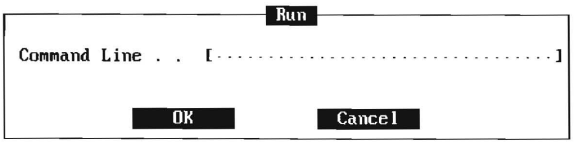
Starting a Program by Using the Run Command

You can also start a program by using the Run command. Unless the file is in the current directory, you must know the directory location and name of the file before you can start the program. For example, if the program is Microsoft Word, the file that starts it is WORD.EXE. This file typically is in the C:\WORD directory. You would type **c:\word\word.exe** in the Run dialog box.

▶ To start a program by using the Run command:

1. From the File menu, choose Run.

   The Run dialog box appears.



2. Type the path and name of the program file.
3. Choose the OK button.

Switching Between Programs

You can run more than one program at a time and easily switch between them by enabling Task Swapper.

▶ To enable Task Swapper:

- From the Options menu, choose Enable Task Swapper.

  A diamond (◆) appears next to the command name, and the Active Task List appears.

The Active Task List displays the names of programs that you start after enabling Task Swapper. Once you quit a program, its name no longer appears on the Active Task List.

CAUTION Some mainframe terminal-emulation programs do not run correctly with Task Swapper. Using them with Task Swapper may cause your system to be disconnected from your mainframe session and cause data to be lost.

▶ To run multiple programs:

1. Start the the first program by double-clicking its program file in the file list or program list.

   Or press the UP ARROW or DOWN ARROW key to select the program name, and then press ENTER.

The program appears on your screen.

2. Press CTRL+ESC to return to MS-DOS Shell.

   The name of the program you have just started appears on the Active Task List.

3. Start another program.

   The second program appears on your screen even though you have not quit the first one.

▶ To add a program to the Active Task List:

1. From the file list or program list, select the program file you want to add to the Active Task List.
2. Press and hold down SHIFT, and double-click the name of the program.

   Or press SHIFT+ENTER.

▶ To switch to another program from MS-DOS Shell:

- Double-click the program's name on the Active Task List.

  Or press the arrow keys to select the program you want, and then press ENTER.

▶ To cycle through programs in the Active Task List from MS-DOS Shell:

- Press and hold down ALT while pressing TAB. The first program in the Active Task List appears on your screen. If you want to switch to a program further down the Active Task List, do not release ALT—simply continue pressing TAB. After the first program appears, only the names of subsequent programs appear at the top of your screen. To select a program, release ALT.

- Press and hold down ALT while pressing TAB. The name of the next program in the Active Task List appears at the top of your screen. To switch to another program, do not release ALT—simply continue pressing TAB until the name of the program you want appears. To select the program, release ALT.

▶ To switch to MS-DOS Shell from any program:

- Press CTRL+ESC.

- Or hold down ALT while pressing TAB repeatedly until the words "MS-DOS Shell" appear at the top of your screen. To return to MS-DOS Shell, release ALT.

Quitting a Program

If Task Swapper is enabled, you can have more than one program running at a time. The programs you have started are listed in the Active Task List. To remove a program from the Active Task List, you must quit the program.

▶ To quit a program listed in the Active Task List:

1. From MS-DOS Shell, switch to the program you want to quit.
2. Use the program's exit command to quit the program.

   (If you are running Command Prompt, type **exit** at the prompt and press ENTER. MS-DOS returns you to MS-DOS Shell.)

▶ To quit a program that locks up:

1. Switch to MS-DOS Shell.
2. From the Active Task List, select the program you want to quit.
3. From the File menu, choose Delete.

   Or press DEL.

NOTE A program that fails can affect the stability of MS-DOS version 5.0, so you should leave MS-DOS Shell and restart your system after quitting the program.

Associating Files with a Program

If you have a set of files that you often use with a particular program, you can save time by *associating* the files with the program. Then, when you open an associated file, the program starts with those files loaded.

For example, suppose you use WordPerfect. If you use .WP extensions for your WordPerfect documents, you can associate all .WP filenames with the WordPerfect program file. Then, whenever you open a .WP file, MS-DOS starts WordPerfect and loads that .WP file into the WordPerfect workspace.

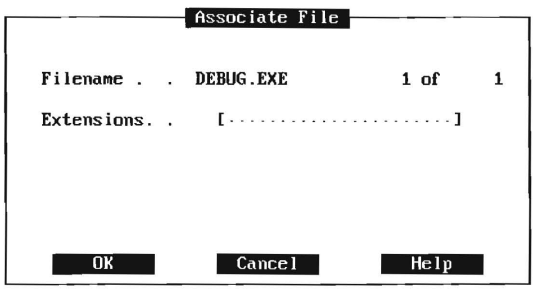
The number of extensions you can associate with a program can vary but cannot exceed 79 characters, including spaces. You do not have to type the period that appears before the extension. However, you can associate an extension with only one program at a time. For example, you cannot associate the extension .WP with WordPerfect and another text-editing program at the same time.

NOTE You can only associate files with a program if the program accepts filenames with the specified extensions at the command prompt.

▶ To associate files with a program:

1. Select the directory that contains the program you want to associate with a type of file.
2. From the file list, select the program's program file.
3. From the File menu, choose Associate.

   The Associate File dialog box appears.



4. In the Extensions box, type the filename extension that you want to associate with the selected program. It is not necessary to type the period that appears before the extension. (You can specify more than one extension by separating each with a space.)
5. Choose the OK button.

   You can also associate programs and files by first selecting a file, and then specifying the program name.

▶ To run an associated file with a different program:

1. From the File menu, choose Run.

   The Run dialog box appears.
2. Type the path and filename of the new program you want to use, followed by the name of the file.
3. Choose the OK button.

▶ To remove an association between a file type and a program:

1. Select the file that has the association you want to remove.
2. From the File menu, choose Associate.

   The Associate File dialog box appears. The program name is displayed in the text box.
3. Press BACKSPACE to delete the program name.
4. Choose the OK button.

   Or press ENTER.

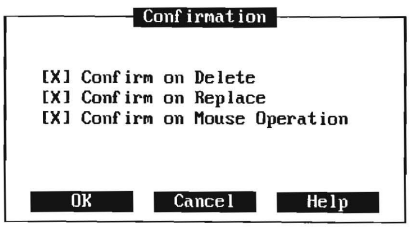
Suppressing Confirmation Messages

If you do not want to see a confirmation dialog box each time you delete or replace files and directories, you can change the confirmation settings. You also can suppress confirmation messages that appear when you are using a mouse.

▶ To suppress a confirmation message:

1. From the Options menu, choose Confirmation.

   The Confirmation dialog box appears.



2. For each confirmation option you want to suppress, clear the check box by clicking it or by using the UP ARROW or DOWN ARROW key, and then pressing the SPACEBAR.
3. Choose the OK button.

When they are selected, the options function as follows:

Confirm on Delete	Displays a warning message before deleting files
Confirm on Replace	Displays a warning message before replacing an existing file
Confirm on Mouse Operation	Displays a warning message before carrying out such commands as copying, dragging, or moving an item with a mouse

Repainting and Updating the Screen

If you run a terminate-and-stay resident (TSR) program from MS-DOS Shell, that program might still be displayed after you quit the program. To make the MS-DOS Shell window visible again, you need to repaint the screen.

▶ To repaint the screen:

Mouse

- From the View menu, choose Repaint Screen.

Keyboard

- Press SHIFT+F5.

  MS-DOS removes the program from the screen, and MS-DOS Shell appears.

Sometimes MS-DOS Shell does not update the screen to reflect files that have been added or deleted. For example, if you run a word-processing program from MS-DOS Shell and create new files with it, MS-DOS Shell does not display their filenames until you update the screen.

▶ To update the screen:

Mouse

- From the View menu, choose Refresh.

Keyboard

- Press F5.

MS-DOS Shell reads all the files on the current disk drive, just as it does when you start MS-DOS Shell. It then updates the screen to reflect any files and directories that you have added to or deleted from the disk.

Getting Help

Online Help provides a quick way to get information about MS-DOS Shell basics, and using menus, commands, dialog boxes, dialog box options, and procedures. You can get Help in three ways: by pressing F1; by selecting the Help button that appears in most dialog boxes; or by using the Help menu.

▶ To request Help on a menu:

1. Press ALT.
2. Select the menu you want Help on by using the LEFT ARROW or RIGHT ARROW key.
3. Press F1.

   A Help window containing information about the selected menu appears.

▶ To request Help on a command:

Mouse

1. Click the menu that contains the command you want Help on.
2. Select the command you want Help on by using the UP ARROW or DOWN ARROW key.
3. Press F1.

   A Help window containing information about the selected command appears.

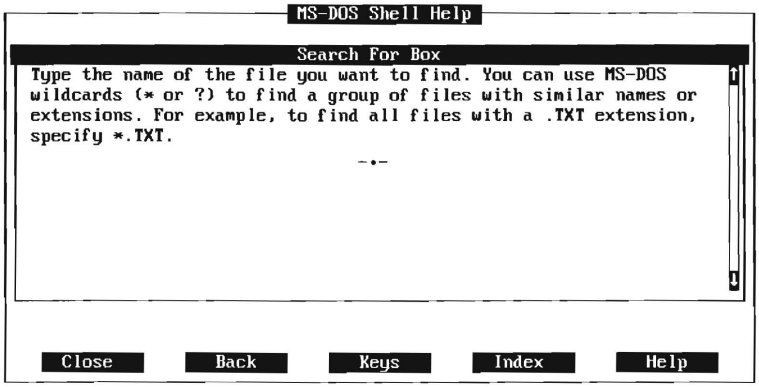
Keyboard

1. Press ALT to select the menu bar.
2. Select the menu that contains the command you want help on by using the LEFT ARROW and RIGHT ARROW keys.
3. Select the command you want Help on by using the UP ARROW and DOWN ARROW keys.
4. Press F1.

▶ To request Help on a dialog box option:

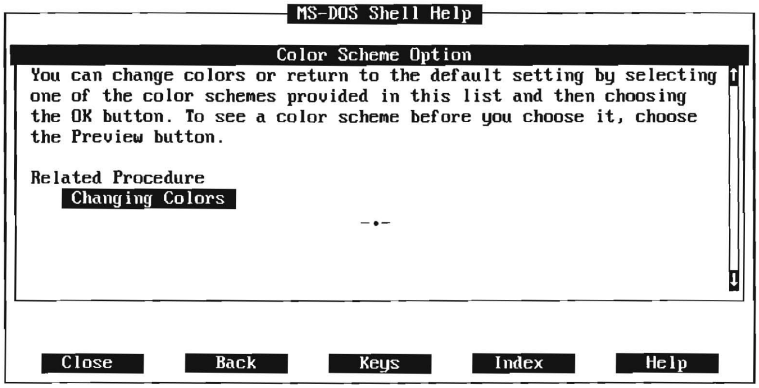
1. Open the dialog box you want Help on.
2. Select a command button or option by clicking it, or by using TAB or the arrow keys.
3. Press F1.

For example, if you have selected the Search For box in the Search File dialog box, and you press F1, MS-DOS Shell displays the following Help window:



Getting Help on a Related Procedure

Often Help refers you to a related procedure. For example, the following Help on the Color Scheme dialog box contains a reference to the procedure for changing colors.



In Help, related procedures are displayed in a different color or in reverse video, depending on the color scheme you have selected.

▶ To view a related procedure:

Mouse

- Double-click the related procedure.

  A Help window containing information about the related procedure appears.

Keyboard

1. Press TAB until the related procedure is selected.
2. Press ENTER.

   A Help window containing information about the related procedure appears.

Using the Help Menu

You can use the commands on the Help menu to view an index of Help topics; information on the keys you can use with MS-DOS Shell; basic skills for working with MS-DOS Shell commands and procedures; and information about using the Help system.

▶ To use the Help menu:

Mouse

- From the Help menu, choose the Help category you want.

  Either information about the subject or a list of topics related to the subject appears.

Keyboard

1. Press ALT, H.
2. Press the highlighted letter for the Help category you want.

   Or press the UP ARROW or DOWN ARROW key to select the Help category you want, and then press ENTER.

   Either information about the subject or a list of topics related to the subject appears.

NOTE If you haven't used Help before, choose Using Help from the Help menu to learn more about the type of information available.

Help Menu Options

The following items are on the Help menu:

Index	Provides a list of all MS-DOS Shell Help topics.
Keyboard	Lists keys and key combinations you can use with MS-DOS Shell.
MS-DOS Shell Basics	Provides an introduction to using MS-DOS Shell.
Commands	Explains all MS-DOS Shell commands. This information is organized according to the menu in which the command appears. (You can get the same information by selecting a command and then pressing F1.)
Procedures	Provides step-by-step instructions for performing tasks in MS-DOS Shell.
Using Help	Provides an introduction to using MS-DOS Shell Help.
About Shell	Displays copyright and version information about MS-DOS Shell.

Leaving MS-DOS Shell

You can leave MS-DOS Shell and move to the command prompt in two ways. You can quit MS-DOS Shell temporarily, in which case you can work at the command prompt while MS-DOS Shell is still in your system's memory. Or you can quit MS-DOS Shell and remove it from your system's memory before you switch to the command prompt.

▶ To leave MS-DOS Shell temporarily:

- Press SHIFT+F9.

  Or from the Main group on the program list, choose Command Prompt.

To return to MS-DOS Shell, type **exit**, and then press ENTER. If Task Swapper is enabled, you can switch back to MS-DOS Shell without quitting Command Prompt by pressing CTRL+ESC.

If you made changes to a directory such as adding or deleting files, the changes will not be displayed in the file-list area until you update the directory. For information about updating a directory, see "Updating a Directory" earlier in this chapter.

▶ To quit MS-DOS Shell:

1. If there are programs displayed in the Active Task List, quit each program.
2. From the File menu, choose Exit.

   Or press F3.

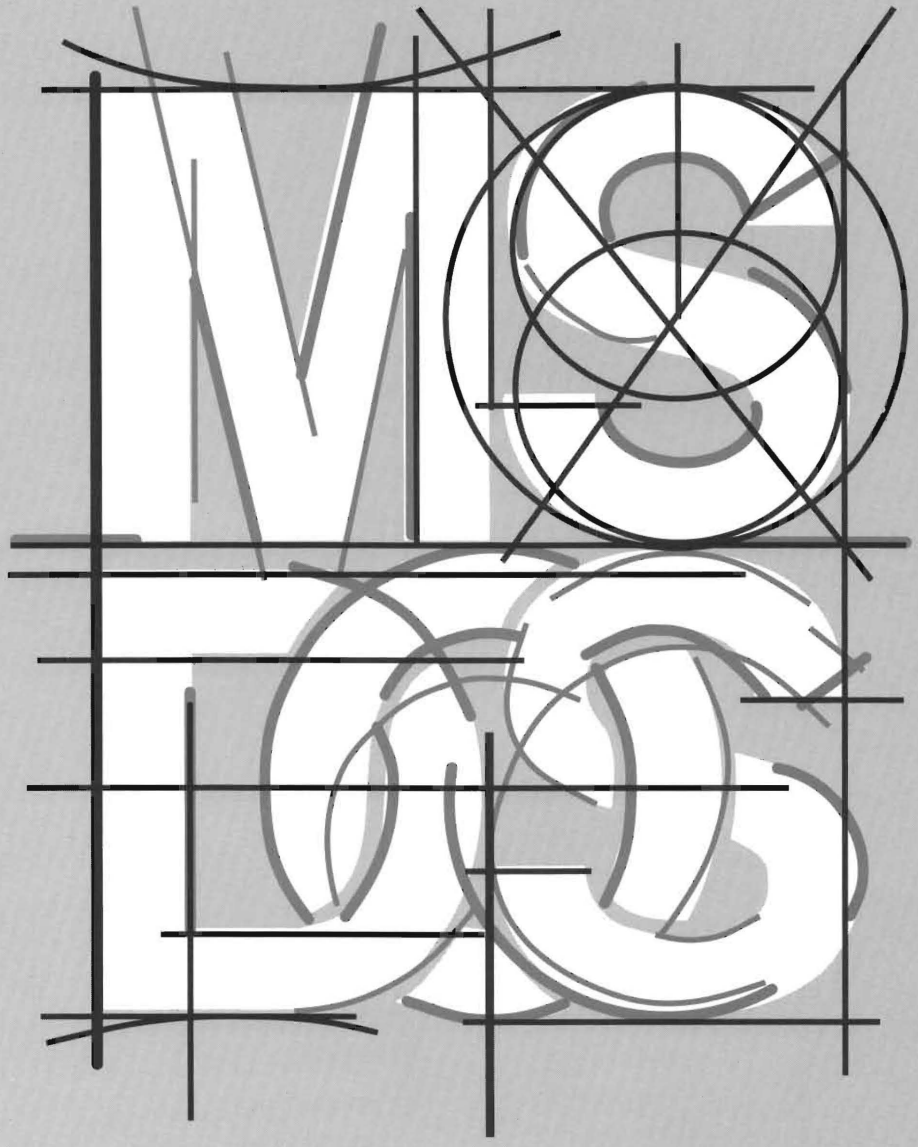
   Or press ALT+F4.

If you try to quit MS-DOS Shell while you still have programs listed in the Active Task List, the Exiting Error dialog box appears. The dialog box tells you that you cannot quit MS-DOS Shell without first quitting all programs that you have running. Choose the OK button to close the dialog box.

▶ To start MS-DOS Shell from the command prompt:

- Type **dosshell**, and then press ENTER.

Part 2
Working with MS-DOS



Chapter 4

Working with Files

4

Files organize information that your computer uses. MS-DOS itself is stored in files that are read into memory when you start your system. There are a number of types of files; each file has a unique name and location in addition to a group of attributes that describes it.

Filenames

Every file has a name. Most files also have an extension. The name always appears first, and the extension is always separated from the name by a period, as in the following example:

```
myfile.ext
```

In this guide, the file's name and extension are referred to as the *filename*.

Names

Each file has a name for identification. For example, the COMMAND.COM file contains MS-DOS commands, and the MOUSE.COM file contains information your system needs if you use a mouse.

The names you choose for files must:

- Have no more than eight characters.
- Contain only the letters A through Z, the numbers 0 through 9, and the following special characters: underscore (_), caret (^), dollar sign ($), tilde (~), exclamation point (!), number sign (#), percent sign (%), ampersand (&), hyphen (-), braces ({ }), parentheses ( ), at sign (@), apostrophe ('), and the grave accent (`). No other special characters are acceptable.

- Not contain spaces, commas, backslashes, or periods (except the period that separates the name from the extension).
- Not be the following reserved filenames: CLOCK$, CON, AUX, COM*n* (where *n* = 1–4), LPT*n* (where *n* = 1–3), NUL, and PRN.

NOTE You can use extended characters in a filename, but if you do, it is recommended that you use code page 850. If you use code page 437, support for extended characters is limited. For information about extended characters, see the Appendix.

Extensions

Extensions help you identify the type of file. MS-DOS uses the following extensions:

- .EXE (executable) or .COM (command) for files that contain programs
- .SYS (system) for files that contain information about your hardware (for example, your mouse or expanded memory)
- .BAT (batch) for files containing lists of commands that MS-DOS carries out as a set

When you create a file, you can choose an extension that helps you identify the file. The extension must contain no more than three characters. The filename restrictions regarding characters and spacing also apply to extensions.

Most programs that create files add an extension. It is generally best to use the extension specified.

Types of Files

Much of the information stored in files is text, such as letters, punctuation, and special characters. Because you perform different tasks with different types of files, it is important to be able to recognize the types of files and what each is used for.

Program Files

Program files contain the programs that your computer runs. Program files usually have an .EXE or .COM extension. Lotus 1–2–3, for example, is stored in a file called 123.EXE.

For information about how to use MS-DOS Shell to organize and run your program files, see Chapter 8, "Customizing MS-DOS Shell."

Specialized Data Files

It is common for a program to produce files that contain codes which can be used only by that program.

For example, when you create a spreadsheet data file, the spreadsheet program saves the file in a format that only it can read. Sometimes a program assigns an extension to the files it creates. For example, Microsoft Word assigns a .DOC extension to its document files.

Unformatted Text Files

Unformatted text files contain only text. Almost all computer programs, including MS-DOS, use a system called the American Standard Code for Information Interchange (ASCII) to represent text. Files of this type often have a .TXT extension.

System Files

System files contain information about your hardware and are sometimes called device drivers. These files usually have a .SYS extension. For more information, see Chapter 15, "Device Drivers."

Batch Programs

Batch programs are unformatted text files that contain MS-DOS commands. If you often type the same set of commands to start a program, you can put them into a batch program. Instead of typing the commands each time, you can use the batch program, which carries out the commands for you. Batch files always have a .BAT extension. For information about batch programs, see Chapter 10, "Working with Batch Programs."

File Size, Date, and Time

MS-DOS stores information about the size of files and the date and time they were created or modified. You can view this information by using the **dir** command. For example, MS-DOS might display the following information in response to a **dir** command:

```
Volume in drive A is LARKA
Volume Serial Number is 1E51-12FB
Directory of A:\

BACKUP   COM     36880  12-07-90  12:00a
DISKCOPY COM     10396  12-03-90  12:00a
FORMAT   COM     22876  12-07-90  12:00a
KEYB     COM     14727  12-06-90   1:40a
        4 File(s)     84879 bytes
                      112384 bytes free
```

NOTE When you use the **dir** command, the periods that separate filenames and extensions are not displayed in the listing. The names and extensions of the files are separated by spaces. If you use the **dir** command with the **/w** switch, however, the periods are displayed. The **/w** switch is described later in this chapter.

Next to the filename, MS-DOS displays the size of the file. Files are measured in bytes. One byte is the amount of space it takes to store a single character. File size indicates how much disk space is occupied by the file.

To the right of the file size, MS-DOS displays the date and time the file was created or last changed. MS-DOS revises the date and time only when you change the contents of the file. The time and date don't change when you copy the file or rename it.

The size, date, and time information can help you keep track of your files. For example, you might want to know whether two files with different filenames contain the same information. One way to determine whether the two files are the same is to look at their sizes, dates, and times. If the size and date are the same for both files, it's likely that their contents are identical. If you want to be sure that two files are the same, use the **fc** command (discussed later in this chapter).

When you are backing up files, you often have two or more files with the same filename in different directories or on different disks. You can use file size and date to determine which file is the most recent.

Using Wildcards

If you want to perform the same task for a group of files, you don't have to use the same command repeatedly for each filename in the group. You can use wildcards to specify groups of files. A wildcard acts as a substitute for a name or extension.

There are two wildcards:

- The asterisk (*) represents a whole word or a group of characters.
- The question mark (?) represents a single character.

Using Wildcards to Specify Groups of Files

Suppose a disk in drive A contains various MS-DOS command (.COM) files. You could use the following **dir** command to view a list of all files that have a .COM extension:

```
dir a:*.com
```

Another way to view filenames on a disk is to use the **dir** command with the **/w** switch. The **/w** switch lists only the filenames and directory names and displays them across the width of the screen in several columns. You could view a list of files on a disk in drive A by using the following command:

```
dir a: /w
```

When you use the preceding command, MS-DOS displays a list similar to the following:

```
Volume in drive A is LARKA
Volume Serial Number is 1E51-12FB
Directory of A:\

AUTOEXEC.BAT  BACKUP.EXE    COMMAND.COM   CONFIG.SYS    COUNTRY.SYS
DISKCOPY.COM  DISPLAY.SYS   EGA.CPI       FDISK.EXE     FORMAT.COM
KEYB.COM      KEYBOARD.SYS  LCD.CPI       MODE.COM      REPLACE.EXE
RESTORE.EXE   SYS.COM
       17 File(s)     566862 bytes
                       77824 bytes free
```

You can use wildcards to view selected groups of filenames in this listing. For example, to view only filenames that have the .COM extension, type the following command:

```
dir a:*.com /w
```

MS-DOS lists the program files that have the .COM extension:

```
Volume in drive A is LARKA
Volume Serial Number is 1E51-12FB
Directory of A:\

COMMAND.COM   DISKCOPY.COM  FORMAT.COM    KEYB.COM
MODE.COM      SYS.COM
        6 File(s)     197439 bytes
                       77824 bytes free
```

In addition to substituting for an entire name or extension, the asterisk wildcard can substitute for parts of a name or extension. For example, to view filenames that begin with the letter C on a disk in drive A, type the following command:

```
dir a:c*.* /w
```

MS-DOS lists the files that have names beginning with the letter C:

```
Volume in drive A is LARKA
Volume Serial Number is 1E51-12FB
Directory of A:\

COMMAND.COM   CONFIG.SYS    COUNTRY.SYS
        3 File(s)      50459 bytes
                       77824 bytes free
```

If you use the asterisk wildcard to copy or delete a file, be careful not to specify a group of files instead of a single file. For example, if you have a file called MYTES.TXT and another named MYSALE.TXT, both files will be copied to drive A if you type the following command:

```
copy my*.txt a:
```

Unlike the asterisk wildcard, which substitutes for any or all characters in a name or extension, the question mark (?) wildcard substitutes for a single character. For example, to list the files that have names of up to three letters, type the following command:

```
dir a:???.* /w
```

MS-DOS lists files that have names of up to three letters, regardless of the name or extension:

```
Volume in drive A is LARKA
Volume Serial Number is 1E51-12FB
Directory of A:\

SYS.COM       EGA.CPI       LCD.CPI
        3 File(s)      71227 bytes
                       77824 bytes free
```

Using Wildcards to Match Files

You can use wildcards to match one file or group of files with another. For example, to change all files with a .BAT extension to a .BAK extension on a disk in drive A, you could use the following **rename** (**ren**) command:

```
ren a:*.bat *.bak
```

The first and second wildcards are used in different ways. MS-DOS uses the first to find all files that have a .BAT extension; it uses the second to create a name that matches the original name of each .BAT file.

To copy all files on a disk in drive A that have names beginning with F and ending with .BAT extensions to a disk in drive B, keeping their original names but changing their extensions to .BAK, you could use the following **copy** command:

```
copy a:f*.bat b:*.bak
```

Viewing Text Files

In Brief

To view the contents of a text file, use the **type** command. For example, to view the contents of the LIST.TXT file on a disk in drive B, you would use the following command:

```
type b:list.txt
```

If the file you want to view is large, you could use a pipe (¦) followed by the **more** command, as follows:

```
type b:list.txt ¦ more
```

By including the **more** command, you can view the file one screen at a time.

Use the **type** command to view the contents of unformatted text files and batch programs. (You can look at the contents of other kinds of files, too. However, only the text is readable.) When you use the **type** command, MS-DOS displays the entire file on your screen. You cannot change the text or view only a portion of the file.

For example, you can use the **type** command to see the contents of your AUTOEXEC.BAT file. AUTOEXEC.BAT is a batch program that carries out certain commands when you start your system. If you have a hard disk, the

AUTOEXEC.BAT file should be in your root directory on drive C. If you have a floppy disk system, you'll find it on the disk you use to start your system.

If you have a floppy disk system, you can view the AUTOEXEC.BAT file by putting the disk you use to start your system in drive A and typing the following command:

```
type a:autoexec.bat
```

If you have a hard disk, and it is drive C, type this command to view your AUTOEXEC.BAT file:

```
type c:\autoexec.bat
```

MS-DOS displays the entire file. If a file contains more information than can fit on the screen, MS-DOS displays the text more quickly than you can read it. To avoid this problem, you can use the **more** command with the **type** command, as follows:

```
type c:\autoexec.bat ¦ more
```

You must precede **more** with a pipe (¦). By using this command, you can view your file one screen at a time. For information about the **more** command, see Chapter 7, "Advanced Command Techniques" or Chapter 14, "Commands."

TIP If you don't use the **more** command when you use **type**, you can temporarily stop the display of a file by pressing CTRL+S or the PAUSE key while the file is scrolling. To start viewing more of the file, press any key (except PAUSE).

To stop viewing a file, press CTRL+C or CTRL+BREAK, which cancels the **type** command.

Shell ▶ **To view a file:**

1. From the file list, select the file you want to see.
2. From the File menu, choose View File Contents, or press F9.

   The selected file appears in the File View window.
3. To scroll through the contents of the file, use the PAGE UP and PAGE DOWN keys or the UP ARROW and DOWN ARROW keys.

   To switch between viewing the file as ASCII text and as hexadecimal code, press F9.
4. When you are finished viewing the file, press ESC. Or, from the View menu, choose Restore View.

Copying Files

The **copy** command is your primary tool for organizing and making copies of files. With the **copy** command you can do the following:

- Copy a file from one directory or disk to another
- Copy a group of files by using wildcards
- Rename a file as you copy it
- Combine two or more files into one file

For information about copying an entire disk, see Chapter 6, "Managing Disks."

CAUTION When you use the **copy** command, avoid inadvertently destroying a file by copying over it. For example, if you copy a file called SCORES.DAT to a directory that already has a file with that name, MS-DOS replaces the existing file with the new copy.

Copying a Single File

In Brief

To copy a file to another disk or directory, use the **copy** command. For example, you would use the following command to copy the PROB.DBS file from a disk in drive A to a disk in drive B:

```
copy a:prob.dbs b:
```

When using the **copy** command, you type the location and filename of the file you want to copy from, followed by the location and filename of the file you want to copy to. The first file is called the *source* file and the second file is called the *destination* file.

For example, to copy the OUTGO.XLS file from a disk in drive A to a disk in drive B, you would type the following command:

```
copy a:outgo.xls b:outgo.xls
```

MS-DOS makes a copy of the OUTGO.XLS file on the disk in drive A and puts it on a disk in drive B in a file having the same filename. If you want the source and destination files to have the same filename, you omit the destination filename.

For example, you could use the following command to produce the same result achieved by using the previous command:

```
copy a:outgo.xls b:
```

After you use the **copy** command, MS-DOS indicates how many files were copied:

```
1 File(s) copied
```

If MS-DOS cannot find the file you want to copy, it displays a "File Not Found" message. Check to see that you typed the filename correctly and that the file is in the directory you specified.

Copying a Group of Files

In Brief

To copy a particular group of files from one disk or directory to another, use the **copy** command with wildcards. For example, you could use the following command to copy all files with a .DBS extension from the current directory to a disk in drive B:

```
copy *.dbs b:
```

Suppose you have a number of files on a disk in drive A that you designated as temporary by giving them a .TMP extension. If you want to copy these files to a disk in drive B, you can use the asterisk wildcard:

```
copy a:*.tmp b:
```

When you use wildcards to specify the files to be copied, MS-DOS displays the filenames as it copies the files.

```
A:JAN.TMP
A:MAR.TMP
A:FEB.TMP
 3 File(s) copied
```

You might have a group of files with filenames that differ only slightly. To copy them, you can use the question mark wildcard. For example, suppose you have four reports—JAN1RPT.DOC, JAN2RPT.DOC, JAN3RPT.DOC, and JAN4RPT.DOC—on a disk in drive A. To copy the files to a disk in drive B, you could use the following command:

```
copy a:jan?rpt.doc b:
```

From drive A, MS-DOS copies the files that have seven-letter names (beginning with the letters JAN and ending with the letters RPT) and a .DOC extension.

If the disk you are copying to doesn't have enough space, MS-DOS stops copying and displays a message that tells you there is insufficient disk space and indicates how many files were copied. The last file displayed in the list before MS-DOS stops copying is not copied to the new disk.

For example, if the disk in drive B becomes full after three of the four .DOC files are copied, you would see a list similar to the following:

```
JAN1RPT.DOC
JAN2RPT.DOC
JAN3RPT.DOC
JAN4RPT.DOC

Insufficient disk space
3 File(s) copied
```

MS-DOS does not copy the JAN4RPT.DOC file to the disk in drive B.

Renaming a File as It Is Copied

In Brief

To rename a file as it is copied, use the **copy** command and specify the new name, as in the following command:

```
copy prob.dbs b:noprob.dbs
```

If you want to assign a new name to a file you are copying, specify the new filename as the destination file. (If you want to rename a file without making a copy, use the **rename** command discussed later in this chapter.)

For example, to copy the OUTGO.XLS file from a disk in drive A to a disk in drive B and rename the file EXPEND.XLS, use the following command:

```
copy a:outgo.xls b:expend.xls
```

You can rename groups of files by using wildcards. For example, if you want to copy .TMP files on a disk in drive A to a disk in drive B and give the files an .OLD extension, use the following command:

```
copy a:*.tmp b:*.old
```

You can copy a file to the same directory if you rename the file. If you don't rename the file when copying it to the same directory, MS-DOS displays the following message:

```
File cannot be copied onto itself
0 File(s) copied
```

Combining Text Files

In Brief

To combine two or more unformatted text files into one file, use the **copy** command and include a plus sign (+) between the files you want to combine, as in the following command:

```
copy prob.txt + dob.txt probdob.txt
```

You can use the **copy** command to combine two or more unformatted text files into one file. For example, the following command joins the files SCENE1.TXT and SCENE2.TXT on a disk in drive A into a new file named ACT1.TXT on a disk in drive B:

```
copy a:scene1.txt + a:scene2.txt b:act1.txt
```

MS-DOS joins the files in the order you type them. In this example, MS-DOS adds the SCENE2.TXT file to the end of the SCENE1.TXT file.

If you don't specify a destination file, MS-DOS combines all the files in the first file you specify. For example, to include the SCENE3.TXT file at the end of the ACT1.TXT file, you would use this command:

```
copy act1.txt + scene3.txt
```

You can also use wildcards to combine groups of files:

```
copy *.txt all.txt
```

Copying Text from the Keyboard to a File

In Brief

To copy text from the keyboard to a file, first create an unformatted text file by using the **copy** command followed by CON, as in the following example:

```
copy con note.txt
```

You can use the **copy** command to copy from your keyboard to a file. To copy from your keyboard, specify CON as the source file and specify a filename as the destination file. In effect, your keyboard becomes the source file. For

example, you could use the following command to type directly into a file named NOTE.TXT on a disk in drive A:

```
copy con a:note.txt
```

This command copies whatever you type on the keyboard to the file NOTE.TXT on a disk in drive A. After you type the command, MS-DOS displays a cursor but no command prompt.

If NOTE.TXT doesn't exist on the disk in drive A, MS-DOS creates it. If it does exist, what you type replaces what is already in the file. You need to press ENTER at the end of each line of text you type. When you finish typing into the file, press CTRL+Z and then press ENTER to close the file.

NOTE You can use the editing keys described in Chapter 7, "Advanced Command Techniques," to edit the line of text you are typing. However, after you press ENTER you cannot change text you have typed.

Copying a File to a Printer

In Brief

To print an unformatted text file, copy it to the port to which your printer is attached, as in the following command:

```
copy note.txt lpt1
```

To copy a file to your printer, you can specify as the destination file the name of the port to which the printer is attached. For example, the following command copies a file named NOTE.TXT on a disk in drive A to the printer attached to the LPT1 port:

```
copy a:note.txt lpt1
```

Using CON and the name of a port, you can copy directly from your keyboard to a printer—for example, to the printer attached to the LPT1 port:

```
copy con lpt1
```

When you are finished sending information to the printer, press CTRL+Z and then press ENTER to print what you have typed.

For more information about printer ports, see "Printing Files" later in this chapter.

Copying Files by Using MS-DOS Shell

If you are using MS-DOS Shell, you can use the Copy command on the File menu to copy any number of files from one directory to another. When you choose Confirm On Replace from the Options menu, MS-DOS prompts you to confirm the command if you are copying over an existing file.

NOTE When a file or group of files is selected, you use the Copy command on the File menu to copy files. However, when an item from the program list is selected instead, you use the Copy command as a tool for organizing how program items are displayed. For more information about organizing program items, see Chapter 8, "Customizing MS-DOS Shell."

Shell ▶ **To copy files:**

Mouse

1. Make sure the file(s) and destination directory (or drive icon) are visible.
2. Press and hold down CTRL while you drag the file(s) to the destination directory or drive icon.
3. Release the mouse button, and then release CTRL.

   The Confirm Mouse Operation dialog box appears. (You can suppress this confirmation message by choosing Confirmation from the Options menu. For more information, see Chapter 3, "MS-DOS Shell Basics.")
4. Click the Yes button.

Keyboard

1. From the file list, select the file or files you want to copy.
2. From the File menu, choose Copy.

   The Copy File dialog box appears, with the selected filename(s) in the From box, and the currently selected directory in the To box.
3. In the To box, type the drive and directory that you want to copy your files to.
4. Choose the OK button.

Renaming a File

In Brief

To rename a file, use the **rename** (**ren**) command, as in the following example:

```
ren mytax.dat ourtax.dat
```

To change the name of a file without changing its location, use the **rename** command. This command is especially helpful for organizing files. For example, suppose you have two versions of a file named PRICES.LST. The version on the disk in drive A contains last year's prices, whereas the version on drive C is current. To avoid confusion between the two files, you can use the following command to rename the file that contains outdated prices:

```
ren a:prices.lst prices.old
```

You can use wildcards to rename a group of files. For example, if you want all .TMP files to be renamed to .TXT files in a directory, use the following command:

```
ren *.tmp *.txt
```

Shell ▶ **To rename files:**

1. From the file list, select the file or files you want to rename.
2. From the File menu, choose Rename.

   The Rename File dialog box lists the current name of the first selected file.
3. In the New Name box, type the new filename.
4. Choose the OK button.

   If you select more than one file, the Rename File dialog box prompts you to rename each file separately.

Printing Text Files

You can print unformatted text files from MS-DOS by using the **print** command. Generally, it is best to print formatted text files and other specialized files from the program you used to create them. For information about specialized data files and unformatted text files, see "Types of Files" earlier in this chapter.

MS-DOS has print options that another program may lack. With MS-DOS, you can start a print job and do other tasks while the printer prints. You can also specify a list of files to print so you don't have to print each file separately.

When you use the **print** command, a print queue is set up. For information about the print queue, see "Using the Print Queue" later in this chapter.

Before printing, make sure your printer is properly connected to your system, that the power is turned on, and that the printer is online. For information about your printer, see your printer documentation.

Printing Files

In Brief

To print an unformatted text file from MS-DOS, use the **print** command, as in the following example:

```
print config.sys
```

MS-DOS prompts you for the port you want to print to. If you don't specify a port, MS-DOS connects to LPT1 (also called PRN).

You can set the default printer port by using the **/d** switch the first time you use the **print** command. For example, if your printer is connected to your COM1 port, use the following command:

```
print /d:com1
```

Suppose you want to print a file named TOGO.TXT. Type the following command:

```
print togo.txt
```

The first time you use the **print** command, MS-DOS prompts you for the port to print to. Whenever you use the **print** command again, MS-DOS uses the port you specified the first time. To change the port setting, you must restart your system and use the **print** command again. You can set the default printer port by using the **/d** switch, as in the following example:

```
print /d:com1 togo.txt
```

The **/d** switch can only be used the first time you use the **print** command. Always use a colon (:) to separate the **/d** switch from the port name.

To print more than one file at a time, you can type the various filenames and use spaces to separate them, or you can use wildcards (described earlier in this chapter).

For more information about the **print** command, see Chapter 14, "Commands."

Shell ▶ **To print files:**

1. Before using MS-DOS Shell, type **print** at the command prompt.

   You can add the **print** command to your AUTOEXEC.BAT file to avoid having to type it each time you run MS-DOS Shell. The **print** command must precede the **dosshell** command in the AUTOEXEC.BAT file. For information about the AUTOEXEC.BAT file, see Chapter 11, "Customizing Your System."
2. From the file list, select one or more files.
3. From the File menu, choose Print.

   MS-DOS adds the files to the print queue (the list of files waiting to be printed).

Using the Print Queue

In Brief

To view the contents of the print queue, use the **print** command:

```
print
```

To cancel printing and empty the print queue, use the **/t** switch:

```
print /t
```

The MS-DOS print queue is a list of files waiting to be printed. The file at the top of the queue is the one currently being printed. MS-DOS adds your files to the print queue when you use the **print** command. After the first file is printed, MS-DOS begins printing the second file.

To see the list of files in the print queue, type the following:

```
print
```

MS-DOS lists the files in the queue and indicates which one is currently printing.

To cancel printing and empty the print queue, use the **/t** switch:

```
print /t
```

MS-DOS stops sending information to your printer. The printer keeps printing until it prints all the information stored in its memory.

NOTE The print queue requires memory that MS-DOS and your programs could otherwise use. Therefore, using the print queue may affect the efficiency of your system. To print without the print queue, use the **copy** command described earlier in this chapter.

Deleting Files

As you work with MS-DOS, you may want to remove files that are no longer useful. You can delete a single file, selected groups of files, or all files in a directory or on a disk. Once you delete files, you might not be able to recover them. Be sure that the files you specify for deletion are the ones you want to remove.

If you accidentally delete files you wanted to keep, use the **undelete** command as soon as possible to recover them. The **undelete** command might not be able to recover the deleted files if you created or changed other files on the disk in the meantime. The **undelete** command works best if you set up your system to keep track of files you delete.

Tracking Deleted Files

In Brief

If you want MS-DOS to keep track of deleted files on a disk, you can use the Mirror program. To install the program, use the **mirror** command with the **/t** switch. For example, the following command sets up deletion tracking for a disk in drive A:

```
mirror /ta
```

The drive letter must follow the **/t** switch. However, do not use a colon (:) with the drive letter.

The Mirror program is a 6.4K terminate-and-stay-resident (TSR) program that monitors the system. Whenever the program detects a deleted file, it records information that the **undelete** command needs to recover the file. To install this program, use the **mirror** command with a **/t** switch for each drive you want to track. For example, to install the program so that it tracks file deletions on drives A and C, you would type the following:

```
mirror /ta /tc
```

The first time you delete a file on drive A or C after you have installed Mirror, a file called PCTRACKR.DEL is created in the root directory of that drive. This is a system file that contains the information which **undelete** needs to recover the deleted file. Each time you delete a file, PCTRACKR.DEL is updated with information about that file.

CAUTION Do not use deletion tracking for any drive that has been redirected by using the **join** or **subst** command. If you intend to use the **assign** command, you must do so before using Mirror to install deletion tracking.

The Mirror program stores a default number of entries in this file, depending on the size of the disk for which it is tracking deletions. Once you have deleted the default number of files, Mirror replaces the information for the first file you deleted with information for the next file you delete. The entry for the second deleted file is replaced when you delete yet another file, and so on. Limiting the number of deletions keeps the deletion-tracking file from growing so large that there is no longer enough free space on the disk for other files.

The following table shows the default number of entries and the size of the PCTRACKR.DEL file that the Mirror program creates for several disk sizes:

Disk size	Number of entries	Size of PCTRACKR.DEL file
360K	25	5K
720K	50	9K
1.2 MB	75	14K
1.44 MB	75	14K
20 MB	101	18K
32 MB	202	36K
Over 32 MB	303	55K

You can override these default values when you use the **/t** switch by adding a hyphen (-) after the drive letter and then typing the number of file deletions to be tracked for that drive. You can track as few as one or as many as 999 deletions.

For example, to install the deletion-tracking program for drive C and set the maximum number of deletions to be tracked to 500, you could use the following command:

```
mirror /tc-500
```

NOTE Using Mirror may slow down your system if you delete a large number of files.

For more information, see the **mirror** command in Chapter 14, "Commands."

In most cases, you should load the deletion-tracking program whenever you start your system. A convenient way of doing this is to put the **mirror** command

in your AUTOEXEC.BAT file, specifying the **/t** switch for each drive in your system. For information about the AUTOEXEC.BAT file, see Chapter 11, "Customizing Your System."

Deleting a Single File

In Brief

To remove a file from a disk, use the **del** (**erase**) command, as in the following example:

```
del a:temp.exe
```

You delete a single file by typing the **del** command, followed by the location and name of the file you want to delete. For example, to delete a file named TEST.TMP file from a disk in drive B, type the following:

```
del b:test.tmp
```

MS-DOS removes the file from the disk. Instead of **del**, you can use **erase**. For example, the following is equivalent to the previous command:

```
erase b:test.tmp
```

If you add the **/p** switch to the **del** command, MS-DOS prompts you to check your filename, as follows:

```
b:\TEST.TMP,  Delete (Y/N)?
```

Type **y** to delete the specified file, or type **n** to cancel the command.

Deleting a Group of Files

In Brief

To delete a group of files, use the **del** command with one or more wildcards, as in the following example:

```
del a:*.tmp
```

You can use wildcards to delete a group of files. For example, the following command deletes all files with the .TMP extension on a disk in drive A:

```
del a:*.tmp
```

Before using wildcards to delete a group of files, it is helpful to use the **dir** command to determine what files the wildcards might delete. Developing this habit may prevent you from accidentally deleting files you intended to keep. For

information about the **dir** command, see Chapter 5, "Working with Directories" or Chapter 14, "Commands."

Deleting All Files in a Directory

In Brief

To delete all files in a directory or drive, use the **del** command with wildcards, as in the following example:

```
del b:*.*
```

To clear a directory of all files, you can use the **del** command and wildcards. For example, to delete all the files in the \TMP directory on drive C, type the following:

```
del c:\tmp\*.*
```

If you do not specify a directory, all files in the current directory are deleted.

Whenever you specify ***.*** with the **del** command, MS-DOS prompts you to confirm the deletion. If you are sure you want to delete all the files, type **y**. MS-DOS then deletes every file in the specified directory.

If you type the directory without specifying any files, it is assumed you want to delete all the files in that directory. For example, to delete all files in the \TMP directory on drive C, you could type the following command:

```
del c:\tmp
```

Deleting Files by Using MS-DOS Shell

You can use the Delete command in MS-DOS Shell to delete one or more files.

Shell ▶ **To delete a single file:**

1. From the file list, select the file you want to delete.
2. From the File menu, choose Delete.

   Or press the DEL key.

   The Delete File Confirmation dialog box appears. (You can suppress this confirmation message by choosing Confirmation from the Options menu. For more information, see Chapter 3, "MS-DOS Shell Basics.")
3. Choose the Yes button.

Shell ▶ **To delete more than one file:**

1. From the file list, select the files you want to delete.
2. From the File menu, choose Delete.

   Or press the DEL key.

   The Delete File dialog box appears.
3. Choose the OK button.

   The Delete File dialog box appears. (You can suppress this confirmation message by choosing Confirmation from the Options menu. For more information, see Chapter 3, "MS-DOS Shell Basics.")
4. Choose the Yes button.

Recovering Deleted Files

In Brief

To recover an accidentally deleted file, use the **undelete** command, as in the following example:

```
undelete a:temp.exe
```

When you delete a file, MS-DOS does not delete the data in the file. Instead, it *marks* the file as deleted so MS-DOS can reuse the area of the disk that was occupied by the deleted file. The data remains on the disk until MS-DOS records the data of another file in the same region of the disk.

If you have installed the Mirror program, it keeps track of the area of the disk that was used by the deleted files. You install this deletion-tracking program by using the **mirror** command with the **/t** switch. For more information, see "Tracking Deleted Files" earlier in this chapter.

Because the data in a deleted file remains intact for a while, it is possible to recover a file that was accidentally deleted. As soon as you discover that the file has been deleted, use the **undelete** command to restore the file.

CAUTION Some MS-DOS commands (such as **more**) create temporary files that can replace areas of a deleted file. Therefore, avoid using any programs or any MS-DOS command except **undelete** until you restore the file you accidentally deleted.

You can use one or more wildcards if you want to recover more than one file. For example, to restore all files that have a .BAT extension on a disk in drive A, you would type the following:

```
undelete a:*.bat
```

If you don't supply a filename or wildcard, **undelete** attempts to restore all deleted files that it can locate on the disk.

CAUTION The **undelete** command cannot restore a directory that has been removed, and it cannot recover a file if you have removed the directory that contained the file.

Although **undelete** works best when you have installed Mirror, **undelete** can sometimes recover deleted files when your system is not keeping track of deleted files. If deletion tracking is not installed, MS-DOS attempts to recover the files by using information from the disk's root directory and file allocation table. For more information about the root directory and file allocation table, see Chapter 6, "Managing Disks."

When MS-DOS attempts to recover files, and deletion tracking is not installed, you are prompted to type the first letter of the filename for each file you want to recover.

If you want **undelete** to use the information from the root directory and file allocation table even though deletion tracking is installed, you can use the **/dos** switch with the **undelete** command. When you use this switch with **undelete**, you are prompted to type the first letter of the filename for each file you want to recover.

If you are trying to recover several files, you can use the **/all** switch with the **undelete** command. In this case, MS-DOS does not display the message asking whether you want to recover a file; it recovers all files that it can. If deletion tracking is not installed and you use the **/all** switch, with **undelete** the first character of each recovered file is replaced with the number (#) sign.

For more information about the **undelete** command, see Chapter 14, "Commands."

Moving Files

In Brief

To move a file from one disk or directory to another, use the **copy** command, then the **del** command, as in the following example:

```
copy a:outgo.xls b:
del a:outgo.xls
```

Using the **copy** and **del** commands, you can move one or more files from one directory or disk to another. Moving files requires that you first copy them to their new location. After copying the files, delete them from their original directory. For example, to move a group of .TMP files from a disk in drive A to a disk in drive B, use the following two commands:

```
copy a:*.tmp b:*.tmp
del a:*.tmp
```

Shell ▶ **To move files:**

Mouse

1. Select the file(s) that you want to move.
2. Drag the file(s) to the destination directory or drive icon.
3. Release the mouse button.

   A confirmation message appears. (You can suppress this confirmation message by choosing Confirmation from the Options menu. For more information, see Chapter 3, "MS-DOS Shell Basics.")
4. Click the Yes button.

Keyboard

1. From the file list, select the file or files you want to move.
2. From the File menu, choose Move.

   The Move File dialog box appears, and the From box lists the file or files you selected.
3. In the To box, type the drive and directory that you want to move the file or files to. If you want to rename a file in its new location, type the new filename after the directory name.
4. Choose the OK button.

Comparing Files

In Brief

To see whether two files or groups of files have the same contents, use the **fc** command, as in the following example:

```
fc a:mytax.dat b:mytax.dat
```

MS-DOS compares the files and displays differences.

To get an approximate comparison of two files, you can look at file size and time of creation. To get a precise comparison of two files, use the **fc** command. For example, suppose you have an unformatted text file named FOODPRP.TXT on a disk in drive A and also on a disk in drive B. To find out whether the files are identical, type the following:

```
fc /a a:foodprp.txt b:foodprp.txt
```

The **/a** switch in this example abbreviates the output for the comparison of two text files. MS-DOS starts at the beginning of the two files and compares each byte. When MS-DOS finds a difference, it displays the filename, the line of text that begins a set of differences, and the line that ends the set of differences, as in the following example:

```
*****foodprp.txt
Our expected revenues for the month of January are expected to rise
\...
when the results are not yet certain.
*****foodprp.txt
Our expected revenues for January are less than projected
\...
when the results are not yet certain.
*****
```

If you want the results to be stored in a file rather than displayed on your screen, use the greater-than sign (>) to redirect the output. For example, the following command stores the results of the **fc** command in the COMPARE.TXT file:

```
fc /a a:foodprp.txt b:foodprp.txt > compare.txt
```

For more information about redirection characters, see Chapter 7, "Advanced Command Techniques." For more information about the **fc** command, see Chapter 14, "Commands."

Viewing and Changing File Attributes

Every file can have four qualities associated with it. These qualities are called file attributes.

- The archive attribute (**a**) is used with the **backup**, **xcopy**, and other commands to control which files are backed up. For information about backing up files, see Chapter 6, "Managing Disks."
- The read-only attribute (**r**) prevents a file from being changed or deleted. When a file has this attribute, you can look at the file but you cannot delete it or change its contents. This attribute is discussed later in this section.
- The hidden attribute (**h**) prevents MS-DOS from displaying a file in a directory list. The file remains in a directory, but you cannot use the file unless you know its filename. This attribute is useful if you are working on confidential files. For more information about hidden attributes, see the **attrib** command in Chapter 14, "Commands."
- The system attribute (**s**) designates a file as a system file. Files with the system attribute are not shown in directory listings. For more information about system attributes, see the **attrib** command in Chapter 14, "Commands."

Viewing File Attributes

In Brief

To view the attributes of a file, use the **attrib** command, as in the following example:

```
attrib outgo.xls
```

MS-DOS displays the attributes of the file and the filename.

To see a file's attributes, type the **attrib** command followed by the filename. For example, you would use the following command to view the attributes of the CONFIG.SYS file on a disk in drive A:

```
attrib a:config.sys
```

MS-DOS displays up to four attributes in front of the filename. For example, if the CONFIG.SYS file has the archive and read-only attributes, MS-DOS displays the following:

```
a r config.sys
```

You can see the attributes for a group of files by using wildcards with the **attrib** command. You would use this command to view the attributes of all files in the root directory of drive C:

```
attrib c:\*.*
```

Shell ▶ **To view the attributes of a file:**

1. From the file list, select the file for which you want to view attributes.
2. From the Options menu, choose Show Information.

   MS-DOS displays the Show Information dialog box, which lists the file's attributes along with other information about the file.
3. When you are finished viewing the attributes, choose the OK button.

NOTE If you are using the All Files view in MS-DOS Shell, you can view the attributes of a file by selecting the file.

Changing a File Attribute

In Brief

To assign an attribute to a file, use the **attrib** command along with the attribute letter and a plus sign (+). To remove an attribute from a file, use the **attrib** command along with the attribute letter and a minus sign (-). For example, use the following command to make the OUTGO.XLS file read-only:

```
attrib +r outgo.xls
```

If you want to be sure that no one alters the contents of a file, assign a read-only attribute to the file. For example, to assign a read-only attribute to a file named GEMINI.XLS on a disk in drive B, use the following command:

```
attrib +r b:gemini.xls
```

If you decide later that you want to change the file, you can remove the read-only attribute by using this command:

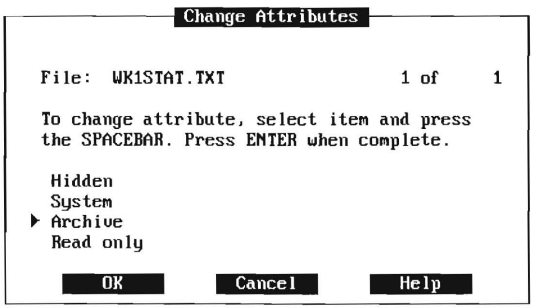
```
attrib -r b:gemini.xls
```

You can use wildcards to change the read-only attribute in groups of files.

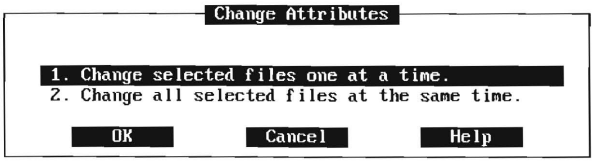
For information about other file attributes, see the **attrib** command in Chapter 14, "Commands."

Shell ▶ To change the attributes of a file:

1. From the file list, select the file or files you want to change.
2. From the File menu, choose Change Attributes. If you selected one file, this Change Attribute dialog box appears:



If you selected more than one file, this Change Attribute dialog box appears:



If you selected more than one file, choose option 1 to change the attributes of each selected file one at a time; choose option 2 to change the attributes of all the files at once.

3. Select the attribute you want to change or assign to the file or files listed in the box. A mark appears next to the attribute to show that it is assigned.

   To cancel a selection, choose it again.

4. Choose the OK button.

Finding Text Within a File

In Brief

To view the lines of a file that contain specified text, use the **find** command, as in the following example:

```
find "Error" output.txt
```

To view only the number of lines that contain the text you specify, include the **/c** switch:

```
find "Error" output.txt /c
```

The capitalization and spacing of the text you specify must be identical to the text in the file.

If you want to search one or more files for specified text, you can use the **find** command. For example, if your personal phone book is in the PHONE.TXT file, you can use the following command to view all lines of the file that contain the text "Area Code: 206":

```
find "Area Code: 206" phone.txt
```

MS-DOS searches the PHONE.TXT file and displays each line that includes the text "Area Code: 206". You must enclose the search text in quotation marks. MS-DOS finds only text that exactly matches the characters you specify, including capitalization and spacing. If the text in the file has formatting codes (for example, if the words "Area Code" are underlined), MS-DOS cannot find the specified text.

You cannot use wildcards to search more than one file, but you can include in the **find** command all the files you want to search. For example, the following command searches the ADDR.TXT file in addition to the PHONE.TXT file:

```
find "Area Code: 206" phone.txt addr.txt
```

If you only want to know how many lines of the file contain the specified text, use the **/c** switch with the **find** command. For example, use the following command to find out how many lines in the ADDR.TXT file contain the text "Bellerose":

```
find "Bellerose" addr.txt /c
```

If you want the results to be stored in a file rather than displayed on your screen, use the greater-than sign (>) to redirect the output. For example, the following command stores the results of the **find** command in the NWEST.NUM file:

```
find "Area Code: 206" phone.txt >nwest.num
```

For information about redirection characters, see Chapter 7, "Advanced Command Techniques."

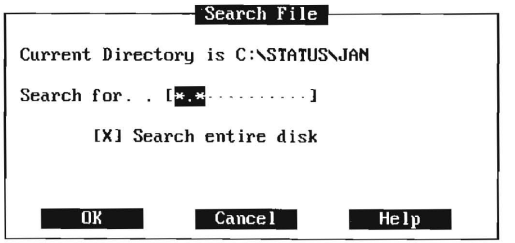
Searching for Files by Using MS-DOS Shell

You can locate files by using the Search command in MS-DOS Shell. This command searches the selected directory or the entire current disk for the file or files you specify. You can use any of the commands on the File menu to work with the files you locate.

Shell ▶ **To search for a file:**

1. From the File menu, choose Search. (A file or directory must be selected in order for the Search command to be available.)

   The Search File dialog box appears.



2. Type the name of the file you want to find.

   You can search for a single file by typing its name, or you can use wildcards to search for files with similar names.

3. MS-DOS searches the entire disk. To search only the current directory and its subdirectories, make sure the Search Entire Disk check box is cleared.

4. Choose the OK button.

MS-DOS begins searching at the root directory (or, if you canceled Search Entire Disk, it begins at the current directory). It searches through all branches of the directory tree on the current disk. It then displays a window called Search Results that lists all the files and directories found.

You can select files from this list and work with them by using the commands on the File menu.

Getting Information by Using MS-DOS Shell

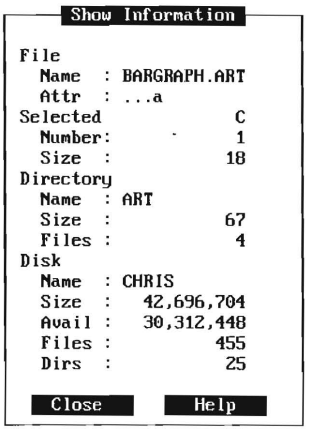
Using MS-DOS Shell, you can view information about a file, its directory, and the disk it is on.

NOTE If you are using the All Files view, this information is always displayed for the currently selected file.

Shell ▶ **To get information about a file:**

1. From the file list, select the file you want information about.
2. From the Options menu, choose Show Information.

   The Show Information dialog box appears:



This dialog box contains the following sections:

File	Shows the filename and attributes of the file you selected. The attributes can be (**a**) archive, (**r**) read-only, (**h**) hidden, and (**s**) system. For more information about file attributes, see "Viewing and Changing File Attributes" earlier in this chapter, or see the **attrib** command in Chapter 14, "Commands."

Selected	Shows the number of selected files on the current disk and their combined total size. If you have worked with two disks, the Selected section has two columns: one for the most recently selected disk and one for the previously selected disk.
Directory	Lists the name, size, and number of files in the directory that contains the selected file.
Disk	Lists the name, size, and number of files and directories, in addition to the amount of available space, on the disk that contains the selected file.

3. When you finish viewing the information, choose the OK button.

Chapter 5

Working with Directories

5

A disk is like a file cabinet; it has "folders" that contain groups of files. These folders, called directories, help you organize your files.

When a folder in a file cabinet contains so much information that you can no longer easily find what you want, you subdivide it. In the same way, when a directory on your disk becomes too large, you can divide it into subdirectories.

Understanding Directories

Directories are most important when you use a hard disk. If you use only floppy disks, you can probably keep files organized by putting them on different disks. With a hard disk, which usually can store much more information than a floppy disk, it becomes necessary to organize files into categories so you can find them more easily.

The Directory Tree

Every disk has at least one directory. When you format a floppy or hard disk, MS-DOS creates a directory where all other files and directories will be stored. This is called the *root directory*. You can create subdirectories of the root directory to organize files. For example, you could put your spreadsheet files in one directory and your text files in another. Directories and subdirectories form a structure called a *directory tree*, as in the following example:

```
[A:\] tree
Directory PATH listing for Volume SPRDSHT_TXT
Volume Serial Number is 373E-17D0
A:.
├─TEXT
└─SPREADSH
```

You can create subdirectories within subdirectories to further organize your files. For example, suppose you use a drawing program for work, school, and personal projects. To keep your files organized, you can create three subdirectories in your program directory. You can put your work files in a WORK directory, your personal files in a HOME directory, and your school files in a SCHOOL directory. Your directory structure would resemble the following:

```
[C:\] tree
Directory PATH listing for Volume CHRIS
Volume Serial Number is 157A-6C23
C:.
├──DOS
└──ART
    ├──WORK
    ├──HOME
    └──SCHOOL
```

You can continue to add directories at any level of the structure, as long as there are no more than 512 files and directories in the root directory of your hard disk (a root directory on a floppy disk can hold fewer files and directories). Other directories can contain as many files and directories as needed. However, MS-DOS runs more slowly if there are more than 150 or so files and subdirectories in one directory.

Strictly speaking, all directories other than the root directory are subdirectories. However, it is common to use the term *directory*. In this guide, the term *subdirectory* is used only to emphasize the relationship between two directories. A subdirectory is sometimes called a *child directory*, and a directory that contains subdirectories is sometimes called a *parent directory*.

Directory Names

Except for the root directory, which is always represented by a backslash (\), each directory has a name. A directory can also have an extension. Follow these rules when choosing a directory name and extension:

- The name must contain one to eight characters.
- There can be an extension of up to three characters, separated from the directory name by a period.

- The name and extension can contain the letters A through Z, the numbers zero through nine, and the following special characters: underscore (_), caret (^), dollar sign ($), tilde (~), exclamation point (!), number sign (#), percent sign (%), ampersand (&), hyphen (-), braces ({ }), and parentheses ( ). No other special character is acceptable.

- The name cannot contain spaces, commas, periods, or backslashes.

- Two subdirectories in the same directory cannot have the same name. For example, the WORK directory cannot have two subdirectories named FILES. However, subdirectories of different directories can have the same name. For example, you can have a subdirectory named FILES in both the HOME directory and the SCHOOL directory.

The name can contain extended characters. However, if you use an extended character, it is recommended that you use code page 850. If you use code page 437, support for extended characters is limited. For information about extended characters, see the Appendix, "Keyboard Layouts and Code Pages."

The current directory can be referred to by its name or by a period. The parent directory of the current directory can be referred to by its name or by a double period (..). When you use the **dir** command to view the files and directories in a directory (other than the root directory), you'll see these symbols listed, representing the current and parent directories.

Paths

The *path* specifies the location of a file within the directory tree. You can think of it as the route MS-DOS must travel, starting at the root directory, to get to files in another directory. For example, suppose that drive C has this directory tree:

```
[C:\] tree
Directory PATH listing for Volume CHRIS
Volume Serial Number is 157A-6C23
C:.
├──DOS
└──ART
    ├──WORK
    ├──HOME
    └──SCHOOL
```

To get to files in the HOME directory, MS-DOS must go through the following directories: root (\), ART, and HOME.

In MS-DOS commands, you designate the path in the following way:

```
\art\home
```

This is the path of the HOME directory. The first backslash represents the root directory; the second backslash separates the HOME directory from its parent directory, ART.

If you want MS-DOS to find the HOME directory, you type the path of the directory. To specify the FIG1.MSP file in the \ART\HOME directory, you add another backslash and the filename to the path:

```
\art\home\fig1.msp
```

There may be more than one file called FIG1.MSP in other directories, and there may be other directories called \ART\HOME on other disks. For this reason, to fully distinguish one file from all other files, you must add a drive letter to the path and filename. For example, the full path designation for the FIG1.MSP file in the \ART\HOME directory on drive C is:

```
c:\art\home\fig1.msp
```

MS-DOS recognizes paths of up to 66 characters (including the drive letter and colon).

The Current Drive

Unless you indicate otherwise, it is assumed that you want to use the directory tree on the drive you are currently using. The letter of the current drive is usually shown as part of the command prompt. For example, if you are currently using the root directory of drive A, and you want to delete the A:\FIG1.MSP file, you wouldn't need to type the letter of the current drive. You would type the following:

```
del fig1.msp
```

There can be only one current drive at a time. To work with files on a drive that isn't current, type the other drive letter followed by a colon (:) and press ENTER to make it the current drive.

The Current Directory

The directory you are working in is the current directory for that drive. MS-DOS can display the path of the current directory as part of the command prompt. If you want to perform some operation on a file, and you are currently using the directory

that the file is in, you do not need to type the path of the current directory. For example, if C is the current drive and \ART\HOME is the current directory, you can delete the C:\ART\HOME\FIG1.MSP file by typing this command:

```
del fig1.msp
```

Because C is the current drive and \ART\HOME is the current directory, you do not need to type them.

If you are working with two drives, each of them has a current directory. For example, suppose C is the current drive and \ART\HOME is the current directory. On your disk in drive A, suppose the \FIGS directory is the current directory. You can type the following command to copy the FIG2.MSP file from A:\FIGS to C:\ART\HOME:

```
copy a:fig2.msp c:
```

Unless you specify a different path, you work in the current directory on each drive. When you start your system, all current directories are the root directories of the drives in your system. The current directory on a floppy disk drive changes to the root directory if you change disks.

To work with files in a directory that is not current, you have two options:

- You can type the path of the other directory.
- You can make the other directory current by using the **cd** (change directory) command. The **cd** command is described in "Moving Between Directories" later in this chapter.

If you are working with program files that aren't in the current directory, you can include the path of the other directory in the **path** command. For information about the **path** command, see "Specifying a Search Path" later in this chapter.

If you want to type the path of another directory, you include the part of the path that is different from the path of the current directory. For example, if the current directory is \ART, you can delete the \ART\HOME\FIG1.MSP file by typing the following:

```
del home\fig1.msp
```

You do not need to type the complete path, because the file you want to delete is in a subdirectory of the current directory. If the file is not in a subdirectory of the current directory, you must either type its entire path or specify the path in relation

to the current directory. For example, if the current directory is \ART, to delete \TEXT\REPORTS\STOCK.TXT, you must type one of the following commands:

```
del \text\reports\stock.txt
```

Or

```
del ..\text\reports\stock.txt
```

In the second command, the double periods (..) represent the parent of the current directory. In this case, the root directory is the parent of ART, and TEXT is a subdirectory of the root directory.

Changing the Command Prompt

You can use the **prompt** command to change the way the command prompt looks. Unless you indicate otherwise, MS-DOS displays the letter of the current drive followed by a greater-than sign (>) as the command prompt. For example, the following prompt tells you that the current drive is A:

```
A>
```

You can use various parameters with the **prompt** command to change your command prompt. For example, you may want the current directory, in addition to the current drive, to be displayed as the command prompt. Knowing which directory is current can be useful when you have numerous subdirectories. The following command creates a command prompt that shows both the letter of the current drive and the current directory name:

```
prompt $p
```

If the current directory is C:\PBRUSH, MS-DOS displays this prompt:

```
C:\PBRUSH
```

The following command creates a command prompt that indicates the current drive and directory and has a greater-than sign (>) to separate the prompt from any commands you type:

```
prompt $p$g
```

If the current directory is C:\PBRUSH, MS-DOS displays this prompt:

```
C:\PBRUSH>
```

NOTE If you use **$p** to create a prompt that displays the current drive and directory, MS-DOS checks the directory after every command. This may cause a delay if your current drive contains a floppy disk. Type **prompt** and press ENTER to return the command prompt to its original form—the greater-than sign (>) preceded by the current drive letter.

You can include the **prompt** command in the AUTOEXEC.BAT file so that every time you start your system, MS-DOS sets the command prompt. If your prompt already contains more than the current drive letter when you start your system, there is already a **prompt** command in your AUTOEXEC.BAT file. For more information about the AUTOEXEC.BAT file, see Chapter 11, "Customizing Your System."

For information about other parameters you can use, see the **prompt** command in Chapter 14, "Commands."

Viewing Directories

To see a list of files in any directory on any disk, use the **dir** command. The simplest form of this command displays a list of files in the current directory. By adding to the command, you can do the following:

- View only certain filenames in a directory
- View a directory listing, one screen at a time
- Designate the way files are displayed in a directory listing

This section describes how to view directories by using the command line. For information about viewing directories in MS-DOS Shell, see Chapter 3, "MS-DOS Shell Basics."

Viewing Whole Directories

In Brief

To view the contents of a directory, use the **dir** command. For example, you could use the following command to view the contents of the C:\WORK directory:

```
dir c:\work
```

The **dir** command (without parameters) lists the contents of the current directory. For example, if C:\ is your current directory, you can use the following command to view its contents:

```
dir
```

MS-DOS displays a listing similar to the following:

```
[C:\] dir

 Volume in drive C is CHRIS
 Volume Serial Number is 157A-6C23
 Directory of C:\

COMMAND  COM     46246 11-21-90   4:06a
DOS          <DIR>     11-26-90   6:31a
ART          <DIR>     11-26-90   6:34a
MEETING      <DIR>     11-26-90   6:34a
WORK         <DIR>     11-26-90   4:21p
CONFIG   SYS       289 11-26-90   1:33p
AUTOEXEC BAT       598 11-26-90   2:15p
STATUS       <DIR>     11-28-90   2:55p
MOUSE    SYS     34581 11-07-90   4:21p
EXPENSE      <DIR>     11-26-90   6:36a
PROSE        <DIR>     11-26-90   6:36a
SPREADSH     <DIR>     11-26-90   6:38a
TEXT         <DIR>     11-26-90   6:38a
CONFIG   OLD       260 11-19-90   5:04p
       14 file(s)      81974 bytes
                    36835328 bytes free
```

If you add the path of a directory to the command, MS-DOS displays the contents of the specified directory rather than the current directory. Regardless of which directory is current, you would use the following command to view a list of files in the root directory of a disk in drive B:

```
dir b:\
```

To view the filenames listed in a subdirectory, you can specify the path relative to the current directory, or you can specify the entire path. For example, if C:\YEAR is your current directory and you want to view the contents of the C:\YEAR\JAN directory, you could type the following command:

```
dir jan
```

Viewing Groups of Filenames

In Brief

To view a group of filenames in a directory listing, include wildcards with the **dir** command. For example, you would use the following command to view a list of all files that have a .COM extension in the current directory:

```
dir *.com
```

Unless you specify otherwise, MS-DOS displays all filenames and subdirectory names in a directory. To view only certain filenames in a directory, you can use wildcards. For example, to see a list of files that have a .DOC extension in the root directory of a disk in drive B, type the following command:

```
dir b:\*.doc
```

To view the filenames beginning with JAN in a subdirectory called YEAR on drive C, type this command:

```
dir c:\year\jan*.*
```

The filename is separated from the directory name by a backslash.

For more information about wildcards, see Chapter 4, "Working with Files."

Arranging the Directory Display

In Brief

To view the contents of a directory one screen at a time, use the **/p** switch, as in the following command:

```
dir c:\year /p
```

To view an abbreviated directory, with only directory names and filenames listed, use the **/w** switch:

```
dir c:\year /w
```

Often a directory contains too many filenames to be displayed on one screen. To view a directory one screen at a time, use the **/p** switch, as in the following command:

```
dir /p
```

After displaying the first screen of files, MS-DOS pauses. To continue viewing the contents of the directory, press any key except the PAUSE key.

If you include the **/w** switch with the **dir** command, MS-DOS displays only the directory names and filenames, instead of all the additional information that is displayed when you use the **dir** command alone. Directory names are enclosed in brackets; for example, [DOS]. Using this switch, you often can view all the files on a single screen. For example, you would use the following command to view the directory names and filenames in the root directory of drive C:

```
dir c:\ /w
```

MS-DOS displays up to five columns of names.

Sorting a Directory Listing

In Brief

To sort a directory listing by filename (**n**), extension (**e**), date (**d**), or file size (**s**), use the **/o** switch. For example, the following command alphabetizes a directory listing by filename extension:

```
dir c:\temp /oe
```

This command alphabetizes a directory listing in reverse alphabetic order (from Z to A):

```
dir c:\temp /o-e
```

MS-DOS sorts a directory listing when you include the **/o** switch. The following list shows the various ways of using the **/o** switch to sort a directory listing:

/on	Alphabetizes the directory by name
/o-n	Reverse-alphabetizes the directory by name (Z to A)
/oe	Alphabetizes the directory by extension
/o-e	Reverse-alphabetizes the directory by extension (Z to A)
/od	Sorts the directory by date (oldest first)
/o-d	Sorts the directory by date (newest first)
/os	Orders the directory by file size (smallest first)
/o-s	Orders the directory by file size (largest first)

For example, the following command displays the contents of the current directory of a disk in drive B, largest file first:

```
dir b: /o-s
```

You can also combine switches to get the results you want. For example, the following command alphabetizes the listing of the root directory on the C drive and displays the filenames in five columns:

```
dir c:\ /on /w
```

Viewing All Directories on a Disk

In Brief

To view the organization of a directory and its subdirectories, use the **tree** command. For example, the following command displays the relationship between the C:\TEMP directory and its subdirectories:

```
tree c:\temp
```

In the following command, the **/f** switch is used to include filenames:

```
tree /f
```

The easiest way to see the relationship between directories on a disk, or between a directory and its subdirectories, is to use the **tree** command. You can also see subdirectories by using the **dir** command, but the **tree** command gives you more information.

The **dir** command usually lists only the first level of subdirectories, whereas the **tree** command lists all levels. The **dir** command displays subdirectory names along with filenames in a list, whereas the **tree** command displays the relationship between a directory and its subdirectories graphically, showing how each level fits together.

For example, to see the subdirectories in the current directory, type the following command:

```
tree
```

MS-DOS displays a directory tree similar to this one:

```
[C:\] tree
Directory PATH listing for Volume CHRIS
Volume Serial Number is 157A-6C23
C:.
├──DOS
├──ART
│   ├──HOME
│   └──SCHOOL
├──MEETING
├──WORK
├──EXPENSE
│   └──BUSINESS
├──PROSE
├──SPREADSH
│   └──BUDGET
├──STATUS
│   ├──JAN
│   └──FEB
└──TEXT
    └──LETTERS
```

To see the tree structure, starting at the root directory of the current drive, type this command:

```
tree \
```

If you want to see the filenames in each directory of the tree, include the **/f** switch. For example, the following command displays all the directory names and filenames on drive C:

```
tree c:\ /f ¦ more
```

For information about the **more** command, see Chapter 7, "Advanced Command Techniques."

Creating Directories

In Brief

To create a directory, use the **md** (**mkdir**) command. For example, suppose the \TAX\QUARTER directory is the current directory. The following command creates a new subdirectory called EST:

```
md est
```

If the new directory is a subdirectory of the current directory, you only need to type the new part of the path, as in the preceding example.

When you have a group of related files (such as specialized files that you use with one program, or files from a specific project), you may want to store them in their own directory. To create a directory, you can use the **md** (make directory) command. For example, suppose the current directory is the root (\). To create a subdirectory called REPORTS, you would type the following command:

```
md reports
```

You can use either **md** or its longer form, **mkdir**, with the same results.

The **md** command makes a subdirectory in the current directory, unless you specify otherwise. For example, suppose the current directory is \WORK. To make a directory called \WORK\HOME, you can type the following:

```
md home
```

In this example, MS-DOS makes a subdirectory called HOME in the \WORK directory. If you are not making a subdirectory of the current directory, you must type the entire path of the new directory or specify the path relative to the current directory. For example, suppose the current directory is \TAX. To make the subdirectory \WORK\HOME, you would type one of the following commands:

```
md \work\home
```

Or

```
md ..\work\home
```

If you include a drive letter with the **md** command, you can create a directory on a disk that is not current. For example, suppose you want to copy files in the current directory, C:\WORK\HOME, to a subdirectory \HOME on a disk in drive A. To create this directory on the disk in drive A without leaving the current directory, you would type this command:

```
md a:\home
```

Shell ▶ **To create a subdirectory of the currently selected directory:**

1. Select the directory for which you want to create a new subdirectory.
2. From the File menu, choose Create Directory.

   The Create Directory dialog box appears. The currently selected directory is shown as the parent directory.
3. Type the name of the new directory. The name can have up to eight characters. The extension, if any, can have up to three characters.
4. Choose the OK button.

You can create directories at any level in the directory tree. Any directories you create are subdirectories of the currently selected directory.

Moving Between Directories

The disk in each drive of your system has a current directory. If you have two floppy disks and one hard disk, there are three current directories.

If a disk has no subdirectories, the root directory is always the current directory for that disk. If a disk has subdirectories, you can use the **cd** (change directory) command to move from one directory to another.

Changing Directories

In Brief

To move to a different directory on the current drive, use the **cd** (**chdir**) command. For example, the following command changes the current directory to \OFFICE\REPORTS\FINANCE:

```
cd \office\reports\finance
```

You can also use two periods (..) after the **cd** command to change to the parent directory, as follows:

```
cd ..
```

If the current directory is the root (\), you can move to the \WORK directory and make it current by typing the following command:

```
cd work
```

You can use either **cd** or its longer form, **chdir**.

When you use the **cd** command, you change to a subdirectory of the current directory, unless you specify otherwise. For example, if the current directory is \WORK, to change to \WORK\HOME, you type the last subdirectory in the path:

```
cd home
```

If the current directory is \OFFICE, to change to \WORK\HOME, you must type the entire path:

```
cd \work\home
```

To change to the parent of the current directory (the directory one level closer to the root), you can use two periods (..) in the command. For example, if the current directory is \WORK\HOME, you can change to \WORK by using this command:

```
cd ..
```

Regardless of which directory is current, you can change to the root directory of the current drive by typing this command:

```
cd \
```

You cannot use the **cd** command to change the current drive, but you can use it to change the current directory on a drive that is not current. For example, suppose the current drive is A. To change the current directory on drive C to \WORK without making drive C current, type the following command:

```
cd c:\work
```

The current directory on drive C is used if you specify drive C without a directory. For example, suppose A is the current drive and \WORK is the current directory on drive C. You would type the following command to copy all files from the current directory on drive A to C:\WORK:

```
copy *.* c:
```

The current directory on any drive is the root, unless you change it.

Deleting Directories

In Brief

To delete a directory, use the **rd** (**rmdir**) command, as in the following example:

```
rd \office\reports\finance
```

MS-DOS removes the FINANCE subdirectory from the \OFFICE\REPORTS directory on the current drive. The directory you remove cannot contain any files or subdirectories.

If you no longer need a directory, you can remove it by using the **rd** command. In order for you to remove a directory, it must be empty and it cannot be the current directory. For example, before you can remove a directory called \WORK, which does not contain any subdirectories, you must delete its contents. To do so, you can type the following command:

```
del \work\*.*
```

You receive the following message: "All files in directory will be deleted! Are you sure (Y/N)?" Type **y** to delete the files; type **n** to cancel the command.

You can then type the following command to remove the directory:

```
rd work
```

You can use either **rd** or its longer form, **rmdir**.

MS-DOS removes a subdirectory of the current directory, unless you specify otherwise. For example, if the current directory is \WORK, and \HOME is a subdirectory of \WORK, you can remove \HOME by typing the following:

```
rd home
```

If the current directory is \OFFICE, to remove \HOME you must type the entire path or the path relative to the current directory:

```
rd \work\home
```

Or

```
rd ..\work\home
```

If you include a drive letter with the **rd** command, you can remove a directory from a drive that is not current. For example, to remove the A:\HOME directory while the current drive is C, type the following command:

```
rd a:\home
```

If the directory you want to delete contains files or subdirectories, you must first delete them.

NOTE If MS-DOS does not delete a directory after you have deleted all files and subdirectories in it, there may be hidden or read-only files in the directory. For information about changing the attributes of hidden or read-only files, see the **attrib** command in Chapter 14, "Commands."

Shell ▶ **To delete a directory or subdirectory:**

1. Select the directory.
2. Make sure the directory does not contain any files or subdirectories.
3. From the File menu, choose Delete.

   You are asked to confirm the deletion. (You can suppress the confirmation message; see Chapter 3, "MS-DOS Shell Basics.")
4. Choose the Yes button.

If you attempt to delete a directory that contains files or subdirectories, MS-DOS displays the Deletion Error dialog box with the following message: "You cannot delete a non-empty directory; delete all files and subdirectories in this directory first." If this message appears, delete the files and subdirectories in the directory, and try again.

Copying Directories

To copy a directory and its subdirectories, you can use the **xcopy** command. The **xcopy** command is similar to the **copy** command. Both commands copy files from one directory or drive to another. The **copy** command works with a single file or a group of files, whereas the **xcopy** command works with a single directory or a group of directories. Both commands create new files in the destination directory, but only the **xcopy** command can also create new subdirectories.

Copying All Files in a Directory

In Brief

To copy a single directory (without subdirectories), use the **xcopy** command without switches. For example, the following command copies all files in the C:\NEW\REPORTS\FINANCE directory to the \FINANCE directory on a disk in drive A:

```
xcopy c:\new\reports\finance a:\finance
```

Because the **xcopy** command copies all files in a directory, you do not need to use wildcards. For example, the following **xcopy** command copies all files in the current directory from drive A to drive B:

```
xcopy a: b:
```

While MS-DOS prepares to copy the files, it displays a "Reading source file(s)" message. As with the **copy** command, MS-DOS displays the names of the files it copies and indicates how many files were copied when the operation is complete.

Shell ▶ **To copy all files in one directory to another directory:**

1. Select the directory containing the files you want to copy. Make the file list active by selecting any filename in the list.
2. From the File menu, choose Select All.
3. From the File menu, choose Copy.

   The Copy File dialog box appears.
4. In the To box, type the path of the directory you want to copy files to. Any text you type replaces what's in the To box.
5. Choose the OK button.

Creating Directories as You Copy Files

In Brief

If the destination path in an **xcopy** command does not exist, MS-DOS creates it. For example, the following command copies all files from the root directory of a disk in drive A to the C:\TMP directory:

```
xcopy a:\ c:\tmp
```

If the directory does not exist, there is a prompt asking you whether the name specified is a file or directory. (To prevent MS-DOS from prompting you, add a backslash at the end of the directory name.)

You can use the **xcopy** command to create a directory as you copy files. For example, suppose you want to copy all files in the root directory of a disk in drive A to drive C. Using the following **xcopy** command, you can put the files in a directory called \NEWFILE:

```
xcopy a:\ c:\newfile
```

If the \NEWFILE directory does not exist on drive C, there is a prompt asking you whether it is a file or directory. Type **d** for directory. MS-DOS then creates it as a subdirectory of the root directory. (To prevent MS-DOS from prompting you, add a backslash at the end of the directory name.) MS-DOS copies the files from the disk in drive A to the \NEWFILE directory. In this example, only the files in the root directory of the disk in drive A are copied. If there are subdirectories on the disk in drive A, MS-DOS does not copy them.

If you do not type a path, MS-DOS copies the files to the current directory.

Copying Subdirectories

In Brief

To reproduce an entire directory structure in another directory or on another disk, use the **/s** and **/e** switches. For example, the following command recreates on a disk in drive B the directory structure and files in C:\REPORTS:

```
xcopy c:\reports b:\ /s /e
```

To copy files in a directory along with any subdirectories that contain files, add the **/s** switch to the **xcopy** command. For example, suppose a disk in drive A contains the following subdirectories: SCHOOL, WORK, and HOME. The

following command copies the files in the root directory of drive A, including the three subdirectories and all their files, to the \MEMOS directory on drive C:

```
xcopy a:\ c:\memos /s
```

The backslash (\) after **a:** indicates that MS-DOS should start at the root directory. When the **/s** is added, every file in every subdirectory that contains files is copied. MS-DOS copies files from the root directory of drive A to C:\MEMOS, from A:\SCHOOL to C:\MEMOS\SCHOOL, from A:\WORK to C:\MEMOS\WORK, and from A:\HOME to C:\MEMOS\HOME. If any of the directories do not exist on drive C, MS-DOS creates them. In this example, empty subdirectories on drive A are not copied.

To copy an empty directory, use the **/e** switch with the **/s** switch. For example, suppose a disk in drive A has an empty subdirectory called MISC, in addition to the three subdirectories just mentioned. You could type the following command to copy all subdirectories, including the empty subdirectory:

```
xcopy a:\ c:\memos /s /e
```

You can use the **/s** switch without the **/e** switch, but you cannot use the **/e** switch without the **/s** switch.

Renaming Directories

In Brief

To rename a directory, use the **xcopy**, **del**, and **rd** commands. For example, the following sequence of commands renames the \OPS\STATS directory as \OPS\FIGURES:

```
xcopy \ops\stats \ops\figures

del \ops\stats\*.*

rd \ops\stats
```

If the original directory contains subdirectories, add the **/s** and **/e** switches to the **xcopy** command, and delete each subdirectory separately.

The **rename** command, which you use to rename files, cannot be used to rename directories. Instead, you use a combination of the **xcopy**, **del**, and **rd** commands to rename a directory.

▶ To rename a directory:

1. Using the **xcopy** command, copy the contents of the directory to a directory with the new name.
2. Delete the contents of the original directory.
3. Delete the original directory.

For example, suppose you want to rename the C:\TEMP directory as C:\LETTERS. The first step is to copy the contents of the directory to the new directory. You can type the following **xcopy** command:

```
xcopy c:\temp c:\letters
```

The following message appears: "Does LETTERS specify a filename or directory name on the target (F=file, D=directory)?" After you type **d**, MS-DOS creates a subdirectory (called LETTERS) in the root directory on drive C, and copies all the files from C:\TEMP into it.

The second step is to delete the files in C:\TEMP. To do this, type the following **del** command:

```
del c:\temp\*.*
```

MS-DOS prompts you to confirm deletion of all the files in this directory. If you are sure the files were successfully copied to C:\LETTERS, type **y**. If you have any doubt, type **n** and use the **dir** command to double-check.

Once the C:\TEMP directory is empty, the final step is to delete it by typing the following **rd** command:

```
rd c:\temp
```

Shell ▶ **To change the name of a directory or subdirectory:**

1. Select the directory you want to rename.
2. From the File menu, choose Rename.

   The Rename Directory dialog box appears.
3. Type the new name for the directory.
4. Choose the OK button.

Updating Directories

Sometimes you may want two directories to contain the same files. For example, when you are making backup copies of files and directories, the primary directory contains the files you are working on and a secondary directory or disk contains the backup or most recent version of the files. To keep the secondary directory current, you can use the **replace** command.

Replacing Outdated Files

In Brief

To replace files in a destination directory that are older than the corresponding files in a source directory, use the **replace** command with the **/u** switch, as in the following command:

```
replace c:\home\*.* a: /u
```

MS-DOS replaces the files in the root directory of the disk in drive A with the more recent versions in C:\HOME.

Suppose a directory called C:\OPS\STATS contains files you periodically update. To keep a backup copy of these reports, you can copy them to a floppy disk and periodically update them by using the **replace** command with the **/u** switch, as follows:

```
replace c:\ops\stats\*.* a: /u
```

MS-DOS compares the files in C:\OPS\STATS with the files on the disk in drive A. If a file on the disk in drive A has a more recent version on drive C, MS-DOS replaces the older version. Using the **replace** command with the **/u** switch does not add any new files to the backup disk; it only updates those already there.

Adding New Files

In Brief

To add files to the destination directory that are currently only in the source directory, use the **replace** command with the **/a** switch. For example, the following command compares the files in C:\HOME with those on a disk in drive A. If there are any files in C:\HOME that are not on the disk in drive A, MS-DOS copies them to the disk.

```
replace c:\home\*.* a: /a
```

To add files to a backup disk, use the **replace** command with the **/a** switch rather than the **/u** switch. For example, the following command compares the files in C:\OPS\STATS with the files on a disk in drive A:

```
replace c:\ops\stats\*.* a: /a
```

If there are any files in the C:\OPS\STATS directory that are not on the disk in drive A, MS-DOS copies them to the disk.

Specifying a Search Path

In Brief

To specify a search path for program files, use the **path** command. For example, the following command specifies that MS-DOS is to search for files in the three directories listed, in addition to the current directory:

```
path \;c:\bin;c:\utilities
```

Each directory in the **path** command must be separated by a semicolon (;). The first backslash (\) indicates that the search should begin in the root directory of the current drive.

Unless you specify otherwise, MS-DOS looks for program files in the current directory only. You can direct MS-DOS to look for program files in other directories by specifying a *search path*. The search path indicates the route from directory to directory that MS-DOS follows when it searches for a file.

To run a program stored in a directory other than the current directory, you can first make its directory current by using the **cd** command. An alternative to this procedure is to include in the **path** command all the paths you use frequently. If MS-DOS cannot find a file in the current directory, it searches the other directories you have specified. The **path** command remains in effect until you restart or reset your system.

For example, suppose you frequently run programs stored in the C:\PBRUSH, C:\WORK, and C:\ACCTS directories. You can save time by including them in the following **path** command:

```
path c:\pbrush;c:\work;c:\accts
```

Each directory must be separated from the others by a semicolon (;). MS-DOS searches the directories in the order you have typed them (in this case, MS-DOS would search \WORK before \ACCTS). If you want your root directory on the current drive to be searched first, add it to the beginning of the list of directories:

```
path \;c:\pbrush;c:\work;c:\accts
```

The **path** command can include up to 127 characters (including the word *path*).

You can specify a search path by including a **path** command in your AUTOEXEC.BAT file. For information about your AUTOEXEC.BAT file, see Chapter 11, "Customizing Your System."

Chapter 6

Managing Disks

6

Disks provide long-term information storage. The information you save on disks remains intact until you delete it. In contrast, random-access memory (RAM) provides short-term information storage. The information stored in RAM is lost each time you turn off your computer.

Types of Disks

Disks store information on magnetic surfaces. In a floppy disk, the magnetic surface is a thin, flexible disk inside a protective plastic cover. A hard disk has two or more rigid disks stacked on top of each other in a sealed case. A hard disk is also called a *fixed disk* because it remains in your system. Once your hard disk is installed, it should not be removed unless it is damaged or you upgrade to a larger disk.

Information on disks is divided into tracks, somewhat like the grooves on a record. Each track is a concentric circle that can hold a certain amount of information. The more tracks a disk has, the more information it can hold. A hard disk holds more information than a floppy disk because it has more sides and more tracks per side.

Floppy disks vary in physical size and the amount of information they can hold. The following list shows the major kinds of floppy disks that MS-DOS can work with and the amount of information each can hold:

5.25-inch, single sided/double density	160K
5.25-inch, single sided/double density	180K
5.25-inch, double sided/double density	320K
5.25-inch, double sided/double density	360K
5.25-inch, double sided/quad density	1200K or 1.2 MB
3.5-inch, double sided/double density	720K
3.5-inch, double sided/quad density	1440K or 1.44 MB
3.5-inch, double sided/high density	2880K or 2.88 MB

Most floppy disks have labels indicating their type. You can also use the **dir** or **chkdsk** command to view information about the storage capacity of a formatted disk. For information about these commands, see Chapter 14, "Commands."

Bytes, Kilobytes, and Megabytes

File sizes are measured in *bytes*. One byte is the amount of space it takes to store a character. A kilobyte is 1024 bytes. In this guide, the word *kilobyte* is abbreviated as K.

A megabyte is 1024K (about a million bytes). In this guide, the word *megabyte* is abbreviated as MB. For example, if a disk can store about 1.2 million bytes of information, it is a 1.2-MB disk. The following terms are equivalent:

1.2 MB = 1228.8K = 1,258,291 bytes

Types of Disk Drives

Not all types of floppy disks are compatible with all types of floppy disk drives. In general, the disk must be formatted at a capacity less than or equal to the capacity of the drive in order for the disk and drive to be compatible. For example, if you have a quad-density 5.25-inch disk drive designed to work with 1.2 MB-floppy disks, you can use floppy disks formatted as 360K disks. However, if you have a

360K drive, you usually cannot use disks formatted as 1.2-MB disks. If you are unsure whether a disk works with a certain drive, you can try using the disk by inserting it in the disk drive and using the **dir** command. If the disk and drive are incompatible or if the disk is unformatted, MS-DOS displays a "General failure error" message.

MS-DOS adjusts its operations to work with the type of disk drive you are using. When using some commands, you must add a switch if your disk drive and disk do not have the same capacity.

Formatting Disks

Before you can use a disk, you must prepare it by using the **format** command. The disk may or may not have been previously formatted.

When you format a disk, MS-DOS performs a *safe format* by default. Because of the safe format, you can restore the disk to its condition before the format by using the **unformat** command, provided you have not stored files on the newly formatted disk.

You can add the **/u** switch to the **format** command to perform an unconditional format. The unconditional format destroys all information on the disk, which means you cannot restore the contents of the disk after the disk has been formatted.

If you mistakenly format a disk unconditionally, you may still be able to recover lost information, provided that you installed the Mirror program before using the **format** command. The Mirror program is described in the section that follows.

When you format a floppy or hard disk, MS-DOS reserves a small part of the disk for its tracking system. The tracking system consists of two parts: a *file allocation table* (which tracks the location of each file on the disk) and the *root directory* (which stores the name, size, creation date and time, and file attributes for the files on the disk).

A *sector* is the basic unit of storage on a disk. Each sector on a disk holds half of a kilobyte of information. When MS-DOS formats a disk, it marks defective sectors so that it will not store information there. When MS-DOS stores a file on a disk, it uses groups of sectors called *allocation units*. The number of sectors per allocation unit depends on the size of the disk.

If you are using a new hard disk, you must partition it before you can format it. While you are running the MS-DOS Setup program, you can partition and format

the hard disk. For information about setting up MS-DOS on a hard disk, see *Microsoft MS-DOS Getting Started.* You can also partition a new hard disk by using the Fdisk program. For information about Fdisk, see "Using Fdisk" later in this chapter.

CAUTION Because the **format** command destroys all information on a disk, it's a good idea to develop the habit of using the **dir** command before formatting a disk so that you do not destroy important files. MS-DOS displays a warning message if you attempt to format your hard disk. If you accidentally format your hard disk, you may be able to use the **unformat** command to recover its contents. For information, see "Unformatting a Disk" later in this chapter.

Safeguarding Information on a Disk

The Mirror program saves information about a disk. This information provides a safeguard in case you mistakenly formatted a disk by using the **/u** switch with the **format** command. The Mirror program is also useful for recovering data from a corrupted disk.

The **unformat** command (described later in this chapter) recovers data from disks. If you unconditionally formatted a disk or the disk is corrupted, **unformat** uses information saved by the Mirror program.

Because **unformat** restores the disk's system area to the condition it was in when you last used Mirror, you should save this information frequently for every hard disk drive in your system. To ensure that the information is saved each time you start your system, add a **mirror** command to your AUTOEXEC.BAT file.

To save information about the current disk, type the following command:

```
mirror
```

To save information about drive A, type the following command:

```
mirror a:
```

You can also use Mirror to save information about hard-disk partitions onto a floppy disk or network drive. If your hard disk is corrupted, **unformat** can use this saved information to restore the disk. For more information about the **mirror** command, see Chapter 14, "Commands."

Formatting a Disk

In Brief

To format a floppy or hard disk, use the **format** command. For example, the following command formats a floppy disk in drive A:

```
format a:
```

You must specify the drive that contains the disk you want to format.

MS-DOS performs a safe format by default. If you want to disable safe formatting, add the **/u** switch to the **format** command. The **/u** switch deletes all existing data on a disk, which means you cannot later use the **unformat** command to restore the contents of the disk.

When you use the **format** command with the **/u** switch to format a hard disk, the following message appears:

```
Warning, all data on non-removable disk drive C: will be lost!
Proceed with format (Y/N)?
```

Type **y** to proceed, or **n** to cancel the command.

Using the **/q** switch with the **format** command, you can perform a quick format on a previously formatted disk, which decreases the time MS-DOS requires to format it. Use the **/q** switch only if you haven't received read or write errors on the disk you are formatting.

As it formats the disk, MS-DOS displays the percentage of the disk that has been formatted. After the disk is formatted, you are prompted to give the disk a *volume label.* Type the name you want to give the disk, or press ENTER if you don't want a label.

MS-DOS then displays information about how the disk was formatted:

```
  1213952 bytes total disk space
  1213952 bytes available on disk

      512 bytes in each allocation unit
     2371 allocation units available on disk

Volume Serial Number is 382C-17F4
```

Bytes total disk space Indicates the storage capacity of the disk.

Bytes used by system Appears if you have transferred the MS-DOS system files to the disk. This line shows how much disk space is used by the three system files.

Bytes in bad sectors Indicates how much of the disk is unusable because of bad sectors. If there are no bad sectors, this line is omitted. If a floppy disk has any bad sectors, you should consider not storing important files or backup files on it. Most hard disks have a small number of bad sectors. In general, the portion of a hard disk taken up by bad sectors should be a small fraction of the total space available.

Bytes available on disk Indicates the total disk space minus the space taken up by the system files and any bad sectors. If the disk does not contain system files and there are no bad sectors, this number should be the same as the "bytes total disk space" number.

Bytes in each allocation unit and *allocation units available on disk* Indicate how MS-DOS has divided the available disk for file storage. If you multiply the two numbers on these lines, the result is the same as the "bytes available on disk" number.

Volume serial number Indicates the serial number assigned to the disk. This number does not change unless the disk is formatted again.

Following this information, you are prompted to format another disk. Type **y** to format another disk in the same drive with the same switches, or type **n** to return to the command prompt.

Specifying Disk Capacity

In Brief

To format a floppy disk that has less capacity than that supported by the drive, use the **/f** switch. For example, to format a 360K disk in a 1.2-MB, 5.25-inch drive, type the following:

```
format a: /f:360
```

Unless you state otherwise, it is assumed that the disk you want to format has the maximum capacity for the drive. For example, if you have a high-density (1.2 MB) 5.25-inch disk drive, it is assumed that any disk you format in the drive is also high density. To format a disk with a lower capacity, use the **/f:** switch.

For example, if drive A is a 1.2-MB, 5.25-inch drive, and you want to format a 360K disk in it, use the following command:

```
format a: /f:360
```

If drive B is a 1.44-MB, 3.5-inch drive, and you want to format a 720K disk in it, use this command:

```
format b: /f:720
```

Some of the newest disk drives are able to detect the capacity of a floppy disk. If you have this type of drive, you do not need to specify these switches.

NOTE Because of hardware differences, some 360K drives cannot reliably read disks formatted on a 1.2-MB drive when you use the **/f:360** switch.

Formatting a Floppy Disk by Using MS-DOS Shell

To format a floppy disk when running MS-DOS Shell, you use the **format** command in the Disk Utilities group. You use the same switches and some of the same formatting procedures as you would at the command prompt.

You can use switches to format a floppy disk with a smaller storage capacity than your floppy disk drive and to transfer system files to a disk.

Shell ▶ **To format a floppy disk:**

1. From the Disk Utilities group in the program list, choose Format.

   The Format dialog box appears.

2. To format a disk in drive A and use no switches, choose the OK button. To format a different drive or to specify parameters or switches, type the information in the Parameters box, and then choose the OK button.

   From this point on, MS-DOS displays the same messages and prompts in MS-DOS Shell as it does at the command prompt.

3. After formatting is complete, you are prompted to give the disk a volume label. Type a name if you want to label the disk. If you don't want to give the disk a label, press ENTER.

4. At the next prompt, type **y** if you want to format another disk, or type **n** if you want to return to MS-DOS Shell.

Shell ▶ **To perform a quick format of a floppy disk:**

1. From the Disk Utilities group in the program list, choose Quick Format.

   The Quick Format dialog box appears.

2. To format a disk in drive A, choose the OK button. To format a different drive, type the drive letter in the Parameters box, and then choose the OK button.

   From this point on, MS-DOS displays the same messages and prompts in MS-DOS Shell as it does at the command prompt.

3. After formatting is complete, you are prompted to give the disk a volume label. Type a name if you want to label the disk. If you don't want to give the disk a label, press ENTER.

4. At the next prompt, type **y** if you want to format another disk, or type **n** if you want to return to MS-DOS Shell.

Unformatting a Disk

In Brief

To recover as much information as possible from a hard or floppy disk that has been reformatted, use the **unformat** command, as in the following example:

```
unformat c:
```

You can restore a disk that has been reformatted by using the **unformat** command. **Unformat** works most reliably if the disk was safe-formatted (that is, if you used the **format** command without the **/u** switch), or if you installed the Mirror program before you formatted the disk. You can use **unformat** without having done either of these things, but there is a higher risk of losing data. **Unformat** is most effective if used immediately after a disk has been reformatted.

If the disk was safe formatted, **unformat** restores the disk to its condition at the time of the format. If the disk was not safe formatted but you did use the Mirror program, **unformat** attempts to restore the disk to the condition it was in when you last used Mirror.

To restore a disk that has been safe formatted, use the **unformat** command. For example, to restore a hard disk (drive C), use the following command:

```
unformat c:
```

You cannot restore a formatted disk if you use the **/u** switch with the **format** command. The **/u** switch performs an unconditional format (removes the safe formatting). You also cannot restore a floppy disk if you changed its storage capacity when you reformatted it. You need to use the **unformat** command immediately after you have formatted a disk. If you have saved anything on the disk between formatting and unformatting, you will probably lose some of the original information.

For more information about the **unformat** and **mirror** commands, see Chapter 14, "Commands."

Creating a System Disk

In Brief

To create a disk that can start your system, use the **format** or **sys** command.

To create a system disk during formatting, add the **/s** switch to the **format** command. For example, the following command formats the disk in drive A, then copies system files to the disk:

```
format a: /s
```

To make a disk that is already formatted a system disk, use the **sys** command. For example, the following command copies system files to a formatted disk in drive B:

```
sys b:
```

System disks contain three MS-DOS system files—IO.SYS, MSDOS.SYS, and COMMAND.COM. When you start your system, these three files are copied from the system disk to your system's random-access memory (RAM). The IO.SYS and MSDOS.SYS files are hidden files; you don't see them in directory listings unless you use the **/a** switch with the **dir** command. The COMMAND.COM file is usually in the root directory of every system disk.

If you have a hard disk, it is usually your main system disk. However, generally when you start your system, drive A is checked first. If there is a system disk in drive A, it starts your system.

You can copy the three system files to a disk as part of the **format** command or by using the **sys** command. You cannot make a system disk by copying these files with the **copy** command.

To create a system disk during formatting, use the **format** command with the **/s** switch. After the disk has been formatted, MS-DOS copies the three system files

to the disk. For example, the following command formats the disk in drive B and makes it a system disk:

```
format b: /s
```

To make a formatted disk a system disk, use the **sys** command. For example, to copy the system files COMMAND.COM, IO.SYS, and MSDOS.SYS to a formatted disk in drive A, type the following:

```
sys a:
```

When system files are on a floppy disk, you can use it to start your system from drive A.

Labeling a Disk

Each disk can have a name, called the *volume label*, and a number, called the *volume serial number*. MS-DOS uses the volume serial number to keep track of which disk is in a drive. MS-DOS assigns a serial number to a disk when you format it. The serial number does not change unless the disk is formatted again. Only disks formatted by MS-DOS version 4.0 and later have a serial number. MS-DOS displays the disk's volume label and serial number above the list of files in every directory.

You can change a disk's volume label by using the **label** command. The volume label you choose can contain no more than 11 characters, and it cannot include the following characters: asterisk (*), question mark (?), slash (/), backslash (\), pipe (¦), period (.), comma (,), colon (:), semicolon (;), plus sign (+), equal sign (=), less-than sign (<), greater-than sign (>), caret (^), quotation mark ("), brackets ([ ]), ampersand (&), parentheses, or any key combinations. Volume labels can include spaces but not tabs.

NOTE You can use extended characters in a label, but if you do, it is recommended that you use code page 850. If you use code page 437, support for extended characters is limited. For information about extended characters, see the Appendix.

Assigning and Deleting Labels

In Brief

To assign a volume label to a disk, use the **label** command. For example, the following command gives the disk in drive A the label *disk 1*:

```
label a:disk 1
```

The label can contain no more than 11 characters, including spaces.

To delete a volume label, use the **label** command without a name.

If you work with a large number of disks, it may be convenient to create a label for each disk. You can view the label when you use the **dir** or **vol** command.

To assign a volume label, use the **label** command. For example, use the following command to assign the label *disk 2* to a disk in drive A:

```
label a:disk 2
```

If you type a drive letter, but no label, MS-DOS prompts you for a label. For example, to label the disk in drive B, type the following command:

```
label b:
```

MS-DOS displays the current label and serial number of the disk in drive B and then prompts you to type a new volume label.

To delete a volume label, use the **label** command without a name. When MS-DOS prompts you to type a new volume label, press ENTER. A message appears, asking you to confirm deletion of the volume label. Type **y** to delete the label.

Viewing Labels

In Brief

To view the volume label and serial number of a disk, use the **dir** or the **vol** command. You can type the following command to view this information for the disk in the current drive:

```
vol
```

To view a disk's volume label and serial number, use the **dir** or **vol** command. When you use the **dir** command, the volume label and serial number for the disk that you specify are displayed above the list of files.

The **vol** command displays the volume label and serial number of the disk in the drive you specify (if the disk has no serial number, only the volume label is displayed). For example, type the following command to view the volume label and serial number of the disk in drive A:

```
vol a:
```

Making a Backup Disk

There are several ways to make backup copies of files. If you want to make backup copies of a few files, the simplest way is to use the **copy** or **xcopy** command.

If you have numerous files to back up, you can use the **backup** command to do the following:

- Back up a single directory
- Back up a directory and its subdirectories
- Back up selected files
- Add files to a backup disk that you previously created

When you've backed up files by using the **backup** command, you cannot directly access the files. If you need to retrieve any of the files, you must use the **restore** command. The **restore** command reads the backup disk and puts the files you specify back where they came from. For more information about the **restore** command, see Chapter 14, "Commands."

Backing Up a Directory

In Brief

To create and maintain a backup directory, use the **backup** command. For example, the following command backs up the files in the C:\WORK directory onto a disk in drive A:

```
backup c:\work a:
```

The simplest form of the **backup** command backs up a single directory. For example, the following command backs up the files in C:\WORK\HOME to a disk in drive B:

```
backup c:\work\home b:
```

The following prompt appears, telling you to insert your backup disk and warning that any existing files on the disk will be deleted.

```
Insert backup diskette 01 in drive B:

WARNING! Files in the target drive
B:\ root directory will be erased
Press any key to continue . . .
```

When you press a key, MS-DOS begins copying files from C:\WORK\HOME. Files in subdirectories of this directory are not copied. To cancel the command without deleting any files from the destination disk, press CTRL+C or CTRL+BREAK.

In the preceding example, MS-DOS creates two files on drive B—BACKUP.001 and CONTROL.001. It combines all the files in C:\WORK\HOME and stores them in BACKUP.001. It stores the paths of the files in CONTROL.001. MS-DOS also changes the volume label of the disk to BACKUP 001.

If more than one disk is needed to back up your files, you are prompted to insert another disk in drive B. The second disk contains the files BACKUP.002 and CONTROL.002. If there is a third disk, the files are BACKUP.003 and CONTROL.003, and so on.

When you specify the location of the files that you want to back up, you can specify a drive, a directory, a filename, or a combination of all three. If you specify only a drive, **backup** looks in the current directory of that drive for the files to back up. For example, if C:\WORK\HOME is the current directory, you could use this shorter command to back up all the files in C:\WORK\HOME:

```
backup c: b:
```

You can also back up files onto a hard disk. The **backup** command creates a directory named BACKUP in the root directory of the destination drive and stores the backup files there.

Backing Up a Directory and Its Subdirectories

In Brief

To back up a directory and its subdirectories, include the **/s** switch with the **backup** command. For example, the following command backs up every file in every directory on drive C onto a disk in drive A:

```
backup c:\ a: /s
```

You can save time by backing up a directory and all its subdirectories with one command. To include subdirectories, use the **/s** switch with the **backup** command. For example, to back up C:\WORK\HOME and all its subdirectories onto a disk in drive B, type the following command:

```
backup c:\work\home b: /s
```

MS-DOS copies the files from C:\WORK\HOME and all its subdirectories to the BACKUP.001 file on drive B. The directory structure of the files is preserved in the CONTROL.001 file.

To back up all the files on drive C onto a disk in drive B, type this command:

```
backup c:\ b: /s
```

MS-DOS starts at the root directory and backs up all the files on drive C. MS-DOS prompts you to insert new disks as needed.

Backing Up Selected Files

In Brief

To back up selected files in a directory, use wildcards, as in the following command:

```
backup c:\work\*.xls a:
```

It is not always necessary to back up all the files in a directory. Sometimes you need to back up only files of a certain type, or files that have changed since the last backup.

To back up a single file, specify its filename after its drive letter and path. For example, the following command backs up the OUTGO.XLS file in the C:\WORK\HOME\ directory onto a disk in drive B:

```
backup c:\work\home\outgo.xls b:
```

To back up files of a certain type, you can use wildcards. For example, the following command backs up only the files in C:\WORK\HOME that have a .DOC extension:

```
backup c:\work\home\*.doc b:
```

To back up files that have changed since a specified date, use the **/d** switch with the **backup** command. For example, to back up all files in C:\WORK\HOME that have been modified on or after January 20, 1991, use the following command:

```
backup c:\work\home *.* b: /d:01-20-91
```

Adding Files to a Backup Disk

In Brief

To copy files to a backup disk without deleting any files on the disk, use the **/a** switch with the **backup** command, as in the following example:

```
backup c:\work a: /a
```

To add only files that have later versions than currently backed up files or that don't already exist on the backup disk, include the **/a** and **/m** switches, as in this example:

```
backup c:\work a: /a /m
```

To copy files to a backup disk without deleting files already on the backup disk, use the **/a** switch with the **backup** command. For example, the following

command adds the files from C:\WORK\SCHOOL to those already on the disk in drive A:

```
backup c:\work\school a: /a
```

If some of the files on the C:\WORK\SCHOOL directory are already on the backup disk, MS-DOS adds the files to the disk again—it doesn't replace them. When the operation is complete, the backup disk has its original files plus the files in C:\WORK\SCHOOL that you most recently added.

If you want to back up only files that have been added or changed since the last time you backed up a directory, use the **backup** command with the **/m** switch and the **/a** switch.

For example, suppose that after you backed up the C:\WORK\SCHOOL directory, you added three new files and changed two that have already been backed up. To back up the new and changed files, you would put the original backup disk in drive A and type the following command:

```
backup c:\work\school a: /a /m
```

MS-DOS adds the new and changed files to the backup disk.

CAUTION If you use the **/m** switch without the **/a** switch, MS-DOS deletes existing files on the backup disk and copies only those that have changed since the last backup.

Making Backup Disks by Using MS-DOS Shell

To back up files when running MS-DOS Shell, you use the same parameters, switches, and wildcards and some of the same procedures as you do when you are working at the command prompt. You can back up files stored on either a hard disk or a floppy disk.

You can use parameters to specify the source and destination drives. You can use switches and wildcards to back up subdirectories and selected files, and to add files to an existing backup disk. For more information about using parameters, switches, and wildcards with the **backup** command, see previous sections in this chapter.

Shell ▶ **To back up files on a hard disk:**

1. From the Disk Utilities group in the program list, choose Backup Fixed Disk.

   The Backup Fixed Disk dialog box appears.

2. To back up your entire hard disk onto disks in drive A, choose the OK button. Or type the appropriate parameters, switches, and wildcards in the Parameters box, and then choose the OK button.

   From this point on, MS-DOS displays the same messages and prompts as it does at the command prompt.

3. When the backup is finished, press any key to return to MS-DOS Shell.

Shell ▶ **To back up files on a floppy disk:**

1. From the Disk Utilities group in the program list, choose Backup Fixed Disk.

   The Backup Fixed Disk dialog box appears.

2. Specify the drive that contains the floppy disk you want to back up and the drive that contains the disk you want backup files stored on. You can also use switches and wildcards, as appropriate.

3. When the backup is finished, press any key to return to MS-DOS Shell.

To restore the files you backed up, you can use the Restore Fixed Disk command described in "Restoring Files by Using MS-DOS Shell" later in this chapter.

Restoring Directories and Files

If you lose files you backed up, you can retrieve them by using the **restore** command. You can use the **restore** command to do the following:

- Restore all files on a disk to a specific directory or to a directory and its subdirectories
- Restore selected files

CAUTION Unless you use the **/p** switch, the **restore** command replaces all files that you specified, including those that have changed since they were backed up. For information about the **/p** switch, see the following section.

Restoring Files to a Directory

In Brief

To retrieve files that were backed up by using the **backup** command, use the **restore** command. For example, the following command restores the files on the backup disk in drive B to their original locations in the root directory of drive C:

```
restore b: c:\*.*
```

To restore files to a directory and its subdirectories, use the **/s** switch, as in the following example:

```
restore a: c:\*.* /s
```

If you want to be prompted before MS-DOS replaces files that have changed since the last backup, use the **/p** switch, as in the following command:

```
restore b: c:\*.* /p
```

The **restore** command retrieves files you previously backed up. The **restore** command requires two parameters: the first parameter tells MS-DOS where to get the files from (the location of the backup disk), and the second parameter tells MS-DOS which files to restore. For example, to restore all the files from the backup disk in drive B to the C:\WORK\HOME directory, type the following command:

```
restore b: c:\work\home\*.*
```

Drive B is the drive that contains the backup disk, and C:\WORK\HOME*.* specifies the files you want to restore—every file on the backup disk that came from the C:\WORK\HOME directory.

When MS-DOS restores files, it puts them in the directory they came from. If the directory they came from no longer exists, MS-DOS creates it.

When you use the **restore** command, MS-DOS prompts you to insert the disk that contains the backup files. When you press a key, MS-DOS displays the date of the backup and starts copying from the BACKUP.001 file to the destination directory. As files are restored, MS-DOS lists them on your screen.

If the files you want to recover were backed up on more than one disk, MS-DOS prompts you to insert the other disks. If the files you want to restore are not on the disk you specified, MS-DOS displays a "No files found to restore" message.

To have MS-DOS prompt you before it replaces a file that is read-only or that has been changed since the last backup, use the **/p** switch, as in the following command:

```
restore b: c:\work\home\*.* /p
```

NOTE The **/p** switch depends on your system clock to determine which file is most recent. Make sure your clock is accurate when using the **/p** switch.

To restore files to a directory and its subdirectories, use the **/s** switch. For example, the following command restores files from the backup disk in drive B to the C:\WORK\HOME directory and all its subdirectories:

```
restore b: c:\work\home\*.* /s
```

MS-DOS reads the CONTROL.001 file to find out which directory originally contained the files and then stores them there. Any directories that don't exist are created.

To make sure that every file on a backup disk is restored, use the **/s** switch and start restoring at the root directory. For example, the following command restores every file on the backup disk in drive A to its original location on drive C:

```
restore a: c:\*.* /s
```

Restoring Selected Files

In Brief

To restore selected files in a directory, use the **restore** command with wildcards, as in the following command:

```
restore a: c:\work\*.xls
```

If you want to be prompted before MS-DOS replaces files that have changed since the last backup, use the **/p** switch, as in the following command:

```
restore b: c:\*.* /p
```

You can restore a subset of the files that were backed up by typing a single filename or by using wildcards. For example, to restore only the C:\WORK\HOME\OUTGO.TXT file from the backup disk in drive B, you would type this command:

```
restore b: c:\work\home\outgo.txt
```

To restore only files that have a .TXT extension, you would type this command:

```
restore b: c:\work\home\*.txt
```

Viewing a List of Backup Files

You can use the **restore** command with the **/d** switch to view a list of backup files on a disk. For example, to view a list of backup files on drive C (without restoring the files), you would type the following:

```
restore c: a: /d
```

Notice that you have to specify a destination drive, as if you were actually restoring files. When you use the **restore** command with the **/d** switch, a list of backup files is displayed, but the files are not restored.

Restoring Files by Using MS-DOS Shell

To restore files when running MS-DOS Shell, you use the same switches and wildcards and some of the same procedures as you use at the command prompt. For information about the switches and wildcards you can use to restore subdirectories or selected files, see the previous sections of this chapter.

Shell ▶ **To restore files:**

1. From the Disk Utilities group in the program list, choose Restore Fixed Disk.

   The Restore Fixed Disk dialog box appears.

2. In the Parameters box, specify the source and destination drives.

3. To restore all the files on the backup disk in drive A, choose the OK button. To restore selected files or change the drive that the backup disk is in, type the appropriate drive letter, switches, and wildcards in the Parameters box, and then choose the OK button.

   From this point on, MS-DOS displays the same messages and prompts as it does at the command prompt.

4. When the operation is finished, press any key to return to MS-DOS Shell.

Recovering Files from Defective Disks

If MS-DOS or a program can no longer read a file or directory, there may be one or more damaged sectors on the disk. To recover the parts of the file or directory that are not damaged, you can use the **recover** command.

CAUTION The root directory, where the recovered files are stored, can hold only a limited number of entries. If you try to recover more files than the root directory can hold, some files will be lost. In general, you should use the **recover** command only when it is absolutely necessary.

Recovering Files

In Brief

To recover as much information as possible from a file or directory that has been stored on damaged sectors of a disk, use the **recover** command. For example, the following command recovers the COMB.TXT file on a disk in drive A:

```
recover a:comb.txt
```

You cannot retrieve the part of a file that is stored in a defective sector, but you can recover the rest of it by using the **recover** command. For example, if part of the GRAY.HIC file on a disk in drive A is no longer readable by the program that created it, you could use the following command to try to recover some of the information in the file:

```
recover a:gray.hic
```

MS-DOS reads the file one sector at a time. If any of the sectors are damaged, MS-DOS removes them from the file. MS-DOS marks the bad sectors so that information cannot be stored there in the future.

When the operation is complete, MS-DOS stores the recovered file in the root directory of the disk it came from. MS-DOS names the recovered files sequentially, beginning with FILE0001.REC.

NOTE Even if a part of a file is successfully recovered, the file might be unusable if the information that was unrecoverable is critical to the file.

If a directory is unusable, you can use the **recover** command to recover as much of the information on the disk as possible. For example, to recover files in a directory on a disk in drive A, you would type this command:

```
recover a:
```

All files that MS-DOS recovers are stored in the root directory of the disk they came from.

Substituting a Drive Letter with a Directory

In Brief

To substitute a drive letter for another drive letter and path, use the **subst** command. For example, the following command substitutes the C:\DEV directory for the drive letter A:

```
subst a: c:\dev
```

To remove the substitution, use the **/d** switch, as follows:

```
subst a: /d
```

Some programs only accept drive letters A and B. In these cases, you can use the **subst** (substitute) command, which temporarily substitutes a drive letter with another drive letter and path. While a substitution is in effect, MS-DOS regards any reference to drive A or B as a reference to a directory on your hard disk.

For example, suppose you are using a communications program that only accepts files from drive A. To substitute the drive letter A with the \COMM directory on drive C, you would type the following command before starting the program:

```
subst a: c:\comm
```

Then, when the program requests files from drive A, MS-DOS looks in C:\COMM instead.

The drive letter you specify in the **subst** command must not be greater (in alphabetic order) than the letter specified in the **lastdrive** command in your CONFIG.SYS file. For more information about the **lastdrive** command, see Chapter 11, "Customizing Your System."

When you finish using the program, remove the association between the drive and the directory by using the **/d** switch:

```
subst a: /d
```

The following commands ignore any substitutions you make when using the **subst** command: **backup**, **format**, **chkdsk**, **diskcomp**, **diskcopy**, **fdisk**, **label**, **recover**, **restore**, and **sys**.

Partitioning Your Hard Disk

Each operating system has conventions for storing files on a hard disk. If you use only MS-DOS, your entire hard disk can be set up to use MS-DOS conventions. However, if you want to use another operating system in addition to MS-DOS, you must *partition* your hard disk into DOS sections and non-DOS sections.

If you use only MS-DOS, you can create a single DOS partition that occupies your entire disk. If you use only MS-DOS and want to separate groups of directories, you can create a second DOS partition. When you use multiple partitions, MS-DOS still has access to the entire hard disk. However, the files in the second partition appear to be on a different drive.

If you are going to use your hard disk with another operating system (for example, XENIX), you must create a partition for MS-DOS and a partition for the other operating system. You use an operating system by making its partition *active.*

Partitioning your disk is different from formatting it. When you partition a disk, you specify which sections of the disk MS-DOS or another operating system can use. When you format a disk, MS-DOS prepares an existing partition to receive files. After partitioning your disk, you must still format each partition before it can be used. See "Formatting Your Hard Disk After Using Fdisk" later in this chapter.

To create one or more DOS partitions on a hard disk, use the Fdisk program described in "Using Fdisk" later in this chapter.

Understanding Hard-Disk Partitions

You can create two kinds of DOS partitions on a hard disk:

- The *primary DOS partition* is the area that stores the IO.SYS, MSDOS.SYS, and COMMAND.COM files necessary to run MS-DOS. The primary partition can contain other files as well. If you want to start MS-DOS from a hard disk, that disk must have a primary DOS partition.

- An *extended DOS partition* is an area where other non-system files can be stored on a disk. An extended partition is optional.

You can have two partitions on a hard disk: one primary DOS partition and one extended partition. The extended partition can contain up to 23 *logical drives.* A logical drive is a section of a hard disk that serves as a separate disk drive. If you create a primary partition that doesn't occupy the entire hard disk, you can create an extended partition in the remaining space. In the extended partition, you can create logical drives.

The Primary DOS Partition

If you start MS-DOS from a hard disk, the disk must have a primary DOS partition that contains the three MS-DOS system files (IO.SYS, MSDOS.SYS, and COMMAND.COM). This partition must be the active partition. In general, the primary DOS partition on the first hard disk is assigned the drive letter C.

You can reserve a portion of the disk space for the primary DOS partition. The rest of the disk space can be used for other partitions.

The Extended DOS Partition

When you create an extended DOS partition, you divide it into one or more logical drives. There are 26 letters available for logical drives (A through Z). Drives A and B are reserved for floppy disk drives. Drive C is reserved for the first primary DOS partition. Thus, there is a maximum of 23 logical drives that you can create in an extended DOS partition.

You can use logical drives to group your directories and files. Logical drives do not create more disk space, however.

For information about how to create an extended DOS partition, see "Creating an Extended DOS Partition" later in this chapter.

Non-DOS Partitions

Non-DOS partitions are partitions for other operating systems (such as XENIX). You cannot use the DOS version of the Fdisk program to create a non-DOS partition. For information about creating non-DOS partitions, see your system documentation.

The Active Partition

To start your operating system from a hard disk, you must make the primary partition (in which the operating system is stored) the active partition. For example, to use MS-DOS, make your primary DOS partition active. You can use a partition that isn't designated as active, but you cannot start an operating system from that partition. A hard disk can have only one active partition at a time.

If you have only a primary DOS partition, it must be the active partition. For more information about the active partition, see "Setting the Active Partition" later in this chapter.

Using Fdisk

The Fdisk program displays information about partitions, creates partitions and logical drives, sets the active partition, and deletes partitions and logical drives.

If your computer has never had an operating system installed on it, you can choose to have your disk partitioned during the Setup program, or you can run Fdisk to partition your disk after you set up MS-DOS.

CAUTION Fdisk destroys all existing files in partitions you modify. If you are using Fdisk to change the partitions on a disk with files on it, be sure to back up the files you want to keep before you begin. If you want to create smaller partitions on a hard disk that has only a large DOS partition, you must first back up all files you want to save.

Running Fdisk During the Setup Program

If MS-DOS version 5.0 is the first operating system to be set up on your computer, you can choose to partition your disk when you run the Setup program. By default, the Setup program creates one primary DOS partition that occupies the entire disk. If you want to create more than one partition, choose "Partition some of the free space for MS-DOS" during setup. To partition the disk, follow the steps described in the subsequent sections of this chapter. When you finish creating partitions, MS-DOS continues the Setup program. For more information about setting up MS-DOS, see *Microsoft MS-DOS Getting Started*.

Running Fdisk After MS-DOS Has Been Set Up

After you set up MS-DOS version 5.0, you can repartition your disk by typing **fdisk** at the command prompt. When the Fdisk program starts, the main menu appears, as follows:

```
                MS-DOS Version 5.00
            Fixed Disk Setup Program
        (C)Copyright Microsoft Corp. 1983-1990

                    FDISK Options

Current fixed disk drive: 1

Choose one of the following:

1. Create DOS Partition or Logical DOS Drive
2. Set active partition
3. Delete DOS Partition or Logical DOS Drive
4. Display partition information


Enter choice: [1]


Press Esc to exit FDISK
```

CAUTION If you use Fdisk to change existing partitions on a hard disk, you lose the information contained in those partitions. Be sure you have copies of all files in a partition before using Fdisk to change the partition.

To choose a menu option, type the option number, and then press ENTER. To return to a previous menu, press ESC. To quit Fdisk, return to the main menu, and then press ESC.

Each menu displays a "Current fixed disk drive" message, followed by a number. If you have only one hard (fixed) disk drive, the number is always 1. If you have more than one hard disk drive, the number shows which drive Fdisk is currently working with. The first hard disk drive in your system is 1, the second is 2, and so on. Changing the current drive when you are using Fdisk doesn't change the current drive when you return to the command prompt. The current drive refers only to physical disk drives, not logical drives, when you are using Fdisk.

Viewing Partition Data

You can view information about the status, type, and size of the partitions on a hard disk by choosing Display Partition Information (option 4) from the Fdisk main menu. The Display Partition Information screen looks like this:

```
                Display Partition Information

Current fixed disk drive: 1

Partition  Status   Type    Volume Label  Mbytes  System   Usage
 C: 1         A    PRI DOS                 21    FAT16      50%
    2              EXT DOS                 21               50%

Total disk space is 42 Mbytes (1 Mbyte = 1048576 bytes).

The Extended DOS partition contains
logical DOS drives. Do you want to
display logical drive information?  [Y]

Press ESC to continue
```

The information varies, depending on the number, size, and type of partitions on your hard disk.

Column	Description
Partition	Shows the drive letter associated with each partition, and the number of each partition.
Status	Displays the letter A next to the active partition.
Type	Shows whether a partition is a primary DOS partition (PRI DOS), an extended DOS partition (EXT DOS), or a non-DOS partition.
Volume Label	Shows the volume label of the primary partition. This field may be blank.
Mbytes	Shows the size of each partition, in megabytes.
System	Shows the type of file system being used on the partition.
Usage	Shows the percentage of the current disk that each partition occupies.

If there is an extended DOS partition that contains logical drives, a prompt appears, asking whether you want to see information about that partition's logical drives. Type **y** if you want to view this information.

The screen displaying information about logical drives would resemble this:

```
Display Logical DOS Drive Information

Drv  Volume Label  Mbytes  System  Usage
 D:   BACKUPA        18     FAT16    90%
 E:   BACKUPB         2     FAT12    10%

Total Extended DOS Partition size is 20 Mbytes (1 Mbyte = 1048576
bytes)

Press ESC to continue
```

The information varies, depending on the number and size of the logical drives.

Column	Description
Drv	Displays the drive letter of each logical drive.
Volume Label	Shows the label assigned to each drive. This field may be blank.
Mbytes	Shows the size of each logical drive, in megabytes.
System	Shows the type of file system being used on that partition.
Usage	Shows the percentage of available space in the extended DOS partition that each logical drive occupies.

Creating a Primary DOS Partition

The hard disk you use to start MS-DOS must have a primary DOS partition. You can create a primary DOS partition that occupies the entire hard disk or only part of it. If you want to create an extended DOS partition with logical drives, or if you want to have space for a non-DOS partition, you must create a primary DOS partition that doesn't occupy your entire disk.

You cannot change the size of an existing primary DOS partition. If you want a primary DOS partition of a different size, you must delete the existing partition and create a new one. When you delete the existing partition, you lose any information stored there, so back up files you want to save. For information about deleting a partition, see "Deleting a Partition or Logical Drive" later in this chapter.

If your hard disk does not already have a partition, you can use the following procedure to create a primary DOS partition that occupies the entire disk.

▶ To create a primary DOS partition that occupies the entire hard disk:

1. From the Fdisk main menu, choose Create DOS Partition or Logical DOS Drive (option 1) by pressing ENTER.

   Another menu appears.

2. Choose Create Primary DOS Partition (option 1) by pressing ENTER.

   Another prompt appears, displaying this message:

   ```
   Do you wish to use the maximum size for a Primary DOS Partition
   and make the partition active (Y/N).........? [Y]
   ```

3. Type **y**.

   (If you type **n**, Fdisk prompts you to create a smaller primary partition. See the following procedure for more information.)

   Fdisk creates a primary partition that takes up all the available space on the hard disk. If you have only one hard disk, MS-DOS displays the following message:

   ```
   System will now restart

   Insert DOS system diskette into drive A:
   Press any key when ready
   ```

4. Insert an MS-DOS system disk, and then press any key.

After partitioning the disk, you need to format it by using the **format** command with the **/s** switch. For more information, see "Formatting Your Hard Disk After Using Fdisk" later in this chapter.

▶ To create a primary DOS partition that occupies part of the hard disk:

1. From the Fdisk main menu, choose Create DOS Partition or Logical DOS Drive (option 1) by pressing ENTER.

   Another menu appears.

2. Choose Create Primary DOS Partition (option 1) by pressing ENTER.

   Another prompt appears, displaying this message:

   ```
   Do you wish to use the maximum available size for a Primary DOS
   Partition and make the partition active (Y/N).........? [Y]
   ```

3. Type **n**.

   Another menu appears.

4. If you want the default size (100 percent), press ENTER. Otherwise, type the number of megabytes or the percentage of the disk to use. If you type a percentage, follow the number with a percent (%) sign.

   The following message appears:

   ```
   Primary DOS Partition created, drive letters changed or added.
   ```

5. To return to the Fdisk main menu, press ESC.

When you quit Fdisk, you'll need to format the new partition on your hard disk by using the **format** command with the **/s** switch. For more information, see "Formatting Your Hard Disk After Using Fdisk" later in this chapter.

NOTE When you create a primary DOS partition that doesn't occupy your entire hard disk, you must make the partition active before you can use the hard disk with MS-DOS. For more information about making a partition active, see "Setting the Active Partition" later in this chapter.

Creating an Extended DOS Partition

If you want to divide your hard disk into more than one DOS partition, you can create an extended DOS partition in addition to the primary DOS partition. Within the extended DOS partition, you typically can assign up to 23 logical drives. Logical drives are areas of your hard disk that MS-DOS treats as separate disk drives. You must assign at least one logical drive to an extended DOS partition.

If you have one hard disk, there must already be a primary DOS partition that uses only part of the disk before you can create an extended DOS partition. If you have more than one hard disk, only the disk you use to start your system must have a primary DOS partition; your other hard disk(s) can contain only extended DOS partitions. However, if you are using only one partition per disk, you should set the single partition as a primary partition.

▶ To create an extended DOS partition:

1. From the Fdisk main menu, choose Create DOS Partition or Logical DOS drive (option 1).

   The Create DOS Partition menu appears.

2. From the Create DOS Partition menu, choose Create Extended DOS Partition (option 2).

A menu appears, showing the total number of megabytes available for an extended partition. The default for the partition size is the maximum available space on the hard disk drive, minus the size of the primary partition. If there is no space available, you must delete and recreate the primary DOS partition so it is smaller, or reduce the size of any non-DOS partitions that exist.

3. If you want the default size, press ENTER. Otherwise, type the number of megabytes or the percentage of the unused disk space to be used for the extended DOS partition. If you type a percentage, follow the number with a percent (%) sign.

   The Create Logical Drive(s) in the Extended DOS Partition menu appears.

When you create an extended DOS partition, you can set up one or more logical drives. See the following section for more information.

NOTE If Fdisk finds defective tracks at the beginning of an extended DOS partition, it adjusts the partition boundaries to avoid those tracks.

Creating Logical Drives in an Extended DOS Partition

To store information in an extended DOS partition, you must create one or more logical drives. Each logical drive is assigned a drive letter. You can store and retrieve information on a logical drive as though it were a physical disk drive. For example, you can use logical drive D to store files for a particular program, and you can work with those files by specifying drive D rather than a directory.

▶ To create or modify a logical drive:

1. Create an extended DOS partition.

   (See the preceding section for information about how to create an extended DOS partition.)

2. Using the Create Logical Drive(s) menu, type the number of megabytes or the percentage of the partition space you want to use for the first logical drive. If you type a percentage, follow the number with a percent (%) sign. If you want one logical drive to occupy the whole extended DOS partition, press ENTER.

3. Continue specifying the sizes of partitions until you have used up the entire partition or until you have created all the logical drives you want.

   If the entire partition is assigned to logical drives, the Fdisk main menu reappears. To quit the menu before all the space has been allocated, press ESC.

After you create a logical drive, you must format it. For more information about formatting a logical drive, see "Formatting Your Hard Disk After Using Fdisk" later in this chapter.

How Drive Letters Are Assigned

The primary DOS partition on your startup hard disk is drive C. The drive letters of additional hard disks and logical drives depend on the number of disks you are using and how they are partitioned.

If you have only one hard disk, logical drives you create in the extended DOS partition are given letters beginning with D. For example, if you create five logical drives in the extended DOS partition, they are named D, E, F, G, and H.

If your system has more than one hard disk, and you have only one primary DOS partition, all logical drives you create in the extended DOS partitions are assigned letters consecutively.

Suppose your system has two hard disks. The first has a primary DOS partition and an extended DOS partition with two logical drives, and the second hard disk has an extended partition with two logical drives. The primary DOS partition on the first disk is drive C; the two logical drives on the disk are drives D and E. The two logical drives on the second disk are drives F and G.

You may have primary DOS partitions on more than one hard disk. If so, MS-DOS assigns drive letters consecutively to all the primary DOS partitions first, and then assigns drive letters consecutively to the logical drives in the extended DOS partitions.

Suppose a system has two hard disks, each containing a primary DOS partition and an extended DOS partition with two logical drives. The primary DOS partition on the first disk is drive C; the primary DOS partition on the second is drive D. Logical drives on the first disk are drives E and F, and logical drives on the second are drives G and H.

MS-DOS assigns different drive letters to existing logical drives if you add another hard disk to your system and create a primary DOS partition on it. (If you create only an extended DOS partition, the drive letters remain unchanged.) For example, suppose you have a disk with a primary DOS partition named drive C and an extended DOS partition that has a logical drive named drive D. If you add a hard disk to your system and create a primary MS-DOS partition on it, the primary DOS partition on the new disk becomes drive D, because primary DOS partitions are assigned consecutive drive letters. The logical drive on the original disk becomes drive E instead of its prior designation as drive D.

If you add another hard disk to your system and create only an extended DOS partition on it, the drives on the original disk are unaffected. Any logical drive that you create on the new disk is given the next available drive letter. For example, suppose you have a hard disk with a primary DOS partition (drive C) and two logical drives (drives D and E). If you add a second hard disk with two logical drives, these drives are named F and G. The drive letters on the original disk remain unchanged.

Setting the Active Partition

The active partition contains the operating system that is loaded when you start or reset your system. Unless you create a primary DOS partition that occupies your entire hard disk, you must set the active partition by using Fdisk. If you are using a non-DOS partition, you must reset the active partition when you want to switch between MS-DOS and the non-DOS operating system. Only one partition can be active at a time.

▶ To set the active partition:

1. From the Fdisk main menu, choose Set Active Partition (option 2).

   A menu appears that displays the status of each partition. The active partition is indicated by the letter A.

2. Type the number of the partition you want to make active. The default setting is the current active partition number.

3. To return to the Fdisk main menu, press ESC.

You can make only primary partitions active. If you try to make an extended DOS partition active, Fdisk displays a message similar to the following:

```
Partition selected (3) is not startable, active partition not
changed.
```

Deleting a Partition or Logical Drive

You might need to change the size of your partitions. For example, you might need to make room for a non-DOS operating system or change the size and number of the logical drives. You cannot reduce or enlarge an existing partition. If you want to change a partition's size, you must delete the partition and recreate it.

When you delete a partition, all information in the partition is deleted and cannot be recovered. Therefore, be sure you have backup copies of the information you want to save. When you delete a partition, you don't lose information stored in

other partitions on your disk. For example, if you delete the extended DOS partition but not the primary DOS partition, files in the primary DOS partition are not deleted.

If you want to delete the primary DOS partition on a disk, you must first delete each logical drive in the extended partition, then the extended partition itself.

You can delete one or more logical drives in the extended DOS partition of a hard disk. All information on a logical drive is lost when you delete it. Deleting one logical drive does not affect the information on other logical drives.

If there are logical drives that have drive letters greater (in alphabetic order) than the drive you delete, these letters will change. Suppose, for example, that you have logical drives D, E, and F on a disk. If you delete drive D, drive E becomes drive D, and drive F becomes drive E.

NOTE To continue using MS-DOS after you delete the primary DOS partition, you must restart your system, using an MS-DOS system disk in drive A. Make sure you have a floppy disk formatted as a system disk before you delete the primary partition.

▶ To delete a partition or logical DOS drive:

1. From the Fdisk main menu, choose Delete Partition or Logical DOS Drive (option 3).

   Another menu appears.

2. Type the number of the option you want.

   Fdisk displays the status of the partition or logical drives along with a message warning that the data in the partition or logical drive will be lost.

3. Type the number that corresponds with the drive letter, and type the volume label of the logical drive you want to delete.

   Fdisk displays a message confirming the information you typed.

4. Type **y** to delete the partition or drive.

If you deleted your primary DOS partition, you need to create a new one before you can use MS-DOS from your hard disk. Before you quit the Fdisk program, follow these steps:

1. From the Fdisk main menu, choose Create DOS Partition or Logical DOS Drive (option 1).

   Another menu appears.

2. Follow the instructions in the preceding sections to create a DOS partition that occupies either your entire hard disk or only part of it.

   When Fdisk is finished, a prompt appears.

3. Insert a system disk in drive A, and press any key to restart your system.
4. Format the new partition by using the **format** command with the **/s** switch.
5. Remove the system disk from drive A, and restart your system.

   At this point, your hard disk contains the MS-DOS files IO.SYS, MSDOS.SYS, and COMMAND.COM. You can now install the rest of the MS-DOS files by using the MS-DOS Setup disks.

Working with More Than One Hard Disk

If your system has more than one hard disk drive, you can use Fdisk to create and modify partitions on any drive. Only one disk must have a primary DOS partition. Your other disks can have primary DOS partitions or extended MS-DOS partitions, or both. On most computers with multiple hard disks, only drive C can be used to start the operating system.

When you start Fdisk, you work with the first hard disk on your system. To work with a different disk drive, you must choose Change Current Fixed Disk Drive (option 5) from the Fdisk main menu, and specify the number of the drive you want to partition. If you have only one hard disk drive, the Change Current Fixed Disk Drive option does not appear on the Fdisk main menu.

Formatting Your Hard Disk After Using Fdisk

When you quit Fdisk after you change the size of any of the DOS partitions on your hard disk, this message is displayed:

```
System will now restart
```

If you changed the size of your primary DOS partition, Fdisk prompts you to insert an MS-DOS system disk in drive A and press any key. You then return to the command prompt.

After using Fdisk, you must use the **format** command to prepare any partition that you create or change. If you don't format the disk, MS-DOS gives you the following error message when you try to use the hard disk:

```
Invalid media type
```

If you are formatting the primary DOS partition of the hard disk from which you will start your system, be sure to transfer the MS-DOS system files from a floppy

system disk by using the **format** command with the **/s** switch or by using the **sys** command. For more information, see "Creating a System Disk" earlier in this chapter.

When you format your hard disk, you must format each new partition separately. For example, if you made your primary DOS partition (drive C) smaller and created two logical drives in an extended DOS partition (drives D and E), you must use the **format** command three times:

```
format c: /s
format d:
format e:
```

The first command formats the primary partition and transfers the MS-DOS system files from the startup disk to that partition. The second and third commands format the logical drives.

CAUTION If you made changes to some but not all of the partitions or logical drives on your system, be careful when you format the partitions or drives you changed. Because Fdisk may assign different letters to drives after you change partitions or logical drives, you might inadvertently format a drive that has information stored on it.

Before you format a drive, you can use the **chkdsk** command to check the contents of the drive. If you see the message "Probable non-DOS disk" or "Invalid media type" before the disk information is displayed, the drive is not formatted. If the disk information is displayed without this message, the drive is formatted.

You may want to give a descriptive label to each logical drive you create so that you know what information is on it when you make changes to your system. You can do this by using the **/v** switch when you use the **format** command.

For more information about formatting, see "Formatting Disks" earlier in this chapter, or see Chapter 14, "Commands."

Chapter 7

Advanced Command Techniques

7

Redirection characters, editing keys, and the Doskey program enable you to perform many useful tasks. A *redirection character* changes the place that a command gets information from or sends information to. Redirection characters are useful when you want MS-DOS to save information in a file rather than display it on your screen. You can also use a *filter command* to redirect information that a command typically would send to the screen. Filter commands help you sort, view, and select parts of the output of a command.

By using *editing keys*, you can quickly view and edit your last command rather than retype it. The *Doskey program* stores the commands you type so you don't have to retype them to use them. Doskey includes the MS-DOS editing keys plus a number of other keys that are useful for editing commands. With Doskey, you can also create macros that contain a series of commands. A macro runs much like a batch program.

Redirecting Command Input and Output

Unless you specify otherwise, MS-DOS receives input from your keyboard and sends output to your screen. Sometimes it is useful to redirect the input or output to a file or a printer. For example, you might want to redirect a directory listing from the screen to a file.

To redirect the input or output of a command, you use one of the following redirection characters:

- The greater-than sign (>) sends the output of a command to a file or a device, such as a printer.

- The less-than sign (<) takes the input needed for a command from a file rather than from the keyboard.
- The double greater-than sign (>>) adds output from a command to the end of a file without deleting the information already in the file.

Redirecting the Output of a Command

Almost all commands send output to your screen. Even commands that send output to a drive or printer also display messages and prompts on your screen.

To redirect the output from the screen to a file or printer, use the greater-than sign (>). You can use the greater-than sign with most MS-DOS commands. For example, in the following command, the directory listing produced by the **dir** command is redirected to the DIRLIST.TXT file:

```
dir > dirlist.txt
```

If the DIRLIST.TXT file doesn't exist, MS-DOS creates it. If DIRLIST.TXT exists, MS-DOS replaces the information in the file with the output from the **dir** command.

The following command creates a file named CHECKDSK.TXT, which contains the output of the **chkdsk** command:

```
chkdsk a: > checkdsk.txt
```

If CHECKDSK.TXT already exists, MS-DOS replaces its contents with the output that the **chkdsk** command usually sends to your screen.

To add the output from a command to the end of a file without losing any of the information already in the file, use a double greater-than sign (>>). For example, in the following command, the directory listing produced by the **dir** command is appended to the DIRLIST.TXT file:

```
dir >> dirlist.txt
```

To send the output of a command to a printer, use the greater-than sign with the name of the port to which the printer is connected. For example, the following command redirects the output of the **dir** command from the screen to the printer attached to the LPT1 port:

```
dir > lpt1
```

NOTE Some command output, such as error messages, may not be redirected when using the greater-than sign (>).

Redirecting the Input to a Command

Just as you can send the output of a command to a file or printer rather than to your screen, you can take the input for a command from a file rather than from the keyboard. To take input from a file, use the less-than sign (<). For example, the following command takes the input for the **sort** command from the LIST.TXT file:

```
sort < list.txt
```

MS-DOS alphabetizes the lines of the LIST.TXT file and displays the result on your screen.

For more information about the **sort** command, see "Sorting Text Files" later in this chapter.

Passing Information Through Filter Commands

Filter commands divide, rearrange, or extract portions of the information that passes through them. MS-DOS has three filter commands:

- The **more** command displays the contents of a file or the output of a command one screen at a time.
- The **find** command searches through files and command output for the characters you specify.
- The **sort** command alphabetizes files and command output.

To send input from a file to a filter command, use the less-than sign (<). If you want the filter command to get its input from another command, use the pipe (¦).

NOTE Before you can use a pipe, you should set a TEMP environment variable. For information about setting environment variables, see the **set** command in Chapter 14, "Commands."

Controlling the Screen Display by Using the More Command

The **more** command displays the contents of a file or the output of a command one screen at a time. For example, the following **more** command displays the contents of the LIST.TXT file one screen at a time:

```
more < list.txt
```

After a screen of information is displayed, the word "More" appears. To continue to the next screen, press any key. To stop the command without viewing more information, press CTRL+C.

The **more** command is helpful if you are working with a command that produces more than one screen of output. For example, suppose you want to view a directory tree for your hard disk. If you have more directories than MS-DOS can display on the screen, you can use the **tree** command with a pipe (¦) and a **more** command, as in the following example:

```
tree c:\ ¦ more
```

The first screen of output from the **tree** command is displayed, followed by the word "More." MS-DOS pauses until you press any key (except the PAUSE key).

Searching for Text by Using the Find Command

The **find** command searches one or more files for the text you specify. MS-DOS displays every line containing that text. The **find** command can be used as a filter command or as a standard MS-DOS command. For information about using **find** as a standard MS-DOS command, see Chapter 14, "Commands."

To use **find** as a filter command, include a less-than sign (<) and a filename to search through. (The search is case-sensitive.) For example, the following command finds occurrences of the string "Pacific Rim" in the file TRADE.TXT:

```
find "Pacific Rim" < trade.txt
```

To save the output of the **find** command rather than display it, use a greater-than sign (>) and the name of the file that is to store the output. For example, the following command finds occurrences of "Pacific Rim" in the TRADE.TXT file and saves them in the NWTRADE.TXT file:

```
find "Pacific Rim" < trade.txt > nwtrade.txt
```

To print the output rather than display it, use a greater-than sign and the name of the port your printer is attached to, as in the following command:

```
find "Pacific Rim" < trade.txt > LPT1:
```

Sorting Text Files

The **sort** command alphabetizes a text file or the output of a command. For example, you would use the following command to sort the contents of a file named LIST.TXT and display the results on your screen:

```
sort < list.txt
```

In this example, the **sort** command sorts the lines of the LIST.TXT file and displays the results without changing the file. To save the output of the **sort** command rather than display it, include a greater-than sign (>) and a filename in the command. For example, you would use the following command to alphabetize the lines of the LIST.TXT file and store the results in the ALPHLIST.TXT file:

```
sort < list.txt > alphlist.txt
```

To sort the output of a command, type the command followed by a pipe (¦) and the **sort** command. For example, the following command sorts the output of the **find** command:

```
find "Jones" maillst.txt ¦ sort
```

When you type this command, MS-DOS lists in alphabetic order the lines in which the string "Jones" appears.

NOTE You can use the **sort** command on files that are 64K or less in size. For more information about the **sort** command, see Chapter 14, "Commands."

Combining Commands with Redirection Characters

You can combine filter commands, other commands, and filenames to make custom commands. For example, you could use the following command to store the names of files that contain the string "LOG":

```
dir /b ¦ find "LOG" > loglist.txt
```

MS-DOS sends the output of the **dir** command through the **find** filter command and stores the filenames that contain the string "LOG" in the LOGLIST.TXT file. The results are stored as a list of filenames (for example, A.LOG, LOGDAT.SVD, and MYLOG.BAT).

To use more than one filter in the same command, separate the filters with a pipe (¦). For example, the following command searches every directory on drive C, finds the filenames that include the string "LOG", and displays them one screen at a time:

```
dir c:\ /s /b ¦ find "LOG" ¦ more
```

Because you use a pipe (¦), MS-DOS sends the output of the **dir** command through the **find** command. The **find** command selects only filenames that contain the string "LOG". The **more** command displays the filenames that are selected by the **find** command, one screen at a time.

Using Editing Keys

MS-DOS provides several editing keys that you can use to edit the last command you typed. For example, suppose you misspell the name of a file in a **copy** command. Rather than retype the entire command, you can use editing keys to view the command and change the part that is misspelled. This section describes the editing keys you can use if you don't have the Doskey program installed. For information about using the Doskey program, see "Using Doskey to Work with Commands" later in this chapter.

When you type a command, MS-DOS carries out the command and saves it in a temporary location called the *template.* For example, suppose you type the following command:

```
type ada.txt
```

When you press ENTER, MS-DOS displays the contents of ADA.TXT and copies the command "type ada.txt" to the template. The template can contain only the previously typed command. For information about saving and reusing more than one command, see "Using Doskey to Work with Commands" later in this chapter.

You can use the command stored in the template as a starting point for typing your next command. The following editing keys display and edit the previous command, stored in the template:

F1 (or RIGHT ARROW)	Copies the previous command to your screen, one character at a time. One character is displayed each time you press F1.
F2	Copies the previous command to your screen, up to but not including the character you specify. For example, suppose the previous command was "type myfile.bak". If you press F2 and type **i**, MS-DOS displays "type myf".
F3	Copies the remainder of the previous command to your screen.
F4	Deletes the previous command from the template, starting from the beginning of the command, up to but not including the letter you specify. For example, suppose the previous command was "type ada.txt". If you press F4 and type **d**, and then copy the template to your screen by pressing F3, MS-DOS displays "da.txt".

F5	Copies the current command line to the template, but does not carry out the command.
F6	Places a CTRL+Z character (^Z) in the current command line.
LEFT ARROW or BACKSPACE	Deletes the character before the cursor on the current command line, without affecting the template.
DEL	Deletes the character on the template corresponding to the current cursor position.
INS	Starts insert mode so that characters you type do not replace characters in the same position in the template. Press INS again to stop insert mode.
ESC	Cancels the current command line without carrying it out, leaving the template unchanged.

NOTE Some of these keys function differently when Doskey is installed.

Copying a Command Without Retyping It

When you type a command, MS-DOS carries it out, copies the command to the template, and displays the command prompt. For example, suppose you type the following command:

```
copy c:\work\*.txt a:
```

The files are copied to a disk in drive A, the command is copied to the template, and the command prompt appears. At this point, you can view the entire previous command by pressing F3. The command from the template is displayed:

```
copy c:\work\*.txt a:_
```

The cursor appears at the end of the command. To copy your files to another disk, you insert a different disk in drive A, and then press ENTER. MS-DOS carries out the **copy** command again.

Editing a Command

If you only want to modify the previous command slightly for your next command, you can use the editing keys.

Fixing a Mistyped Command

It is often easier to edit your last command than to type it again. Using the F3 and LEFT ARROW keys, you can quickly fix a command that you mistyped. For example, suppose you typed *.DPC when you meant to type *.DOC, as in the following command:

```
copy c:\work\*.dpc a:
```

Rather than retype the command, you can edit the incorrect one. To edit the command, first press F3. The command from the template is displayed:

```
copy c:\work\*.dpc a:_
```

The cursor appears at the end of the command. To change DPC to DOC, press the LEFT ARROW key five times to move the cursor back five spaces:

```
copy c:\work\*.d_
```

To correct the command, type **o**, press F3, and then press ENTER.

Using the F1 and INS Keys to Edit

Suppose the following command is stored in the template:

```
dir c:\work\finals\*.bak
```

If you want the command to be displayed one character at a time, press F1. For example, if you press F1 11 times, the first 11 characters of the command are displayed:

```
dir c:\work_
```

To insert text in the command without changing where you are in the template, press INS before typing the new letters. For example, to change WORK to WORKNEW, press INS and then type **new**:

```
dir c:\worknew_
```

To continue copying characters from the template to the command line, press F1 as many times as necessary. For example, to copy \FINAL from the template to the command line, press F1 six times:

```
dir c:\worknew\final_
```

Using the F2 Key to Edit

If you want only a part of the previous command displayed, use the F2 key. For example, suppose this is your previous command:

```
dir c:\worknew\final\*.bak
```

If you want to change the command so that it gives you a directory listing only for C:\WORKNEW, you can use F2. To copy the characters in the template up to the first backslash (\), press F2 and type a backslash. The following appears on your screen:

```
dir c:_
```

MS-DOS copies the characters up to, but not including, the backslash. The next character in the template is the first backslash in the command. To copy the characters up to the next backslash, press F2 and type a backslash again. The following appears on your screen:

```
dir c:\worknew_
```

To carry out the command, press ENTER.

Using Doskey to Work with Commands

You can use the Doskey program to view, edit, and carry out MS-DOS commands that you have used previously. Doskey includes the MS-DOS editing keys and other keys that make it easy for you to use previous commands. When using Doskey, you can type several commands on one line.

In addition, you can create, run, and save command *macros*. A macro is one or more MS-DOS commands that are stored in random-access memory (RAM); it runs much like a batch program. The first time you use Doskey, MS-DOS loads it into RAM. Thereafter, MS-DOS saves your previous commands and any macros you create.

Although you have more editing power with Doskey than you do with MS-DOS editing keys, Doskey takes up a small amount of your computer's temporary memory. If you need the maximum amount of memory for other purposes, you might want to use MS-DOS editing keys instead of Doskey.

Installing Doskey

To load the Doskey program into memory, type **doskey** at the command prompt.

Unless you indicate otherwise, MS-DOS reserves 512 bytes of temporary memory for the commands and macros you record. If your average command contains 15 characters, you can store about 35 commands with the amount of memory reserved. The resident portion of the Doskey program itself occupies about 3K of memory.

If you want to reserve more or less memory, you can include the **/bufsize=** switch in the command. For example, to reserve 300 bytes of memory for recorded commands, type the following command:

```
doskey /bufsize=300
```

As the memory you reserved for Doskey is used up, the oldest commands are removed so that the new ones can be stored in the buffer. You can clear the buffer by pressing ALT+F7.

Typing More Than One Command on a Line

Typically, you type one command per line. Once you install Doskey, you can type several commands on a line. You separate each command by pressing CTRL+T. A paragraph mark (¶) appears on your screen each time you press CTRL+T. You can type as many commands as you like on one line as long as the total line length does not exceed 128 characters.

For example, to delete all the files in the \TMP directory and then remove the directory, you could type the following two commands on the same line:

```
del \tmp\*.* ¶ rd \tmp
```

MS-DOS carries out the **del** command and prompts you to confirm the deletion. MS-DOS then carries out the second command.

Viewing Previous Commands

Once Doskey is loaded, it keeps a list of your commands as you type them. You can use the following keys to view previous commands. To carry out a command again after it is displayed, press ENTER.

UP ARROW	Displays the previous command in the list.
DOWN ARROW	Displays the next command in the list.
F7	Displays the list of commands Doskey has stored.
F8	Cycles through the stored commands that start with the characters you specify. (You type the search text, and then press F8).
F9	Prompts you to type the number of the stored command you want. To see the numbered list of commands, press F7.

PAGE UP	Displays the oldest command in the list.
PAGE DOWN	Displays the newest command in the list.
ESC	Clears the command from the screen.

Viewing the List of Stored Commands

Doskey displays a numbered list of the commands it saves. For example, suppose you type the following three commands after you load Doskey:

```
copy c:\work\*.txt c:\backup
dir c:\backup\*.txt
dir c:\work\*.txt
```

Doskey saves the three commands. To view the full list of commands, press F7. A numbered list of the commands appears:

```
1: copy c:\work\*.txt c:\backup
2: dir c:\backup\*.txt
3: dir c:\work\*.txt
```

If there are more commands in the list than can fit on one screen, Doskey pauses after each screen of commands. To see the next screen of commands in the list, press any key except PAUSE.

Viewing the Previous or Next Command

The first time you press the UP ARROW key, Doskey displays the most recent command. For example, suppose Doskey has stored the three commands shown in the previous section:

```
1: copy c:\work\*.txt c:\backup
2: dir c:\backup\*.txt
3: dir c:\work\*.txt
```

Doskey displays the following command the first time you press the UP ARROW key:

```
dir c:\work\*.txt_
```

You can reuse the command by pressing ENTER, or you can edit the command by using the keys described later in this chapter.

If you press the UP ARROW key more than once, Doskey displays commands further back in the list. To move backward in the list and view command number 2, press the UP ARROW key again. If you press the UP ARROW once more, Doskey displays command number 1.

To move forward in the list, press the DOWN ARROW key. For example, if command number 2 is displayed and you press the DOWN ARROW key, Doskey displays command number 3:

```
dir c:\work\*.txt
```

Viewing the First or Last Command

To view the most recent command, press PAGE DOWN. To view the oldest command, press PAGE UP. For example, suppose Doskey has saved the following list of commands:

```
1: copy c:\work\*.txt c:\backup
2: dir c:\backup\*.txt
3: dir c:\work\*.txt
```

If you press PAGE DOWN, Doskey displays the following command:

```
dir c:\work\*.txt_
```

If you press PAGE UP, Doskey displays this command:

```
copy c:\work\*.txt c:\backup_
```

Viewing Other Commands in the List

You can use F9 or F8 to view a specific command in the list. Suppose Doskey has saved the following list of commands:

```
1: a:
2: dir
3: c:\myuts\figdsk a: time=30 space=35.8
4: dir
5: del *.tmp
```

If you want to view command number 3 again, you can use the arrow keys or press F9. When you press F9, the following appears:

```
Line number:
```

To view line 3, type **3**, and then press ENTER.

You can also use F8 to view a command that begins with letters you specify. For example, to view the command that begins with C:\, type **c:** at the command prompt, and then press F8.

When you press F8, Doskey displays the most recent command that begins with the characters you typed. You can press F8 again to view the next command in the list that begins with the characters you typed. Keep pressing F8 to cycle through all the matching commands. If Doskey doesn't find a matching command in the list, nothing happens.

Editing and Using Previous Commands

As you type a new command or after you view a previous command, you can use editing keys to change it. You can use the same editing keys with Doskey that you use with the command template described in "Using Editing Keys" earlier in this chapter. When you use some of these keys with Doskey, however, you see slightly different results. Doskey provides a number of additional editing keys that make it easy to change a previously typed command. The editing keys affect only the displayed command; they do not change any commands that Doskey has already stored.

You can use the following editing keys with Doskey:

HOME	Moves the cursor to the beginning of the displayed command.
END	Moves the cursor to the end of the displayed command.
LEFT ARROW	Moves the cursor back one character in the displayed command.
RIGHT ARROW	Moves the cursor forward one character in the displayed command.
CTRL+LEFT ARROW	Moves the cursor back one word in the displayed command.
CTRL+RIGHT ARROW	Moves the cursor forward one word in the displayed command.
BACKSPACE	Moves the cursor back one character. If the cursor is at the end of the line, or if you are in insert mode, it also deletes the character preceding the cursor.
DEL	Deletes the character at the cursor.
CTRL+END	Deletes all characters from the cursor to the end of the line.
CTRL+HOME	Deletes all characters from the cursor to the beginning of the line.
INS	Toggles between insert mode and replace mode.
ESC	Clears the displayed command from the screen.

If you hold down CTRL while you press the RIGHT ARROW or LEFT ARROW key, the cursor moves to the beginning of the next or previous word. A word in this

case is a group of characters separated from other characters by a space. For example, the following command has three words:

```
copy c:\games\suzz.exe a:_
```

If the cursor is at the end of the line, as in this example, you can move it to the C in C:\GAMES\SUZZ.EXE by pressing CTRL+LEFT ARROW twice.

With the cursor anywhere in the word C:\GAMES\SUZZ.EXE, you can move the cursor to the beginning of the next word by pressing CTRL+RIGHT ARROW. If you press CTRL+RIGHT ARROW again, the cursor moves to the end of the line.

By pressing the INS key, you can add characters at the position of the cursor. The INS key toggles between insert and replace mode. In replace mode, new characters you type replace any characters that follow the cursor. After you press INS, you switch to insert mode; the character at the cursor position and the characters following the cursor move right as you type. For example, suppose the following line is displayed, and the cursor is under the S in SUZZ.EXE:

```
copy c:\games\suzz.exe a:
```

To change the line C:\GAMES\SUZZ.EXE to read C:\GAMES\NEW\SUZZ.EXE, you press INS and type **new**. The line now appears like this:

```
copy c:\games\new\suzz.exe a:
```

To turn off insert mode, press INS again. The characters you now type replace any characters following the cursor. Insert mode is turned off when you press ENTER to carry out a command. You can start Doskey and specify insert mode as the default by using the **/insert** switch. For more information about the **doskey** command, see Chapter 14, "Commands."

Deleting the List of Stored Commands

To delete the list of stored commands and begin a new list, you press ALT+F7. You also delete the list when you reinstall Doskey or reset your system.

Saving the List of Stored Commands in a Batch Program

To save the list of stored commands, you can type the **doskey** command with the **/history** switch, the output redirection character (>), and the name of the file in which you want the list stored. For example, to store your list of commands in the SAVCOMMS.TXT file, you would type the following command:

```
doskey /history > savcomms.txt
```

To create a batch program by using Doskey, first press ALT+F7 to delete the list of commands from Doskey. Then type the commands you want to save. Use the **/history** switch to save the commands in a file with a .BAT extension.

For more information about using the **/history** switch, see "Saving Macros" later in this chapter. For more information about batch programs, see Chapter 10, "Working with Batch Programs."

Using Doskey to Work with Macros

A macro is a set of commands that you can carry out by typing the name of the macro. A macro is very much like a batch program. Both contain sets of commands that you carry out by typing a name. A macro differs from a batch program in the following ways:

- A macro is stored in RAM, whereas a batch program is stored on disk. Because macros are stored in RAM, they run much faster than batch programs and you can run them from any directory. On the other hand, when you reset or restart your system, the macros are lost, whereas batch programs on disk are not. Macros also take away memory from your command history buffer.

- You create a batch program by saving commands in a file. You create a macro by putting commands in a macro definition. In a batch program, you put each command on a separate line. There is no limit to the number of commands you can put in a batch program. In a macro, you type all the commands on the same line and separate each with a special character. The total length of a macro cannot exceed 127 characters.

- You can stop running a batch program by pressing CTRL+C once. In a macro, you must press CTRL+C for every command in the macro. Each time you press CTRL+C in a macro, MS-DOS stops the command it is currently carrying out.

- In both batch programs and macros, you can use replaceable parameters. In batch programs the parameters are **%1** through **%9**, and in macros they are **$1** through **$9**. In addition, the redirection characters you use in macros are different from those you use in batch programs and MS-DOS commands.

- In a batch program, you can use the **goto** command to jump to another location in the program. In addition, you can start other batch programs from inside a batch program. You cannot use **goto** commands in a macro or start other macros from inside a macro.

- You can run a batch program from inside a macro, but you can't run a macro from inside a batch program. However, you can include a command that *creates* a macro inside a batch program.

- In a macro, you can't use the **echo off** command to prevent commands from being displayed as MS-DOS carries them out.

- You can define an environment variable in a macro; however, you cannot use the environment variable in the macro. For example, you could set ABC to the C:\DOS directory (**set abc=c:\dos**), but you could not use the environment variable ABC with any commands (for example, you could not use **dir abc**). You could do this in a batch program. Also, you cannot use other environment variables that may have been defined elsewhere. For more information about environment variables, see the **set** command in Chapter 14, "Commands."

Although you cannot run a macro from inside a batch program, you can define a macro from within a batch program. Because you must run a macro definition command to load a macro into memory, the most useful way to use macros is to put the definition commands for the macros you commonly use into one batch program. To make the macros available, you run the batch program.

Creating a Macro

To create a macro, you type **doskey** followed by the name of the macro, an equal sign, and the commands in the macro. For example, you could use the following command to create a macro called **ddir** that displays a directory in wide format:

```
doskey ddir=dir /w
```

If you then type **ddir** at the command prompt, MS-DOS runs the macro, displaying a five-column list of the files in the current directory. Because the macro is stored in memory, it doesn't matter which directory is current when you run it.

To include more than one command in a macro, separate the commands with a dollar sign ($) and the letter T. For example, the following command creates a macro called **cmp** that alphabetizes and lists the .DOC files and then the .BAK files in the current directory:

```
doskey cmp=dir *.doc /o:n $t dir *.bak /o:n
```

While you are creating and testing a macro, it is easiest to type the command that defines the macro at the command prompt. Then, you can use the Doskey editing keys to change and redefine the macro quickly.

Because macros are stored in memory, they are lost when you turn your system off or reset it. Thus, if you create a macro that you use often, put the command that defines the macro in your AUTOEXEC.BAT file so the macro is available each time MS-DOS starts.

Running a Macro

To run a macro, you type its name at the command prompt. For example, to run the **ddir** macro, type the following command at the command prompt:

```
ddir
```

If the macro has parameters, leave a space between the macro name and the parameters. For example, suppose you create a macro called **move** that has as parameters the name of the file you want to move and the name of the directory that you want to move the file to. To move all .TXT files from the current directory to the C:\TXTFILES directory, you would type the following command:

```
move *.txt c:\txtfiles
```

There cannot be any space between the command prompt and the macro name. If there is, MS-DOS does not recognize the name and displays the following message:

```
Bad command or file name
```

Suppose you want to create a macro that has the same name as a command. For example, you would use the following command to create a macro called **dir**, which would replace the MS-DOS **dir** command:

```
doskey dir=dir /w
```

When you have a macro with the same name as a command, MS-DOS runs the macro rather than the command. Thus, when you type **dir** at the command prompt, MS-DOS runs the **dir** macro rather than the **dir** command.

Whenever you want to use the **dir** command instead of the **dir** macro, you can type a space between the command prompt and **dir**. Now, MS-DOS does not recognize **dir** as a macro name, but it does recognize it as a command.

NOTE You cannot run a macro from within a batch program, but you can define it in a batch program. For more information about batch programs, see Chapter 10, "Working with Batch Programs."

Editing a Macro

You can change a macro by editing the command that created it. If the macro is defined in a batch program, you can edit the batch program and then run it again. If the macro is one of the commands that Doskey has stored, you can redisplay the macro command, edit it by using Doskey editing keys, and complete the edit by pressing ENTER. For information about Doskey editing keys, see "Editing and Using Previous Commands" earlier in this chapter.

Saving a Macro

To save macros stored in memory, use the **doskey** command with the **/macros** switch, a greater-than sign (>), and a filename. In the following example, the names and contents of the macros currently in memory are stored in the MACS.BAT file:

```
doskey /macros > macs.bat
```

If you add the **doskey** command to the beginning of each macro that you saved in the MACS.BAT file, you can load the macros into memory by running the batch program. For example, suppose you created the following three macros and saved them in the MACS.BAT file:

```
ddir=dir /oe /p
mv=copy $1 $2 $t del $1
where=dir /s /p $1:\*.$2
```

If you want these macros to be available each time you start your system, first add the **doskey** command to them, as follows:

```
doskey ddir=dir /oe /p
doskey mv=copy $1 $2 $t del $1
doskey where=dir /s /p $1:\*.$2
```

Each time you run the batch program, Doskey loads the three macros into memory. You may want to run the batch program from your AUTOEXEC.BAT file by using the **call** command. For information about the AUTOEXEC.BAT file, see Chapter 11, "Customizing Your System."

Deleting a Macro

To delete a macro, type **doskey** followed by the name of the macro you want to delete, and an equal sign. For example, to delete the **ddir** macro, you would type the following command:

```
doskey ddir=
```

Doskey removes the macro from memory. To delete all macros, press ALT+F10.

NOTE Deleting macros makes the memory available for other macros, but it does not return the memory to the command-history buffer.

Using Replaceable Parameters

You can use replaceable parameters in a macro in much the same way you use them in a batch program. In a macro, the replaceable parameters are **$1** through **$9** rather than **%1** through **%9**.

For example, the following command creates a macro named **findit** that searches through the directories on drive C for filenames that match the one you specify:

```
doskey findit=dir c:\$1 /s
```

The **/s** switch is used to display filenames from all directories on drive C (including the current directory) that match the filename you specify.

To run this macro, type **findit** and a filename at the command prompt. For example, to locate all files on drive C that have the extension .OLD, you would type the following command:

```
findit *.old
```

Doskey substitutes the filename you type for the **$1** in the macro. The resulting command looks like this:

```
dir c:\*.old /s
```

You can use the same parameter more than once in a macro. For example, the following command creates a macro called **ddel**. This macro moves a file to a directory named DELETED on drive C:

```
doskey ddel=copy $1 c:\deleted $t del $1
```

When you run the **ddel** macro, you type the name of the file that is to go into the C:\DELETED directory. Doskey replaces the **$1** in the macro with the filename. It copies the file to DELETED and then deletes it from its original directory.

To delete the files in the C:\DELETED directory, you could use this macro named **cleanup**:

```
doskey cleanup=dir c:\deleted $t del c:\deleted\*.*
```

The macro displays a list of files in the C:\DELETED directory and then starts the **del** command. Because the **del** command prompts you to confirm deletion of all the files, you have a chance to review the filenames before deleting any files.

Using the $* Replaceable Parameter

You can use the **$*** replaceable parameter to assign to a single parameter all of the text following the command that starts a macro. Typically, MS-DOS distinguishes parameters by looking for a space. The text between the first two spaces is the first parameter, the text between the second and third spaces is the second parameter, and so on. If you use the **$*** parameter, Doskey ignores spaces and assigns all text to the **$*** parameter.

The **$*** parameter is most useful when the macro you create uses a variable number of parameters. For example, you can use the following command to create a macro named **d** that allows you to abbreviate the **dir** command:

```
doskey d=dir $*
```

This macro works exactly like the **dir** command, regardless of the number of parameters you specify. For example, all of the following commands are carried out in the same way with the **d** macro as they are with the **dir** command:

```
d *.txt
d *.txt /s
d *.txt /s /b
```

If you use the **$1** parameter instead of **$*** with the macro, MS-DOS substitutes the first parameter and ignores the rest of the command line.

Redirecting Input and Output

You redirect input and output in macros the same way you do in MS-DOS commands. The only difference is that macros require different characters:

$L (or **$l**)	Equivalent to the less-than sign (<). It redirects the input to a command.
$G (or **$g**)	Equivalent to the greater-than sign (>). It redirects the output of a command.
GG (or **gg**)	Equivalent to the double greater-than sign (>>). It appends output onto the end of a file.
$B (or **$b**)	Equivalent to the pipe (¦). It redirects output from one command to another.

For example, the following command creates a macro named **pdir** that prints directory listings:

```
doskey pdir=dir $g lpt1:
```

The following command creates a macro named **mtype** that displays the contents of the file you specify and pauses between each screen of information:

```
doskey mtype=type $1 $b more
```

The following command creates a macro named **asort** that alphabetizes the file and stores the information in a different file:

```
doskey asort=sort $L $1 $g $2
```

To run this macro, you type the filename you want to sort. Doskey replaces the **$1** replaceable parameter with the first filename you specify. The **$L** redirects the file to the **sort** command. The **$g** and **$2** parameters redirect the output of the **sort** command to the second file you specify.

For example, after you create the **asort** macro, you could type the following:

```
asort input.txt output.txt
```

This is the same as typing:

```
sort < input.txt > output.txt
```

When defining a Doskey macro, you must mark the dollar sign ($) when it occurs anywhere other than in parameters, command separators, and redirection characters. You mark it by typing two dollar signs rather than one.

For example, suppose your macro copies a file to the $&CENTS directory. When you type the name of the directory, you must use **$$&cents** in your macro definition. As the command is carried out, the dollar sign is assumed to be a standard character rather than a marker or parameter.

Chapter 8

Customizing MS-DOS Shell

8

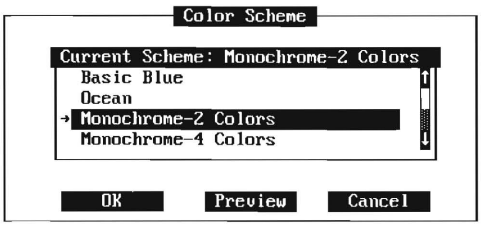
You can customize MS-DOS Shell in several ways. You can change the screen colors or change the way information is displayed in the MS-DOS Shell window. You can also organize your programs into groups and display the groups graphically, making it easier to find and use your programs. When you add a program item to a group, you can further customize the program by creating your own Help text, controlling the memory needed to run the program, and defining application shortcut keys.

Changing Screen Colors

There are several color schemes available for MS-DOS Shell.

▶ To choose a color scheme:

1. From the Options menu, choose Colors.
   The Color Scheme dialog box appears.



2. To select the scheme you want, click the scroll arrows until the color scheme you want comes into view, and then click that color scheme. Or use the UP ARROW or DOWN ARROW key to select the color scheme you want.

3. If you want to see what the selected color scheme looks like on your screen, choose the Preview button.
4. Choose the OK button to implement the color scheme.

Switching Between Text and Graphics Mode

The appearance of MS-DOS Shell on your screen depends on what type of *display adapter* you have and the *screen mode* you are using. The screen mode controls the size and shape of the images that appear on your screen. A display adapter determines the screen display's capabilities, such as resolution and screen mode.

There are two types of screen modes: text and graphics. Only some display adapters support graphics mode. All display adapters support text mode, which is the mode MS-DOS Shell uses the first time you start it.

Within text or graphics mode, you usually have a choice of how many lines you want displayed. For example, if you choose to view 50 lines instead of 25 lines (the default), you see more information on your screen at once, but the words and symbols appear smaller.

▶ To change the screen mode:

1. From the Options menu, choose Display.

   The Screen Display Mode dialog box appears.
2. Select the screen mode you want.
3. If you want to see what the selected screen mode looks like on your screen, choose the Preview button.
4. Choose the OK button to implement the screen mode.

   MS-DOS Shell appears in the screen mode you selected.

Organizing Programs

You can organize programs into program groups to suit your needs. When you create a program group, you give it a title, which appears in the program list. (In the default view, Program/File Lists, the program list appears in the lower left corner of the MS-DOS Shell window.) To work with a group, you choose its title.

Adding and Deleting Groups

You can add groups to the Main group, the Disk Utilities group, or a group you have created. For example, you might use three programs to keep track of your finances—one to maintain your checkbook, one to estimate taxes, and one to track your monthly bills. You could add a group named Account and put these programs into it.

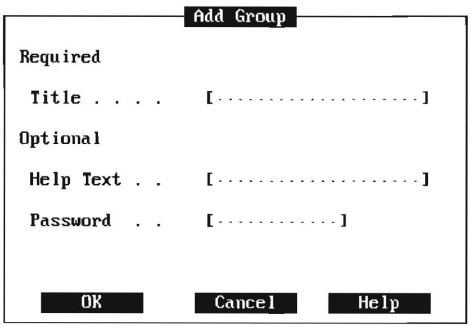
When you create a group, you must give it a title. You can also give it a password that a user must know to view the group, and a Help message that provides information on the group and its program items.

▶ To add a group:

1. Make sure you have selected Program/File Lists on the View menu.
2. Press TAB to move to the program-list area of your screen. Or click anywhere inside the program-list area.
3. If the group to which you want to add the new group is not open, open it by double-clicking the group name. Or use the UP ARROW or DOWN ARROW key to select the group and then press ENTER.
4. From the File menu, choose New.

   The New Program Object dialog box appears.
5. Select Program Group.
6. Choose the OK button.

   The Add Group dialog box appears.



7. In the Title box, type a title for the new group.

8. If you want the group to have a Help message, type up to 255 characters (including blanks) in the Help Text box.

   For example, you might type a Help message that reads, "Use the programs in this group to perform statistical operations." When you select this group and press F1, MS-DOS Shell displays the message. The message appears exactly as you have typed it and is formatted to fit in the Help dialog box. If you want a line break to appear in the text, type **^m** (a caret followed by the letter M) at the point where you want the new line to start.

9. If you want the group to have a password, type the password in the Password box.

10. Choose the OK button.

To delete a group:

1. Select the group you want to delete.
2. From the File menu, choose Delete.

   Or press DEL.

   The Delete Item dialog box appears.
3. Choose the OK button.

When you delete a group, MS-DOS Shell removes the group name from the program list and deletes the group's password and Help message.

Changing the Contents of a Group

You can change the contents of a group by adding program items, copying program items from another group, reordering items, and deleting items.

Adding a Program Item

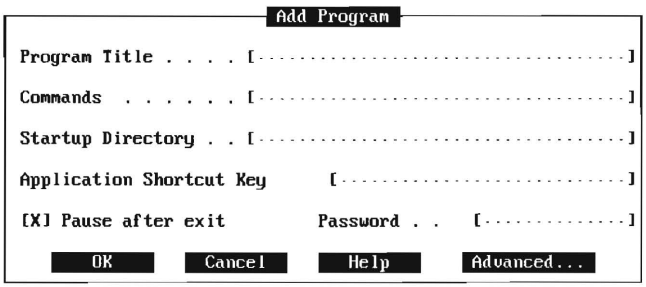
When you add a program item to a group, you give it a title and specify the command that starts the program. In addition to assigning titles and startup commands, you can associate a variety of other information with a program item. For more information, see "Working with Properties" later in this chapter.

To add a program item to a group:

1. Make sure you have selected Program/File Lists on the View menu.
2. Press TAB to move to the program-list area of your screen. Or click anywhere in the program-list area.
3. If the group to which you want to add the new item is not open, open it by double-clicking the group name. Or use the UP ARROW or DOWN ARROW key to select the group, and then press ENTER.
4. From the File menu, choose New.

   The New Program Object dialog box appears, with New Program Item already selected.
5. Choose the OK button.

   The Add Program dialog box appears.



6. In the Program Title box, type the *program-item title* that you want to appear in the program list.
7. In the Commands box, type the *startup command*, the command that starts the program. If the command is not in the current directory or in a directory specified by the PATH environment variable, or if it is not an internal command, you must include the complete path of the file in the command.

   You may include more than one command in this box. For more information about startup commands, see "Specifying a Startup Command" later in this chapter.

8. Specify optional information you want to associate with the program item. You can specify the following:

   - A startup directory, which MS-DOS Shell changes to before starting the program.
   - An application shortcut key, which, once you have started a program, you can use to switch to the program from other programs or from MS-DOS Shell.
   - Pause after exit, which prompts you to press any key to return to MS-DOS Shell after the program has finished running.
   - A password, which will be required before starting the program item.

   For more information about these options, see "Working with Properties" later in this chapter.

9. If you want to specify other options, choose the Advanced button.

   The Advanced dialog box appears.

   For information about the options in the Advanced dialog box, see "Specifying Advanced Properties" later in this chapter.

10. Choose the OK button. (If you chose the Advanced button, you return to the Add Program dialog box when you choose OK. You must then choose OK in that box, also.)

Copying a Program Item to Another Group

To copy a program item to another group, you choose the Copy command on the File menu. For example, if you have a Microsoft Excel program item in your Account group, you can also put it in your Tax group. You can copy a program item to as many groups as you like.

▶ To copy a program item from one group to another:

1. Select the program item you want to copy.
2. From the File menu, choose Copy. (Instructions appear in the status bar.)
3. Open the group you want to copy the program item to.

   If the group you are copying to has a password, the Password dialog box appears. Type the password, and choose the OK button.

4. Press F2.

Reordering Items in a Group

To move a program item or group title from one position in a group to another, use the Reorder command.

▶ To reposition a program item or group title:

1. Select the program item or group title you want to reposition.
2. From the File menu, choose Reorder. (Instructions appear in the status bar.)
3. Double-click the new location.

   Or move the selection cursor to the new location, and then press ENTER.

   The selected program item or group title moves to the new location.

Deleting a Program Item from a Group

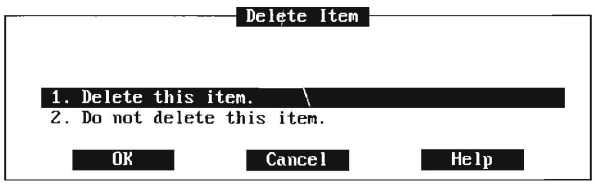
You can delete a program item that you no longer need. Deleting a program item from a group does not delete the program file from the directory that contains it.

▶ To delete a program item from a group:

1. Select the program item you want to delete.
2. From the File menu, choose Delete.

   Or press DEL.

   The Delete Item dialog box appears. If the program item has a password associated with it, you'll see a warning message, but you can still delete the program item.



3. Choose the OK button.

Working with Properties

A *property* is a piece of information that you associate with a program item. You can specify numerous properties for each program item.

The following two properties are required:

- Program-item title
- Startup command

These properties are optional:

- Startup directory
- Application shortcut key
- Pause after exit
- Password
- Additional advanced properties

When you create a program item, you must specify two properties in the Add Program dialog box: program-item title and startup command. You can also specify any of the optional properties. For information about creating program items, see "Adding a Program Item" earlier in this chapter.

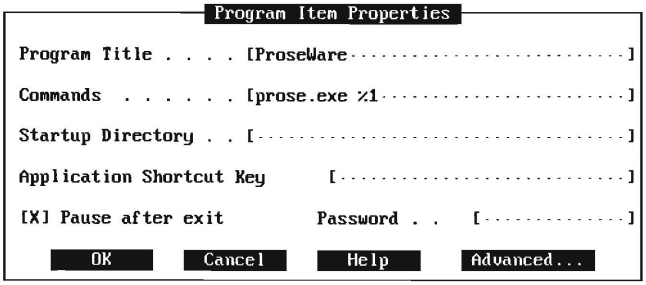
After you create a program item, you can change its properties by using the Properties command.

▶ To change the properties of a program item:

1. Select the program item.
2. From the File menu, choose Properties.

   If the program item has a password, the Password dialog box appears. Type the password, and choose the OK button.

   The Program Item Properties dialog box appears. The Program Title box and the Commands box will show the properties that have been specified for the program item you selected, along with any optional properties.



The Program Item Properties dialog box is the same as the Add Program dialog box, which is displayed when you create a program item.

3. Type the information for the properties you want to change.
4. Choose the OK button.

Specifying a Startup Command

When you create a program item, you must type a startup command in the Commands box of the Add Program dialog box. If the command is not in the current directory or in a directory specified by the PATH environment variable, or if it is not an internal command, you must include the complete path of the program file. For example, if you are creating a program item named WordPerfect and the path for WordPerfect is C:\WP\WP.EXE, you would type that path in the Commands box.

You can also specify additional commands, run batch programs, and include replaceable parameters for the program. The following sections describe how to include these options.

Specifying Additional Commands

MS-DOS Shell carries out each command in the order it appears in the Commands box. Each command must be separated by a semicolon (;). There must be one or more spaces on each side of the semicolon. Text in the Commands box cannot exceed 255 characters.

For example, suppose you want to put an abbreviated list of the files in a directory into a text file, load the file into your text editor, edit the file, save it under a different name, and delete the original file when you are finished. If you are using MS-DOS Editor, your startup command might look like this:

```
dir /b  > tmp.txt ; c:\dos\edit.com tmp.txt ; del tmp.txt
```

In this example, MS-DOS Shell first stores the names of files in the current directory in a file named TMP.TXT. Then it runs MS-DOS Editor, loading the TMP.TXT file. When you quit MS-DOS Editor, the TMP.TXT file is deleted, and you return to MS-DOS Shell.

Running Batch Programs in a Startup Command

You can run batch programs by including **call** commands in the startup command. For example, suppose you want to run a batch program named PREP.BAT before you start WordPerfect, and one called POST.BAT after you quit WordPerfect. You would type the following in the Commands box:

```
call prep ; wp ; call post
```

For information about batch programs, see Chapter 10, "Working with Batch Programs."

Using Replaceable Parameters

A *parameter* is additional information you give a program when you start it. For example, when you start MS-DOS Editor from MS-DOS Shell, the File to Edit dialog box appears. You can specify a filename in the Text box. If you type **recipes.txt**, for example, MS-DOS Editor loads the file RECIPES.TXT as soon as it starts.

Many programs accept parameters in this manner. If the program item you add to a program group accepts parameters, you can include these parameters in the Commands box.

If you want to be able to specify a different parameter whenever you run the program item, you can put a replaceable parameter in the Commands box. Each time you choose the program item, MS-DOS Shell displays a dialog box that prompts you to fill in the value for the replaceable parameter before the program starts.

In the Commands box, you indicate a replaceable parameter with the percent sign (**%**) followed by a numeral (**1** through **9**). For example, if you want MS-DOS Shell to prompt you for a filename when you start WordPerfect, you might type the following in the Commands box:

```
c:\wp\wp.exe %1
```

The **%1** in this example indicates that you want MS-DOS Shell to prompt you to type a value in place of **%1** every time you choose the program item for WordPerfect. MS-DOS Shell will prompt you by displaying a prompt dialog box.

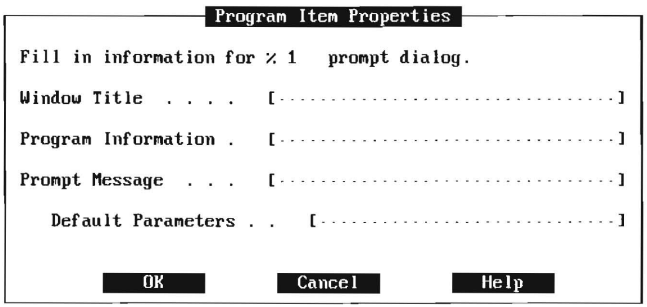
▶ To include a replaceable parameter in a startup command:

1. Select the program item.
2. From the File menu, choose Properties.

   If the program item has a password, the Password dialog box appears. Type the password, and choose the OK button.

   The Program Item Properties dialog box appears.
3. In the Commands box, specify the command and indicate the replaceable parameters by typing a percent sign (**%**) followed by a numeral (**1** through **9**) for each.
4. Choose the OK button.

   A Program Item Properties or Add Program dialog box appears for each replaceable parameter you have specified.



5. The information you type in this dialog box will create a customized dialog box for the program item that you are adding or changing. Type the information that you want to appear in the dialog box. Each time you choose the program item, MS-DOS Shell will display the information you have typed.

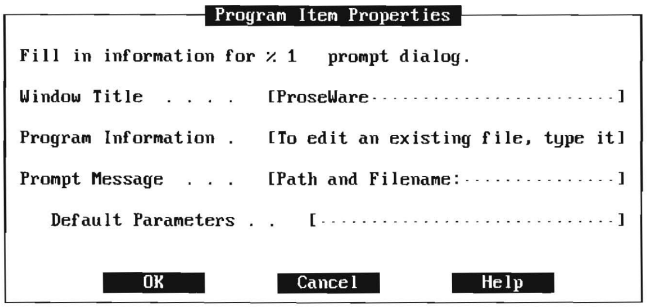
   The Window Title you supply will appear at the top of the dialog box. The Program Information you supply will appear under the title. (You can type up to 106 characters in the Program Information box.) The Prompt Message you specify will appear to the left of the box where you specify the parameter value.

   If you specify a value in the Default Parameters box, the value will appear in the prompt dialog box. You can accept the default parameter or change it. Specify a default parameter value if you plan to use that value frequently when you run the program. For example, if you are working on a project that often

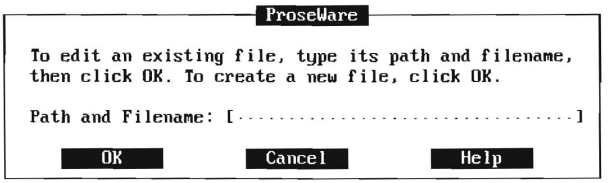
requires you to use a particular document, you might type that document's name as the default parameter for a text editor.

There are two special parameters that you can use in the Default Parameters box to automatically set a default parameter. The **%f** parameter sets the default filename to the filename currently selected in the file list. The **%l** parameter (a percent sign followed by the letter L) sets the default parameter to the parameter that was specified the last time the program item was run.

If you want MS-DOS Shell to prompt you for a file to load when you start a text editor, you could specify values such as the following in the Program Item Properties dialog box:



Each time you select the program item, MS-DOS Shell will prompt you with the following prompt dialog box:



Using the Same Replaceable Parameter More Than Once

You can use the same replaceable parameter more than once in a Commands box. For example, suppose you create files by using WordPerfect, and you store them in C:\EDIT\WP. Suppose that as you create these files, you always back them up on a disk in drive A. To load a file into WordPerfect and then back up the file (onto a disk in drive A) after you have edited it, you would type the following in the Commands box:

```
c:\edit\wp\wp.exe %1 ; copy %1 a:
```

Using More Than One Replaceable Parameter

You can include up to nine different replaceable parameters in the Commands box. For example, suppose you want MS-DOS Shell to prompt you for a file to load with WordPerfect and for a backup directory to copy the file to when you finish editing. You would include two different replaceable parameters in the Commands box, as in the following example:

```
c:\wp\wp.exe %1 ; copy %1 %2
```

If you add this command in the Commands box, you will be prompted to provide the name of the file you want to edit in addition to the directory to which you want to copy the file. Then you can edit your WordPerfect document.

For each replaceable parameter, MS-DOS Shell prompts you for information to appear in the prompt dialog box, as described previously.

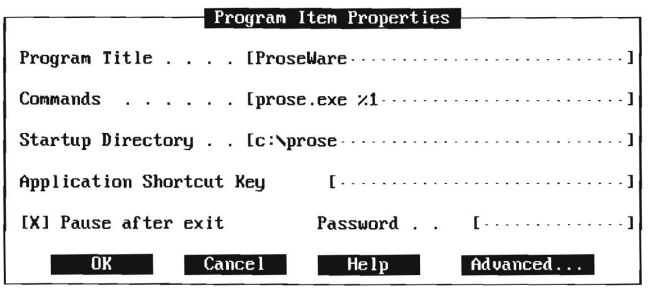
Specifying a Startup Directory

You can specify which directory you want MS-DOS to change to before it starts a program that is in a program group. For example, if you keep budget spreadsheets in a directory named C:\FILES, you could make sure MS-DOS Shell changes to that directory before starting your spreadsheet program.

▶ To specify a startup directory:

1. Select the program item whose properties you want to change.
2. From the File menu, choose Properties.

   The Program Item Properties dialog box appears.
3. In the Startup Directory box, type the drive and path of the directory you want MS-DOS Shell to change to before it starts the program, as in the following example:



Specifying an Application Shortcut Key

If you have started a program but are not currently working with it, you can use a key combination to switch quickly to it from another program or from MS-DOS Shell. The shortcut key must have the form CTRL+*character*, SHIFT+*character*, or ALT+*character*, where *character* is a letter, number, or function key on your keyboard. (Exceptions are noted later in this section.) You can use any combination of the CTRL, SHIFT, and ALT keys with the character.

▶ To specify an application shortcut key:

1. Select the program item you want.
2. From the File menu, choose Properties.

   The Program Item Properties dialog box appears.
3. In the Application Shortcut Key box, specify the key combination by pressing and holding down CTRL, SHIFT, or ALT, and then pressing a character.

For example, suppose you have a program item named My Editor in the Main group and that you have enabled Task Swapper. You could assign CTRL+E as the shortcut key for My Editor. If My Editor is on the Active Task List, you can press CTRL+E to switch back to it from another program or from MS-DOS Shell.

The name of the shortcut key will appear next to the program-item title in the Active Task List. For information about Task Swapper and the Active Task List, see Chapter 3, "MS-DOS Shell Basics."

The following key combinations are reserved and are not available as application shortcut keys:

CTRL+M	SHIFT+CTRL+M
CTRL+I	SHIFT+CTRL+I
CTRL+H	SHIFT+CTRL+H
CTRL+C	SHIFT+CTRL+C
CTRL+[	SHIFT+CTRL+[
CTRL+5 (on the keypad)	SHIFT+CTRL+5 (on the keypad)

Specifying Whether to Pause After a Program Ends

You can specify whether MS-DOS Shell should pause after you quit a program that is in a program group. By default, MS-DOS Shell prompts you to press a key to return to MS-DOS Shell after you quit such a program.

▶ To eliminate the pause after you quit a program:

1. Select the program item you want.
2. From the File menu, choose Properties.
   The Program Item Properties dialog box appears.
3. Clear the Pause After Exit option by clicking the X.
   Or press TAB until you select the option, and press the SPACEBAR to clear it.
4. Choose the OK button.

Specifying a Password

If you want MS-DOS Shell to prompt you for a password before it starts a program item, you can specify the password in the Program Item Properties dialog box.

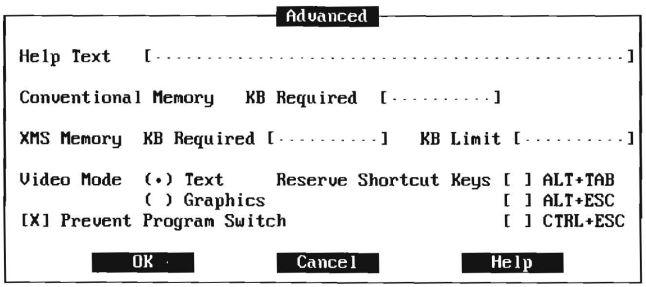
▶ To specify a password for a program item:

1. Select the program item.
2. From the File menu, choose Properties.

   The Program Item Properties dialog box appears.
3. In the Password box, type the password you want.
4. Choose the OK button.

Specifying Advanced Properties

When you choose the Advanced button in the Add Program or Program Item Properties dialog box, another dialog box appears in which you can specify additional properties. The Advanced dialog box looks like this:



Adding Help Text

You can add a Help message of up to 255 characters to any program item. This Help message is displayed if you press F1 when the program item is selected. MS-DOS Shell displays the message exactly as you have typed it and formats it to fit in a Help dialog box. If you want to begin a new line of text, type **^m** (a caret followed by the letter M) at the point where you want the new line to start.

If you do not want the program item to have a Help message, leave the Help Text box blank.

Specifying Conventional Memory

Use the Conventional Memory KB Required box to specify how many kilobytes of conventional memory must be free in order to start a program. This property is useful for programs with specific memory requirements.

Regardless of what you type in the Conventional Memory KB Required box, when you start a program, MS-DOS Shell gives it all available conventional memory. The number you type in this box determines how much memory must be available before MS-DOS Shell starts the program; it does not limit how much conventional memory the program receives.

If MS-DOS Shell cannot provide as much memory as you specify, a message appears, telling you there is not enough memory to run the program.

If Task Swapper is not enabled, MS-DOS Shell ignores any conventional-memory specification.

Specifying Extended Memory

Use the XMS Memory options to specify how much extended memory to give to a program that uses memory according to the Lotus/Intel/Microsoft/AST Extended Memory Specification (XMS) standard. If Task Swapper is not enabled, the XMS memory specifications are ignored.

A description of each XMS Memory option follows. Before using these options, you must have extended memory set up on your system. For more information, see the discussion about extended memory in Chapter 12, "Optimizing Your System."

KB Required

Specifies how many kilobytes of extended memory must be free in order to run a program.

Leave this setting blank for most programs. Specifying a value significantly increases the time it takes to switch to and from a program. You should specify a value only if a program requires a certain amount of extended memory in order to run.

If you run a program that requires extended memory, and MS-DOS Shell cannot provide as much memory as you specify, a message appears when you try to start the program, telling you there is not enough memory. However, if Task Swapper is not enabled, the memory specification is ignored.

KB Limit

Specifies the maximum amount (in kilobytes) of extended memory that MS-DOS Shell can give to a program.

This option is useful for limiting a program's access to extended memory, since some programs take all available extended memory whether they need it or not. If Task Swapper is not enabled, the KB Limit specification is ignored.

Leave this setting blank to prevent a program from gaining access to any extended memory.

Setting this option to –1 gives the program all the extended memory it requests (up to the maximum amount available). Set this option to –1 only if the program requires large amounts of extended memory.

Using Video Mode

Video Mode has two options: text and graphics. You should use text mode unless you are having trouble switching to a program.

Usually the memory reserved by text mode is enough, but you may need more memory if you are using a CGA monitor. Graphics mode requires more memory than text mode. Use text mode for all program items if you have a high-resolution graphics (VGA or EGA) monitor or a monochrome monitor.

Reserving Shortcut Keys

Use the Reserve Shortcut Keys option when you want a program to use shortcut keys typically used by Task Swapper (ALT+TAB, ALT+ESC, and CTRL+ESC). For information about Task Swapper, see Chapter 3, "MS-DOS Shell Basics." If you want to reserve a shortcut key for a program, select it in the Reserved Shortcut Keys area of the Advanced dialog box.

For example, suppose you have a text editor that typically uses the ALT+TAB shortcut key to insert a special character. If you want to maintain that function while your text editor is running under MS-DOS Shell, you need to select the shortcut key for the program. When you select this key, it is no longer reserved for MS-DOS Shell.

Preventing Program Switching

If you want to prevent a program from switching to another program or to MS-DOS Shell, select the Prevent Program Switch option from the Advanced

dialog box. If you select this option, you must quit the program to return to MS-DOS Shell.

Changing Group Properties

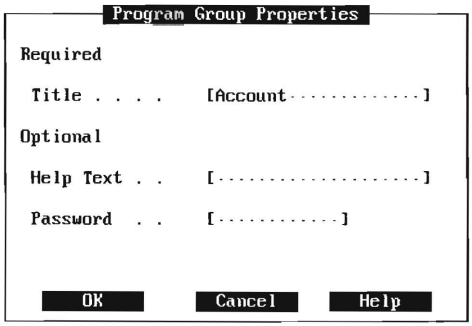
A program group must have a title associated with it. It can also have Help text and a password. You can change any of these properties for all groups except the Main group.

▶ To change group properties:

1. Select the appropriate group.
2. From the File menu, choose Properties.

   If the group has a password, the Password dialog box appears. Type the password, and choose the OK button.

   The Program Group Properties dialog box appears.



3. Change the title, Help text, or password.
4. Choose the OK button.

Chapter 9

Working with MS-DOS Editor

9

MS-DOS Editor is a text editor you can use to create, edit, and print memos, letters, and special files that customize MS-DOS. You can use your keyboard or a mouse.

Files you create by using MS-DOS Editor are unformatted text files. Because MS-DOS batch programs and files such as AUTOEXEC.BAT and CONFIG.SYS must be unformatted text files, MS-DOS Editor is a useful tool for customizing your system.

When you use MS-DOS Editor, you can do the following:

- Choose commands from menus and specify information and preferences in dialog boxes
- Select text and move, copy, or delete it
- Find and replace text you specify
- Use Help to find information about MS-DOS Editor procedures and commands

Edlin, the line editor included in previous versions of MS-DOS, is also available in MS-DOS version 5.0. For information about specific Edlin commands, see the **edlin** command in Chapter 14, "Commands."

CAUTION MS-DOS Editor does not work if the file QBASIC.EXE is not in the search path, the current directory, or in the same directory as the file EDIT.COM. If you delete QBASIC.EXE to save space on your hard disk, you will not be able to use MS-DOS Editor.

Getting Started with MS-DOS Editor

There are two ways to start MS-DOS Editor: at the command prompt and in MS-DOS Shell.

▶ To start MS-DOS Editor at the command prompt:

- Type **edit** at the command prompt. If you want to open an existing text file, include its path and filename, as in the following example:

```
edit a:\work\program.txt
```

MS-DOS Editor starts with the PROGRAM.TXT file loaded.

▶ To start MS-DOS Editor from MS-DOS Shell:

Mouse

1. In the Main group, double-click Editor. The File to Edit dialog box appears.
2. To create a new file, click OK.

   To edit an existing file, type its path and filename in the File to Edit box, and then click the OK button.

Keyboard

1. Press TAB to move to the Main group.
2. Use the UP ARROW or DOWN ARROW key to move the selection cursor to Editor, and then press ENTER. The File to Edit dialog box appears.
3. To create a new file, choose the OK button.

   To edit an existing file, type its path and filename in the File to Edit box, and then choose the OK button.

If you type only the word **edit** at the command prompt, or choose the OK button in the File to Edit box without specifying a filename in MS-DOS Shell, the following screen appears:



To get information about MS-DOS Editor when this screen appears, press ENTER or click the phrase, "Press Enter to see the Survival Guide." An introduction to MS-DOS Editor and its Help system is then displayed.

To start working with MS-DOS Editor, press ESC or click the phrase, "Press ESC to clear this dialog box."

An empty window appears. You work on text files by typing or editing text in the window.

If you specify a filename when you start MS-DOS Editor, your file appears instead of the dialog box. To get information about MS-DOS Editor after a file has been loaded, press F1.

Working with Menus

To perform tasks in MS-DOS Editor, you choose commands from *menus* on the *menu bar*. The menu bar is displayed at the top of the MS-DOS Editor window. Before choosing a command, you select the menu that contains the command.

▶ To select a menu:

Mouse

- Click the name of the menu you want.

Keyboard

1. Press ALT.

   The first menu name, File, is selected.

2. Press the LEFT ARROW or RIGHT ARROW key to move the selection cursor to the menu you want, and then press ENTER.

   Or type the first letter of the menu name.

▶ To choose a command:

Mouse

- Click the command you want.

Keyboard

- Press the UP ARROW or DOWN ARROW key to select the command you want, and press ENTER.

  Or press the highlighted letter in the command name.

  If a command does not have a highlighted letter, you must perform an action before you can use it. For example, you must select some text before you can use the Cut command.

  You can also choose some commands by using shortcut keys. If a command has a shortcut key, it is displayed next to the command name on the menu.

▶ To cancel a menu without choosing a command:

Mouse

- Click anywhere outside the selected menu.

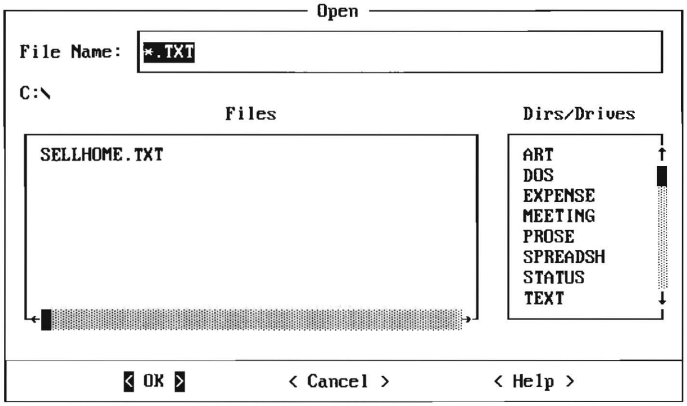
Keyboard

- Press ESC.

Working with Dialog Boxes

When you choose some commands, a *dialog box* appears. You type additional information or select options in a dialog box.

For example, when you choose the Open command from the File menu, the Open dialog box appears:



Most dialog boxes have areas where you can type information or select options. You can move from one area to another by pressing TAB. (You can also use SHIFT+TAB to move in reverse order.) If you're using a mouse, click the area you want to work in.

Most dialog boxes contain one or more of the following items:

- A text box in which you can type text. If the box you select already contains text, you can clear it by typing characters or by pressing the SPACEBAR.
- A group of options with a dot next to the currently selected option. You select an option by clicking between the parentheses next to the option you want, or moving the dot by using the arrow keys.

- Check boxes you can select or clear. To select a check box, click it; or use TAB to move the cursor to the check box, and then press the SPACEBAR. To clear a check box, click it again; or press the SPACEBAR again.
- A list of files, directories, and drives.
- Command buttons, such as OK, Cancel, and Help. They are located along the bottom of the dialog box.

  The OK button carries out the command and closes the dialog box. When the angle brackets around the OK button are highlighted, you can press ENTER to carry out the command, or click the OK button.

  The Cancel button cancels the command and closes the dialog box. There are three ways to cancel a command: by using TAB to select the Cancel button and then pressing ENTER; by pressing ESC; or by clicking the Cancel button.

  The Help button displays information about the command. There are three ways to get Help: by using TAB to select the Help Button and then pressing ENTER; by pressing F1; or by clicking the Help button.

▶ To select an item from a list in a dialog box:

Mouse

- Click the item you want.

  If the item you want is not visible, click the *scroll arrows* until it comes into view. You can also drag the *scroll box* in the *scroll bar*, or click the shaded areas of the scroll bar.

Keyboard

1. Press TAB to move to the list.
2. Use the arrow keys to move the selection cursor to the item you want.

   You can also use the HOME, END, PAGE UP, and PAGE DOWN keys to move the selection cursor.

Using Help

There are several types of Help available in MS-DOS Editor. The following sections describe how to use the different kinds of Help.

Status Bar

One type of Help, which is always available at the bottom of your screen, is a status bar that displays information about commands or keys that you can use. When you choose a command, the status bar displays a short description of it. If you need more information about the command, press F1.

The status bar changes when you select a menu, choose a command, or open a dialog box. It also displays information about the functions of various keys.

Use this Help when you need a quick reminder about the purpose of a command or you need a quick reference to the keyboard.

Command, Menu, and Dialog Box Help

You can get Help on how to use a specific command, menu, or dialog box by selecting the item and pressing F1. You can cancel Help by pressing ENTER or ESC.

In a dialog box, you can also get Help by clicking the Help button.

Use this Help when you need to know how to use a specific command, menu, or dialog box.

Survival Guide

MS-DOS Editor also has Help for getting started with MS-DOS Editor and Help for using the keyboard to navigate through text and Help. This Help is called the Survival Guide. It looks different from the other types of Help in MS-DOS Editor. The Survival Guide contains information about editing keys, cursor movement, dialog boxes, selecting text, opening and saving files, and other basics.

Use this Help when you are learning how to use MS-DOS Editor or when you need help navigating through text.

You can use the Survival Guide by doing one of the following:

- From the Help menu, choose Getting Started or Keyboard.
- When in the MS-DOS Editor window, press F1.
- Immediately after starting MS-DOS Editor (and without specifying a file), press ENTER.

When the Help window appears, you can choose to view any of the related topics enclosed in the highlighted angle brackets. To view a related topic, move the cursor to it and press ENTER, or double-click the topic.

You can use the following keys to move the cursor from one topic to the next in the Survival Guide:

TAB	Moves the cursor to the next Help topic
SHIFT+TAB	Moves the cursor to the previous Help topic
A character	Moves the cursor to the next Help topic beginning with that character
SHIFT+a character	Moves the cursor to the previous Help topic beginning with that character
ALT+F1	Displays the Help window for a previously viewed Help topic (you can recall up to 20 Help topics)
CTRL+F1	Displays the next Help topic in the Help file
SHIFT+CTRL+F1	Displays the previous Help topic in the Help file

Specifying the Help Path

MS-DOS Editor uses a file named EDIT.HLP to display Help messages. MS-DOS looks for this file in the current directory and any directories specified by the **path** command. If the EDIT.HLP file isn't found, MS-DOS Editor cannot display Help.

You can specify the location of the EDIT.HLP file by using the Help Path command on the Options menu.

▶ To specify the Help path:

1. From the Options menu, choose Help Path.
   The Help Path dialog box appears.
2. Type the path of the directory that contains EDIT.HLP.
3. Choose the OK button.

Quitting MS-DOS Editor

You can quit MS-DOS Editor and save the file you are working on.

▶ To quit MS-DOS Editor:

- From the File menu, choose Exit.
  If you started MS-DOS Editor from the command line, the command prompt appears. If you started MS-DOS Editor from MS-DOS Shell, MS-DOS Shell appears.

If you are working with a new file or a file that has been changed but not saved, MS-DOS Editor prompts you to save it. For information about saving files, see "Saving a File" later in this chapter.

Creating a Text File

You create a text file by typing text in the MS-DOS Editor window. When you reach the end of a line, you must press ENTER to move the cursor to the next line. A line of text can be up to 256 characters long. You can include escape sequences in your text files by using techniques described in Chapter 11, "Customizing Your System."

The following is a list of keys you can use with MS-DOS Editor:

BACKSPACE or CTRL+H	Deletes the character to the left of the cursor.
DEL or CTRL+G	Deletes the character at the cursor.
CTRL+T	Deletes the word at the cursor (the cursor must be under the first letter).
INS or CTRL+V	Changes from insert to replace mode. The insert mode is the default. Press INS to replace characters instead of inserting them. Press INS again to resume inserting. In insert mode, the cursor is an underscore; in replace mode, the cursor is a box.

Moving the Cursor

You can use a mouse or the keyboard to move the cursor. You can also scroll through the text to see parts of a file that are not visible.

To move the cursor:

Mouse

- Click the area where you want the cursor to be.

Keyboard

- Press the arrow keys.

You can also use the following keys to move the cursor through text:

Arrow keys	Move the cursor one character or one line
CTRL+LEFT ARROW	Moves the cursor one word to the left
CTRL+RIGHT ARROW	Moves the cursor one word to the right
HOME	Moves the cursor to the beginning of the line
END	Moves the cursor to the end of the line
CTRL+ENTER	Moves the cursor to the beginning of the next line
CTRL+Q+E	Moves the cursor to the top of the window
CTRL+Q+X	Moves the cursor to the bottom of the window

If you are working with a file that is longer or wider than the MS-DOS Editor window, you can scroll through text to bring it into view.

▶ To scroll through text:

Mouse ■ Click the scroll arrows, drag the scroll box in the scroll bar, or click the shaded areas of the scroll bar.

Keyboard ■ Use the following keys to scroll through text:

CTRL+UP ARROW or CTRL+W	Scrolls up one line
CTRL+DOWN ARROW or CTRL+Z	Scrolls down one line
PAGE UP	Scrolls up one screen
PAGE DOWN	Scrolls down one screen
CTRL+HOME or CTRL+Q+R	Moves the cursor to the beginning of a file
CTRL+END or CTRL+Q+C	Moves the cursor to the end of a file
CTRL+PAGE UP	Scrolls left one screen
CTRL+PAGE DOWN	Scrolls right one screen

You can also scroll by holding down an arrow key.

Selecting Text

You begin most editing operations by *selecting* a block of text. In a single line, you can select any amount of text, from a single character to the entire line. You can also select several lines or the entire file at once. When you are selecting more than one line, entire lines will be selected. You cannot select all of line one and part of line two, for example. All of line one and two will be selected.

▶ To select text:

Mouse

1. Point to the first character you want to select.
2. Drag the cursor to the last character you want to select.
3. Release the mouse button.

   To cancel a selection, click anywhere in the window.

Keyboard

1. Use the arrow keys or other cursor-movement keys to move the cursor to the first character you want to select.
2. Press and hold down SHIFT, and use the arrow keys to move the cursor to the last character you want to select.
3. Release the keys.

   To cancel a selection, press any cursor-movement key.

Once text is selected, you can delete it by pressing DEL, or edit it by using the commands on the Edit menu.

Editing Text

The Edit and Search menus in MS-DOS Editor list the commands you use to edit text. Using the commands on the Edit menu, you can move or copy a block of text from one part of a file to another. You do this by transferring text from your file to a temporary storage area called a *buffer*.

Using the Search menu, you can search for a set of characters in a file. The characters found in a search can be replaced with a different set of characters.

Moving Text

You can move a block of text by using the Cut and Paste commands. This procedure is useful if you want to rearrange the order of text in a file.

▶ To move a block of text:

1. Select the block of text you want to move.
2. From the Edit menu, choose Cut, or press SHIFT+DEL.

   The block of text is deleted from the file and placed in the buffer.

   You can move all the text from the current line to the buffer by pressing CTRL+Y. You can select text from the cursor location to the end of the current line and move it to the buffer by first pressing CTRL+Q, and then pressing Y.
3. Move the cursor to the location in the file where you want the text to appear.
4. From the Edit menu, choose Paste, or press SHIFT+INS.

   The text in the buffer is inserted in the file at the cursor location.

When you choose the Paste command, the text is not removed from the buffer; it remains there until you move or copy another block of text to the buffer. You can insert text from the buffer into a file as many times as you want by repeating the Paste command.

Copying Text

You use the Copy and Paste commands to copy and reposition a block of text.

▶ To copy a block of text:

1. Select the block of text you want to copy.
2. From the Edit menu, choose Copy, or press CTRL+INS.

   The block of text is copied to the buffer. It is not removed from its original location.
3. Move the cursor to the location in the file where you want the block of text to appear.
4. From the Edit menu, choose Paste, or press SHIFT+INS.

   A copy of the text in the buffer is inserted into the file at the cursor location.

When you choose the Paste command, the text remains in the buffer until you move or copy another block of text to it. You can insert text from the buffer into a file as many times as you want by repeating the Paste command.

TIP If you want the text in the buffer to replace a block of text, select the block you want to be replaced, and choose Paste from the Edit menu.

Clearing Text

You can clear text from the MS-DOS Editor window without copying the text to the buffer. Choosing the Clear command or pressing DEL has no effect on text in the buffer.

▶ To clear text:

1. Select the text you want to clear.
2. From the Edit menu, choose Clear.

   Or press DEL.

Finding Text

To find text in a file, use the Find command on the Search menu. The text can be a word, a phrase, or any combination of characters and spaces. When MS-DOS Editor searches for text, it starts at the current cursor position and selects the first occurrence of the text.

▶ To find text:

1. From the Search menu, choose Find.

   The Find dialog box appears.

   If you select text before you choose the Find command, the selected text appears in the Find What box. Otherwise, the word at the current cursor location appears in this box. If you want to search for different text, type the text you are searching for in the Find What box.

2. If you want to match capitalization exactly, select the Match Upper/Lowercase option. Otherwise, MS-DOS Editor finds, for example, both *Brown* and *brown*.

3. If you want to find only separate occurrences of the specified text, select the Whole Word option. Otherwise, MS-DOS Editor finds, for example, *main* in *remainder*.

4. Choose the OK button to start the search.

   MS-DOS Editor begins searching at the cursor position. The first occurrence of the text is selected. If no occurrences of the text are found, the Match Not Found dialog box appears.

5. To search for the next occurrence of the specified text, choose the Repeat Last Find command from the Search menu, or press F3.

   The Find command continues to search through the file each time you choose the Repeat Last Find command. When it reaches the end of the file, it returns to the beginning and continues searching until it reaches its starting point.

Replacing Text

You can use the Change command to find text and replace it with new text. The Change command begins at the cursor location and continues to the end of the file. When it reaches the end of the file, it returns to the beginning and continues searching and replacing until it reaches its starting point.

▶ To replace text:

1. Position the cursor where you want to start replacing text.
2. From the Search menu, choose Change.

   The Change dialog box appears.
3. In the Find What box, type the text you want to replace. In the Change To box, type the text you want to replace it with.
4. If you want to match capitalization exactly, select the Match Upper/Lowercase option.
5. If you want to replace only separate occurrences of the specified text, select the Whole Word option.
6. Choose either the Find and Verify button or the Change All button to start the command.

   If you choose the Find and Verify button, each occurrence of the original text is selected, and the Change dialog box appears. The dialog box prompts you to make the change, skip the occurrence, cancel the command, or get Help.

   If you choose the Change All button, MS-DOS Editor replaces all occurrences of the specified text.
7. Choose the OK button (or press ESC) when the Change Complete dialog box appears.

Managing Files

You can use the commands on the File menu to open an existing file, work on a new file, save a file, or print a file.

Creating a File

You can create a file by using the New command on the File menu.

▶ To create a file:

1. From the File menu, choose New.

   If you have an open file that has unsaved changes, MS-DOS Editor displays a dialog box prompting you to save your changes. Choose the Yes button to save your changes; choose the No button to close the file without saving your changes.

MS-DOS Editor closes the current file and displays an empty window labeled "Untitled." You can begin typing text in this window.

2. If you want to save the new file, choose the Save or Save As command from the File menu. Then specify the drive and directory where you want to store the file, and give the file a name.

Opening a File

You can open several types of files by using the Open command on the File menu:

- Files created previously by using MS-DOS Editor
- Other unformatted text files (including such files as AUTOEXEC.BAT and CONFIG.SYS)
- Formatted text files created by using another text editor. However, special characters may lose their formatting functions if you open them with MS-DOS Editor.

▶ To open a file:

1. From the File menu, choose Open.

   The Open dialog box appears.

2. Type the name of the file you want to open, or select the filename from the file list.

   If the file you want is not on the current drive or directory, type the path as part of the filename.

   To view a list of the files on a different drive or directory, move to the Dirs/Drives list, select the drive or directory you want, and double-click it or press ENTER. Depending on your directory structure, you might need to move through several subdirectories to get to the directory you want. A list of files in the current drive and directory appears in the Files box.

3. When the name of the file you want to open is displayed in the File Name box, choose the OK button.

   If a file with unsaved changes is open, MS-DOS Editor displays a dialog box prompting you to save your changes before opening a new file. If you want to save your changes, choose the Yes button; if not, choose the No button. If you decide not to open the new file, choose the Cancel button.

Saving a File

After you create a file or make changes to an existing file, you can save it by using the Save or Save As command on the File menu. It is a good idea to save your work often in case there is a power loss or equipment failure.

CAUTION Some files that you open might include special formatting characters. If you save such a file when using MS-DOS Editor, the special characters lose their formatting function.

▶ To save a new file:

1. From the File menu, choose Save. The Save dialog box appears.
2. In the File Name Box, type a name for the file.

   If you want to save the file on a different drive or directory, select the drive or directory in the Dirs/Drives box, or include a path when you type the filename.
3. Choose the OK button.

▶ To save an existing file:

- From the File menu, choose Save.

▶ To save an existing file under a new name:

1. From the File menu, choose Save As.

   The Save As dialog box appears.
2. In the File Name Box, type a name for the file.

   If you want to save the file on a different drive or directory, select the drive or directory in the Dirs/Drives box, or include a path when you type the filename.

   If you attempt to save a file in a directory that contains a file with the same name, MS-DOS Editor displays a dialog box asking whether you want to replace the existing file. Choose the Yes button to replace the existing file; choose the No button if you want to give the file that you are saving a different name.
3. Choose the OK button.

TIP You can use the Save As command to save a modified version of a file without losing the original version. For example, if you have a file named MEMO.TXT, you can keep the original file and save a modified version as MEMO_2.TXT.

Printing a File

You can use the Print command on the File menu to print part or all of an open file. This command works only if you have a printer connected to or redirected through your LPT1 (parallel) printer port.

▶ To print a file:

1. Be sure the file you want to print is open. If you want to print a portion of a file, select the text you want to print.
2. From the File menu, choose Print.

   The Print dialog box appears.
3. If you want to print only the text you selected, choose the Selected Text Only option; otherwise, select the Complete Document option.
4. Choose the OK button.

Printing a Help Topic

You can use the Print command on the File menu to print Help information.

▶ To print a Help topic:

1. From the Help menu, choose Getting Started or Keyboard.
2. Choose the Help topic you want to print.
3. Be sure your cursor is in the Help window. If you want to print part of a Help topic, select the text you want to print.
4. From the File menu, choose Print.

   If you want to print only the text you selected, choose the Selected Text Only option; otherwise, select the Current Window option.
5. Choose the OK button.

Customizing MS-DOS Editor

You can use the commands on the Options menu to set the way the MS-DOS Editor window appears.

Controlling the Screen Display

You can use the Display command on the Options menu to change the colors in the MS-DOS Editor window, display or hide scroll bars, and set tab stops.

▶ To change the screen display:

1. From the Options menu, choose Display.

   The Display dialog box appears.

2. If you want to change your screen colors, select a foreground and a background color. The screen colors listed depend on the type of monitor you have. A sample of the selected screen colors appears to the left of the list boxes.

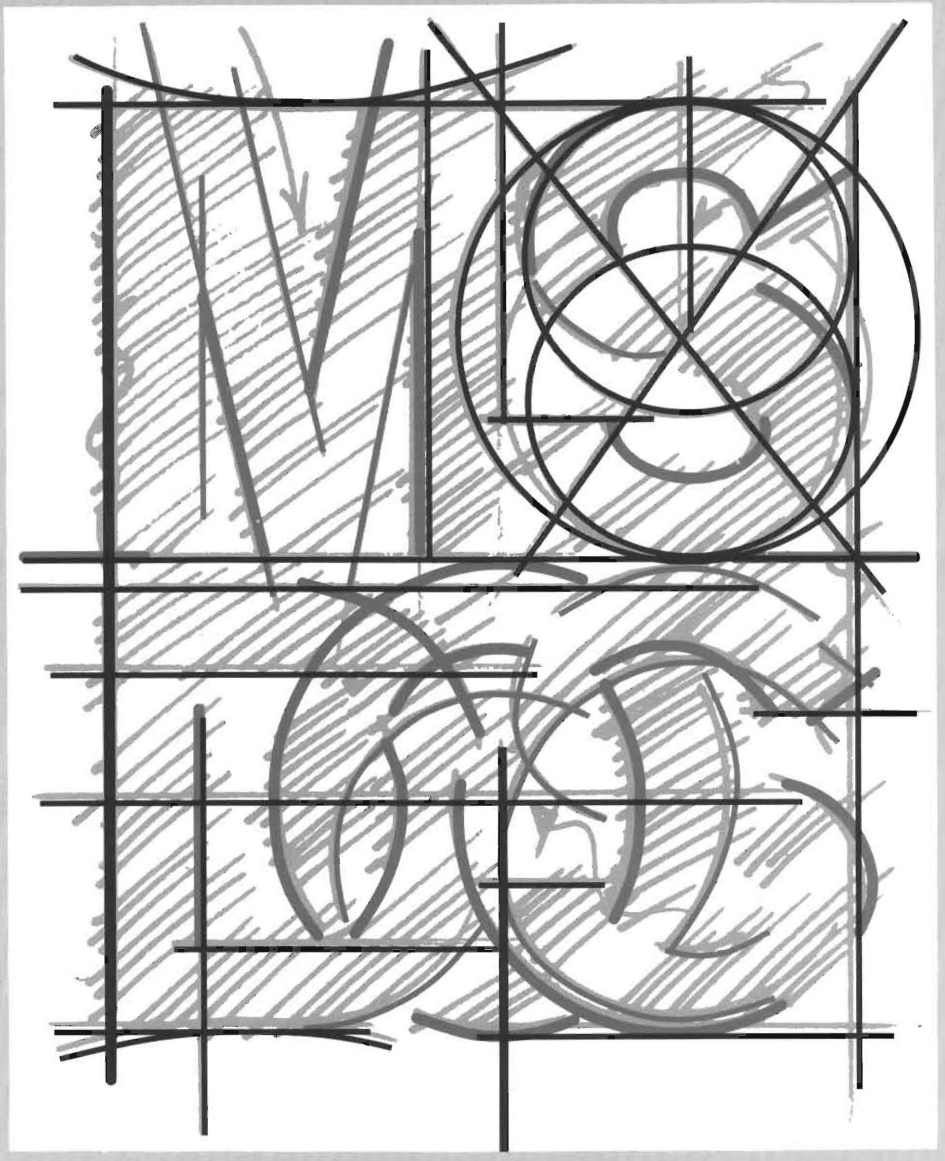
   If you want scroll bars to appear in the MS-DOS Editor window, select the Scroll Bars option. If this option is not selected, the scroll bars are hidden and you can see more of the window while you work. With the scroll bars hidden, you can still scroll through text by using the keyboard.

   If you want to set your tab stops, set the number of spaces between tab stops. The default is 8.

3. Choose the OK button.

You can also change the screen display by specifying one or more switches when you type **edit** at the command prompt. For information about these switches, see the **edit** command in Chapter 14, "Commands."

Part 3
Customizing MS-DOS



Chapter 10

Working with Batch Programs

10

As you work with MS-DOS, you'll find yourself repeatedly typing identical sequences of commands. For example, you might often type the same three commands to change the current drive, change the current directory, and then start a program. By using MS-DOS, you can store commands in a *batch program* or *batch file*. Instead of typing commands individually, you need only type the name of the batch program. MS-DOS carries out this "batch" of commands as if you had typed the commands individually from the keyboard.

Understanding Batch Programs

A batch program is an unformatted text file that contains one or more MS-DOS commands. For example, a batch program might contain the commands you use to change your directory and start a text editor.

Suppose you are backing up files to a floppy disk by using the following commands:

```
cd \work\docfiles
copy *.txt a:
cd \reports\xfiles
copy *.txt a:
```

To put these four commands into a batch program, you store them in an unformatted text file and assign the file a .BAT extension. Each time you want to back up your files, you type the name of the batch program at the command prompt.

Using batch programs has the following advantages:

- Batch programs speed up your work. When you run a batch program, you only have to remember one command, instead of several. You don't have to retype multiple commands or look up commands you can't remember.

- Batch programs customize MS-DOS. Using batch programs, you can create personalized commands that perform the exact task you need. You can also design your own prompts and messages.

Batch Commands

Any MS-DOS command you use at the command prompt or in MS-DOS Shell can also be put in a batch program. In addition, there are eight MS-DOS commands that are specially designed for batch programs. The commands and their functions are as follows:

call	Runs a second batch program, and then returns to the first one
echo	Displays messages on your screen, or turns the echo feature on or off
for	Carries out a command for a group of files or directories
goto	Switches to commands in another part of your batch program, and continues processing commands from that point
if	Carries out a command based on the result of a condition
pause	Temporarily stops your batch program from running; your program starts running again when you press any key
rem	Annotates your batch program so that you can remember what each part of the program does
shift	Changes the position of replaceable parameters

The **call**, **echo**, **goto**, **if**, **pause**, and **rem** commands are explained later in this chapter. For information about the **for** and **shift** commands, see Chapter 14, "Commands."

Tools for Creating a Batch Program

You can create a batch program by using MS-DOS Editor or the **copy** command. If you use a text editor other than MS-DOS Editor to create a batch program, save your files as unformatted (ASCII) text. Most popular text editors have an option for saving files this way. For information about MS-DOS Editor, see Chapter 9, "Working with MS-DOS Editor."

If you are creating a small batch program, it may be more convenient to use the **copy** command, which is described later in this chapter.

Naming a Batch Program

A batch program must have a .BAT filename extension. It is generally not a good idea to give a batch program the same name as an existing MS-DOS command. Suppose, for example, that you create a batch program for a customized formatting command and name it FORMAT.BAT. The program won't run if MS-DOS finds the FORMAT.COM file before it finds FORMAT.BAT, because MS-DOS gives precedence to files with .COM and .EXE extensions.

You can avoid this problem by using a name that is not already assigned to an MS-DOS command. For example, you could name the program MYFMT.BAT.

Running a Batch Program

To run a batch program, you type its name without the extension. For example, to run a batch program named FILES.BAT, type the following command:

```
files
```

If the batch program has parameters, add a space after the filename. For example, if the FILES.BAT program requires a file specification as a parameter, you would type a command like this:

```
files c:\reports\data
```

By default, MS-DOS displays each command in a batch program as the command is carried out. After the batch program runs, MS-DOS might display two command prompts because it treats the end-of-file character in a batch program as a new line.

Stopping a Batch Program

If you want to stop a batch program before all of its commands have run, press CTRL+C or CTRL+BREAK (more than once, if necessary). You get a message asking to confirm that you want to stop the batch program. Press **y** to stop the program or **n** to continue with the next command.

You can temporarily stop a batch program by pressing CTRL+S or the PAUSE key. This "freezes" the screen until you press another key.

If your batch program is on a floppy disk and you remove the disk while the program is running, MS-DOS displays the following message: "Not ready reading drive A. Abort, Retry, Fail?" To continue running the batch program, reinsert the disk and type **r**.

Testing a Batch Program

It is generally best to a create a large batch program in stages. This ensures that one part of the batch program works before you create another part.

When you run a batch program that contains an invalid command, MS-DOS cancels that command and proceeds to the next. If the batch program is set up to display commands as they are carried out, you'll see an error message when a command is invalid. If commands aren't displayed, the batch program contains an **echo off** command. Remove any **echo off** commands if you want commands to be displayed along with the error messages.

For more information about the **echo** command, see "Displaying Messages with a Batch Program" later in this chapter.

Creating a Small Batch Program

You can use the **copy** command to create a batch program directly from the keyboard. Suppose, for example, you want to create a batch program that formats a 360K disk in your high-density disk drive. To create the program and name it MYFMT.BAT, use the following **copy** command:

```
copy con c:\myfmt.bat
```

MS-DOS moves the cursor to the next line. At this point, the file is empty. To add the **format** command to the file, type the following:

```
format a: /f:360
```

Now you are ready to close the file and return to the command prompt. You do so by pressing CTRL+Z and then ENTER.

Once you have created the batch program, you need only type the name of the batch program to format a 360K floppy disk in your high-density disk drive, as follows:

```
myfmt
```

MS-DOS displays the **format** command on the screen, and then prompts you to insert a disk in drive A. (Make sure the directory that contains this batch program is either current or in the directory search path.)

The following batch program, BAKIT, is also small enough to create by using the **copy** command:

```
copy con c:\bakit.bat
```

The program backs up onto a floppy disk in drive A all .TXT and .BAK files from two directories on a hard disk, and then displays a list of the files on the floppy disk:

```
copy c:\work\may\*.txt a:
copy c:\reports\may\*.bak a:
dir a:
```

If there are more than a few lines in your batch program, it's a good idea to use a text editor to create the file. For information about the MS-DOS text editor, see Chapter 9, "Working with MS-DOS Editor."

Displaying Messages with a Batch Program

In Brief

If you want MS-DOS to display a message, use the **echo** command in your batch program, as follows:

```
echo Batchworks Backup Builder
```

To stop MS-DOS from displaying commands as it carries them out, use the following command:

```
echo off
```

You can include messages in a batch program to prompt you for additional information or to remind you of a particular task that the batch program does.

If you want MS-DOS to display a message on your screen, use the **echo** command. For example, you would use the following command to display the message "Put a disk in drive A":

```
echo Put a disk in drive A
```

TIP On networks, your message will be displayed more quickly if you put it in a .TXT file and then use the **type** command in your batch program to display the message.

MS-DOS displays this message on the screen. If you want the message shifted to the right a certain number of spaces, you must include the spaces as part of the message. For example, to center the message on your screen, add the necessary spaces in the command, as follows:

```
echo          Put a disk in drive A
```

If you want to skip a line, type **echo** followed by a period:

```
echo.
```

If **echo** is on, MS-DOS displays batch commands at the command prompt as it carries them out. Therefore, the message in the preceding example ("Put a disk in drive A") is displayed twice: first at the command prompt as part of the batch command, and then as a prompt to carry out the command itself. To suppress commands that appear at the command prompt and display a message only once, use the following command:

```
echo off
```

For example, you could add an **echo off** command at the beginning of the BAKIT.BAT program (described in the previous section):

```
echo off
copy c:\work\may\*.txt a:
copy c:\reports\may\*.bak a:
cls
echo Here are the files on the backup disk:
echo.
dir a:  /p
```

This batch program backs up files with a .TXT extension in one directory and files with a .BAK extension in another, then clears the screen and displays directory listings, one screen at a time. None of the commands listed after the first command (**echo off**) are displayed while the batch program is running. The line of text following the second **echo** command is displayed.

TIP To prevent a single command in your batch program from being displayed, put an at sign (@) in front of it. For example, to prevent the display of the **echo off** command, type **@echo off**.

If you want commands to be displayed, use this command at the beginning of the program:

```
echo on
```

Using the Pause Command

In Brief

To momentarily stop a batch program from running, use the **pause** command in the batch program, as follows:

```
pause
```

When MS-DOS finds a **pause** command in a batch program, it displays the following message:

```
Press any key to continue . . .
```

MS-DOS stops running the program until you press any key (except the PAUSE key).

For example, adding a **pause** command to the following BAKIT.BAT program stops the program from running while you put a disk in drive A:

```
echo off
echo Put a backup disk in drive A then
pause
copy c:\work\may\*.txt a:
copy c:\reports\may\*.bak a:
cls
echo Here are the files on the backup disk:
echo.
dir a:  /p
```

When this batch program pauses, MS-DOS displays the following:

```
Put a backup disk in drive A then
Press any key to continue . . .
```

Including Remarks in a Batch Program

In Brief

To include a remark that you do not want to appear on your screen, use the **rem** command in your batch program, as in the following example:

```
rem This part of the batch program copies files to a backup disk.
```

If your batch program is longer than a few lines, it is helpful to include remarks. You can use remarks to comment on the commands in a batch program and to make the program easier to read by separating it into sections.

You add a remark by typing **rem** followed by a space and the comment you want to include, as in this example:

```
rem This part of the batch program copies files to a backup disk.
```

After you type **rem** and a space, MS-DOS ignores any other text on the line. Therefore, you can type almost any characters you want on a remark line, or you can type **rem** and leave the rest of the line blank to add space to the file. However, do not use the less-than sign (<), the greater-than sign (>), or the pipe (|), since they have special meaning for COMMAND.COM.

For example, the following remarks divide and explain sections of BAKIT.BAT:

```
rem *****Backup the MAY subdirectories*****
rem
echo off
echo Put a backup disk in drive A then
pause
copy c:\work\may\*.txt a:
copy c:\reports\may\*.bak a:
rem
rem Clear the screen and display the files that were backed up
rem
cls
echo Here are the files on the backup disk:
echo.
dir a:  /p
```

Remarks don't affect the way a batch program runs; they simply annotate the commands for anyone who reads the file.

Running One Batch Program from Another

In Brief

To run a batch program from another batch program, include in your program the name of the second program with or without the **call** command. For example, if you want your batch program to start a program named SUBTASK.BAT, you could include the following command:

```
call subtask
```

If you don't need to return to the original batch program, you can type **subtask** without the **call** command.

You can run a batch program from another batch program by including just the name of the program you want to start or by including the **call** command with the name of the program. If you type only the name, the original batch program quits running, and the new batch program runs instead.

For example, the following batch program runs four commands and then starts a batch program named NEXTONE:

```
a:
cd \tmp
copy c:\*.sys a:
cd \perm
nextone
```

When NEXTONE finishes running, MS-DOS displays the command prompt.

If you want to return to the original batch program after running the other batch program, use a **call** command with the name of the program you want to start. When the second batch program finishes running, MS-DOS returns to the original batch program and carries out the next command.

For example, the following batch program carries out two commands, starts NEXTONE, and then carries out two more commands when NEXTONE finishes running:

```
a:
cd \tmp
call nextone
copy c:\*.sys a:
cd \perm
```

Using Replaceable Parameters

In Brief

Replaceable parameters (**%0** through **%9**) are placeholders for parameters typed at the command prompt. For example, suppose the BAKIT.BAT program includes the **copy** command with two replaceable parameters:

```
copy %1 %2
```

At the command prompt, you can type the two corresponding parameters, as in the following command:

```
bakit c:\comm\*.* a:
```

MS-DOS replaces **%1** with c:\comm*.* and **%2** with a:.

MS-DOS includes symbols called *replaceable parameters*, numbered **%0** through **%9**. You can include replaceable parameters in a batch program. When you run the batch program, MS-DOS replaces the symbol with the parameter you include when you type the batch command. The **%0** replaceable parameter substitutes for the name of the batch command as it is typed at the command prompt. Replaceable parameters **%1** through **%9** substitute for command-line parameters typed after the batch-command name. The first parameter on the command line is **%1**, the second is **%2**, and so on. If you want to specify more than nine parameters, use the **shift** command. For information about the **shift** command, see Chapter 14, "Commands."

Suppose you want to create a batch program that moves a file from one directory to another. In its simplest form, the batch program consists of a **copy** command and a **del** command. The **copy** command requires two parameters to specify the source and destination files; the **del** command requires one parameter to specify the file to be deleted. For example, the following batch program named MOVE.BAT copies the CASTILE.EXE file from the root directory of a disk in drive A to the GAMES directory of drive C, and then deletes the file on the disk in drive A:

```
copy a:castile.exe c:\games
del a:castile.exe
```

This batch program is very limited because it applies to only one file. You could use replaceable parameters in your MOVE.BAT program so that you could specify different files, as in the following example:

```
copy %1 %2
del %1
```

If you type the command **move a:castile.exe c:\games** at the command prompt, MS-DOS substitutes **a:castile.exe** for parameter **%1**, and **c:\games** for parameter **%2**.

If you use the percent sign (%) as part of a filename or string within a batch program, you must type it twice. The first occurrence indicates that the second **%** is part of a name, rather than a replaceable parameter.

In addition to replaceable parameters, you can use environment variables in a batch program. For information about environment variables and an example of how to use one in a batch program, see the **set** command in Chapter 14, "Commands."

Controlling Program Flow

To increase the flexibility of a batch program, you can use the **if** command to carry out different commands under different conditions, and the **goto** command to switch to different parts of the program. By using replaceable parameters with **if** and **goto** commands in a batch program, you can perform complex tasks.

Using the If Command

In Brief

If you want a batch command to be carried out only after certain conditions have been met, use the **if** command. For example, the following command starts Microsoft Word when W is used as a parameter:

```
if "%1"=="W" c:\word\word
```

The parameter and the text it is compared with must be enclosed in quotation marks, and they must match exactly. The double equal sign (==) means the parameter must equal the value. In this case, the W must be uppercase.

You can use the **if** command to specify a condition that must be true for a command to be carried out. For example, suppose you want to create a batch program named RUNIT.BAT that starts your chess program, Cmate, when you type the following command:

```
runit A
```

To perform this task, include the following **if** command in RUNIT.BAT:

```
if "%1"=="A" cmate
```

The double equal sign (==) means the parameter must equal the value. When MS-DOS carries out this command, it checks to see whether or not **%1** is an A. If **%1** is an A, MS-DOS carries out the command that follows (in this case, it starts the Cmate program). When you quit Cmate, MS-DOS carries out the command on the next line of RUNIT.BAT.

If **%1** is not an A, MS-DOS skips the command that runs Cmate and moves to the next line of the batch program. Both the parameter and the letter with which it is compared should be enclosed in quotation marks to avoid syntax errors when no parameter is present.

Using the Goto Command

In Brief

To switch to another part of a batch program, use the **goto** command and a label, as in the following command:

```
goto runword
```

The label *runword* (preceded by a colon) must appear on its own line elsewhere in the batch program, as follows:

```
:runword
```

The **goto** command directs your program to switch to another part of the program and continue processing the commands at that point. The line that the program is to switch to is marked with a label preceded by a colon (:). The same label appears in the **goto** command, as in the following example:

```
rem goto example
goto skipdown
echo both of these echo commands
echo will be skipped
:skipdown
cls
```

Using the If and Goto Commands Together

In Brief

To switch to a different line in a batch program after certain conditions have been met, use an **if** command with a **goto** command, as in the following example:

```
if "%1"=="W" goto runword
```

If you use the **goto** command with an **if** command, you can run different sections of a batch program under different conditions. For example, the following command directs MS-DOS to switch to the line labeled *chess* if you type an uppercase A:

```
if "%1"=="A" goto chess
```

Using a series of **if** commands, you can create a batch program that can run several programs. For example, the following batch program changes to the C:\GAMES\CHESS directory and runs the Cmate program if you type an uppercase A, and changes to the C:\GAMES\CHECK directory and runs the Checkers program if you type anything but an uppercase A:

```
if "%1"=="A" goto chess
rem
rem*****************************************************
rem If the user doesn't type A, run Checkers.
rem
cd \games\check
checkers
rem Skip over Chess by jumping to the line labeled :end.
goto end
rem
rem *****************************************************
rem If MS-DOS jumps to this label, the user wants Chess.
rem
:chess
cd \games\chess
cmate
rem The following line marks the end of the batch program.
:end
```

Creating a Menu System

One way to customize MS-DOS is to create a menu system that enables you to type simple commands to start batch programs you use regularly. Menus are particularly helpful if novice users are using your system.

For example, suppose you use Microsoft Word and a number of computer games. In addition, you use a batch program to back up your document files to a floppy disk. You can create a menu system that enables you or anyone else to use the programs without knowing where the programs are or how MS-DOS starts them.

The customized menu you create might look this:

```
Start Menu


Here's what you can do:

1. Back up your document files

2. Start Microsoft Word

3. Play a game

4. Use MS-DOS


Type the number you want and press ENTER:
```

You can easily create this Start menu, which includes a customized command prompt. Using the following **echo** and **prompt** commands in a batch program named MENU.BAT, you can direct MS-DOS to display the menu and prompt:

```
echo off
cls
echo      Start Menu
echo.
echo.
echo Here's what you can do:
echo.
echo 1. Back up your document files
echo.
echo 2. Start Microsoft Word
```

```
echo.
echo 3. Play a game
echo.
echo 4. Use MS-DOS
echo.
echo.
prompt Type the number you want and press ENTER:
```

The **cls** command clears your screen before MS-DOS displays this Start menu. The **prompt** command changes the command prompt so that you are asked to type a menu option. Notice, however, that this MENU.BAT program doesn't do anything but clear the screen and display a number of messages. The real work is done by other batch programs that perform the tasks on the menu. The menu itself simply indicates what you need to type to start the batch program that performs a particular task.

The batch program that performs each task is named according to the menu number used to select it:

Task	Batch program
Backs up your document files	1.BAT
Starts Microsoft Word	2.BAT
Starts a game	3.BAT
Returns to MS-DOS	4.BAT

Because the name of the batch program that performs a task is the same as the number of the option in the list, when you type a number at the menu prompt you are starting a new batch program.

Menu Item 1: Backing Up Files

In the menu system just discussed, menu option 1 is "Back up your document files." The batch program that performs this task is called 1.BAT so that it will start when you type **1** at the menu prompt.

This batch program copies .DOC files from their usual directory to a floppy disk. The batch program includes a **pause** command that causes the program to pause while you put a backup disk in drive A. When the 1.BAT program finishes running, it starts the MENU.BAT program so that you can return to the Start menu to perform another task.

If your document files are in the C:\WORD\DOCFILES directory, and the backup drive is A, the 1.BAT program would contain the following commands:

```
echo off
cls
echo To back up your files, put a backup disk in drive A.
pause
echo These files are being backed up:
copy c:\word\docfiles\*.doc a:
echo.
echo When you are ready to return to the Start menu
pause
menu
```

When you choose 1 from the Start menu, this batch program runs. The batch program clears the screen and prompts you to insert the backup disk. After the files are copied, the batch program pauses so you can read the display. As its last command, the batch program runs the MENU.BAT program to return you to the Start menu.

Menu Item 2: Starting a Word Processor

In the menu system described earlier in this section, menu option 2 is "Start Microsoft Word" and the corresponding batch program is named 2.BAT. If the Microsoft Word program file is C:\WORD\WORD.EXE and is defined in your path, the following command would start the program:

```
c:\word\word
```

However, this option would be more useful if it displayed the available Microsoft Word document files and then offered you the choice of editing an existing file or opening a new file before starting the program. These tasks require two batch programs. The first program displays existing files and prompts you for a file to edit; the second starts Microsoft Word with an existing file or a new file. The first batch program (2.BAT) is like another menu item. It displays a list of files to choose from and then prompts you to choose one.

The 2.BAT batch program would contain the following commands, assuming that your files have a .DOC extension and are located in the C:\WORD\DOCFILES directory, and that the Microsoft Word program file is C:\WORD\WORD.EXE:

```
echo off
cls
cd c:\word\docfiles
echo          Start Microsoft Word
echo.
echo.
echo Here are the files that already exist:
echo.
```

```
dir *.doc /p /b
echo.
echo To create a new file, type W and press ENTER.
echo.
echo To open an existing file, type W and the filename
echo (separated by a space) and press ENTER.
echo.
prompt What will it be?:
```

The 2.BAT batch program makes the document directory current and then lists the available files. It sets the current directory to C:\WORD\DOCFILES so you don't have to type any directory paths. The **dir** command includes the **/p** switch so that MS-DOS pauses if there is more than one screen of files, and the **/b** switch, which displays filenames only. The batch program displays two options for selecting document files. The final command in the batch program changes the prompt. The 2.BAT program produces a screen that might look like the following:

```
            Start Microsoft Word


Here are the files that already exist:

CH1 DOC
CH2 DOC
CH3 DOC
CH4 DOC

To create a new file, type W and press ENTER.

To open an existing file, type W and the filename
(separated by a space) and press ENTER.

What will it be?:
```

To start Microsoft Word with an existing or new file, you need another batch program. With most text-editing programs, including Microsoft Word, you can specify a file to edit when you start it. For example, to start Microsoft Word and edit the ACCT.DOC file in the C:\WORD\DOCFILES directory, you would type this command:

```
c:\word\word c:\word\docfiles\acct.doc
```

If you don't type a filename after the name of the program file, it is assumed that you want to create a new file.

You can also use a replaceable parameter to start Microsoft Word and indicate which file to open (if any). According to the instructions that 2.BAT displays, W creates a new file, and W plus a filename opens an existing file.

You can offer both options by creating a program named W.BAT that includes the following command:

```
c:\word\word %1
```

When you type **w**, MS-DOS starts the W.BAT program. Make sure that W.BAT is in the current directory (C:\WORD\DOCFILES) or in the directory search path so that you don't need to type a path.

If you include a filename when you type **w**, MS-DOS assigns it to the **%1** parameter, and Microsoft Word opens that file. If you don't type a filename, MS-DOS ignores **%1**, and Microsoft Word creates a new file.

The W.BAT program needs two more commands to return you to the Start menu. When you quit Microsoft Word, MS-DOS returns to the batch program and carries out the next command. To return you to the menu, the batch program should change the current directory back to the root directory of drive C. The following shows the command that starts Microsoft Word and the two commands that return you to the Start menu. Together, these commands make up the W.BAT batch program:

```
c:\word\word %1
cd \
menu
```

Menu Item 3: Choosing a Game

In the menu system described earlier in this section, menu option 3 is "Play a game." To create a batch program for this task, you can use **if** and **goto** commands. (See the section "Using the If and Goto Commands Together" earlier in this chapter.)

There are two batch programs for this option: one that displays a games menu, and one that starts the selected game. The first batch program must be named 3.BAT so that you can start it from the Start menu by typing **3**. If there were three games that you wanted to list—chess, checkers, and backgammon—you would include the following commands in 3.BAT:

```
echo off
cls
echo        Let's play a game!
echo.
echo.
echo Here are your choices:
echo.
echo A. Play chess.
echo B. Play checkers.
echo C. Play backgammon.
```

```
echo.
echo To start a game, type "game" and the letter of the
echo game and then press ENTER (for example, game A).
echo.
prompt What will it be?:
```

Like the batch program that displays the Start menu, this batch program simply presents another set of choices. You could create a separate batch program for each option. However, by using **if** and **goto** commands, you can include all the options in a single batch program.

The instructions in the menu prompt you to type **game** and the letter of the game you want to play. Suppose the chess game is \GAMES\CHESS\CMATE.EXE, the checkers game is \GAMES\CHECK\CHECKERS.EXE, and the backgammon game is \GAMES\BACK\BACKGAMM.EXE. A batch program named GAME.BAT containing the following commands can perform all the tasks listed on the menu:

```
rem ********** This batch program runs each game *************
echo off
cls
rem
rem Available games.
rem
if "%1"=="A" goto chess
if "%1"=="B" goto check
if "%1"=="C" goto back
rem
rem ******************************************************
rem Choose a game.
rem
echo.
echo Enter game A, game B, or game C.
pause
rem
rem If no game is selected, skip over all game commands.
rem
goto end
:chess
rem
rem ******************************************************
rem If the user chose chess, jump here.
rem
cd \games\chess
cmate
rem
rem If the user chose chess, skip checkers and backgammon.
rem
goto end
:check
rem
```

```
rem ******************************************************
rem If the user chose checkers, jump here.
rem
cd \games\check
checkers
rem
rem If the user chose checkers, skip backgammon.
rem
goto end
:back
rem
rem ******************************************************
rem If the user chose backgammon, jump here.
rem
cd \games\back
backgamm
rem
rem ******************************************************
rem When the games are finished, reset the current directory
rem to the root directory and display the Start Menu.
rem
:end
cd \
menu
```

MS-DOS switches to one of three locations in the batch program, depending on the parameter you type. If you omit a parameter or type an invalid parameter, an error message is displayed and you return to the Start menu. The batch program GAME.BAT must be in your root directory or in the directory search path.

The batch program changes the current directory to the directory that contains the selected game. However, before the batch program finishes running, it resets the current directory to your root directory. The last command in this batch program starts the batch program that displays the Start menu.

Menu Item 4: Quitting the Menu System

In the menu system described earlier in this section, menu option 4 is "Use MS-DOS." The batch program that returns you to the command prompt is named 4.BAT. It contains the following commands:

```
prompt $p$g
cls
```

Once again, the Start menu prompts you to type a number. If you type **4**, MS-DOS runs the 4.BAT batch program. This batch program sets the command prompt to show the current drive and directory followed by a greater-than sign (>), and then clears your screen.

Chapter 11

Customizing Your System

11

You can customize the way MS-DOS uses hardware, memory, and files. For example, you can customize the way MS-DOS interacts with your keyboard and monitor. You can also change the amount of memory that MS-DOS reserves for itself and for storing files.

Hardware that you use to communicate with your computer is called a *device*. Devices such as a keyboard and mouse provide your computer with necessary information (*input*). Other devices, such as a monitor or printer, receive information from your computer (*output*). Each device has its own characteristics that can be customized.

For every device, there is a program that MS-DOS uses to control it. Such a program is called a *device driver*. For example, MS-DOS uses a device driver to control how information is read to and from a floppy disk drive. MS-DOS has built-in device drivers for your keyboard, monitor, hard and floppy disk drives, and ports. MS-DOS includes other device drivers that are *installable*. Installable device drivers are stored on disk. When you want to use an installable device driver, MS-DOS installs it by transferring it from disk to memory.

When MS-DOS starts, it looks for the CONFIG.SYS file, which specifies which devices to install and which installable device drivers to use. This file also contains commands that determine how MS-DOS uses memory and controls files. For information about MS-DOS and memory, see Chapter 12, "Optimizing Your System."

After carrying out the commands in your CONFIG.SYS file, MS-DOS looks for the AUTOEXEC.BAT file, a batch program that defines the characteristics of each device connected to your system. The AUTOEXEC.BAT file can also include any MS-DOS commands you want to carry out when you start your system. For example, you can define the port to which your printer is connected, control the rate that MS-DOS repeats a keystroke when you press and hold down a key, define the path that MS-DOS uses to find files, and clear your screen of startup messages.

Creating a Startup Procedure

A *startup procedure* is one or more commands that MS-DOS carries out each time you start your system. These commands set the characteristics of your devices, customize information that MS-DOS displays, and start batch programs and other programs.

MS-DOS has a startup procedure that is defined in your AUTOEXEC.BAT file. This file is located in the root directory of your startup disk. Each time you start your system, MS-DOS carries out the commands stored in the AUTOEXEC.BAT file. You can run AUTOEXEC.BAT without restarting your system by typing **autoexec** at the command prompt.

You can create your own startup procedure in your AUTOEXEC.BAT file by using the techniques described in Chapter 10, "Working with Batch Programs." Be sure to store your startup procedure in a file named AUTOEXEC.BAT so that MS-DOS runs your procedure automatically. To avoid losing your original MS-DOS AUTOEXEC.BAT startup file, make a backup copy and save it with a different name.

Startup Commands

Every command in an AUTOEXEC.BAT file can also be used in other batch programs. The following are the most common AUTOEXEC.BAT commands:

- The **mode** command sets the characteristics of your keyboard, monitor, and ports. For more information about the **mode** command, see "Configuring Your Ports" later in this chapter.
- The **date** and **time** commands prompt you for the correct date and time. These commands are important to include in your AUTOEXEC.BAT file only if your system does not have a clock that stays current when the power is off. For more information about the **date** and **time** commands, see the next section, "Sample Startup Procedures."
- The **path** command indicates the directories MS-DOS searches for a program file that you want to run. For more information about the **path** command, see the next section, "Sample Startup Procedures."

- The **echo off** command directs MS-DOS not to display the commands in your AUTOEXEC.BAT file as they start. For more information about the **echo** command, see the next section, "Sample Startup Procedures."

- The **set** command creates an environment variable that can be used by programs. For more information about the **set** command, see the next section, "Sample Startup Procedures."

Your AUTOEXEC.BAT file often contains commands that run batch programs or other programs without your having to type their names at the command prompt. For example, to run MS-DOS Shell each time you start your system, you would include the following command at the end of your AUTOEXEC.BAT file:

```
dosshell
```

After MS-DOS finishes running all the programs in your AUTOEXEC.BAT file, it displays the command prompt (or MS-DOS Shell, if your AUTOEXEC.BAT file is set up to start MS-DOS Shell).

You might want to include in your AUTOEXEC.BAT file the command that installs Doskey and any Doskey macros you commonly use. For more information about the Doskey program, see Chapter 7, "Advanced Command Techniques" or the **doskey** command in Chapter 14, "Commands."

Sample Startup Procedures

You may want to create your own startup procedure in the AUTOEXEC.BAT file. For example, suppose your system has two floppy disk drives and a clock that does not run when the power to your computer is off. You could add the following commands in your AUTOEXEC.BAT file:

```
date
time
path a:
```

The **date** and **time** commands prompt you to set the date and time when you start MS-DOS. Commands such as **xcopy**, **backup**, and **restore** may not work correctly if your clock is not accurate. In this example, the **path** command specifies that MS-DOS should look for commands or programs in the root directory of the disk in drive A, in addition to the current directory.

Suppose your system has one floppy disk drive, one hard drive, a clock that does not need to be set, and MS-DOS Shell installed. It may be useful to include the **path**, **prompt**, **doskey**, and **dosshell** commands in your AUTOEXEC.BAT file, as in the following example:

```
path c:\;c:\dos;c:\utility;c:\batch
prompt $p$g
doskey
dosshell
```

The **path** command in this example directs MS-DOS to look for program files in the current directory and then in the following directories: the root directory of drive C, C:\DOS, C:\UTILITY, and C:\BATCH. A semicolon (;) separates each directory.

In the example just shown, the **prompt** command displays the current drive and directory, followed by a greater-than sign (>), as the command prompt. This is one of the more common forms of the command prompt.

The **doskey** command loads the Doskey program into memory. Because the **doskey** command follows the **path** command, DOSKEY.COM can be located in any of the directories listed in the **path** command. The **dosshell** command starts MS-DOS Shell.

Suppose your system has one floppy disk drive, one hard disk drive, a clock that does not need to be set, a laser printer connected to port COM1, and a menu batch program. It may be useful to have the following commands in your AUTOEXEC.BAT file:

```
echo off
path c:\;c:\dos;c:\utility;c:\batch;c:\word;c:\excel
prompt $p$g
mode lpt1=com1
mode com1:96,n,8,1,p
set temp=c:\temp
menu
```

When the **echo off** command is included, the AUTOEXEC.BAT commands are not displayed as they are carried out. The first **mode** command redirects printer output from LPT1 (its default port) to the serial port COM1. The second **mode** command sets up the COM1 port for use with a laser printer. The **set** command creates an environment variable named TEMP. Many programs use this variable when storing temporary files.

The last command in this AUTOEXEC.BAT file starts another batch file named MENU.BAT. When this and any other programs that are started from the AUTOEXEC.BAT file finish running, MS-DOS displays a command prompt (or MS-DOS Shell, if your AUTOEXEC.BAT file is set up to start MS-DOS Shell).

Configuring MS-DOS for Your System

Before looking for your AUTOEXEC.BAT file, MS-DOS carries out a group of commands that load installable device drivers and reserve space in system memory for information processing. The file that contains these commands is called CONFIG.SYS. Like the AUTOEXEC.BAT file, a version of CONFIG.SYS is created by the MS-DOS Setup program and is in the root directory of your startup disk.

Modifying Your CONFIG.SYS File

You can add and change CONFIG.SYS commands to configure your system as needed. Because the CONFIG.SYS file controls how MS-DOS starts, MS-DOS reads it only when you start your system. If you change this file, you must restart your system in order to have the changes take effect.

To edit your CONFIG.SYS file, use MS-DOS Editor or a text editor that saves files as unformatted (ASCII) text. For more information about using MS-DOS Editor, see Chapter 9, "Working with MS-DOS Editor."

Before you change the CONFIG.SYS file, you should make a backup copy of the file and save it under a different name.

If you change your CONFIG.SYS file, and you later cannot start your system, follow these steps:

1. Insert the system disk in drive A, and start your system.
2. Copy your backup copy of the original CONFIG.SYS file to the root directory of your startup drive; name the file CONFIG.SYS.
3. Remove the system disk from drive A, and restart your computer.

Understanding Configuration Commands

There are 15 commands you can include in the CONFIG.SYS file. Except for the **break** and **rem** commands, you can't type these commands at the command prompt; they must be included in the CONFIG.SYS file. The following list briefly describes the purpose of each CONFIG.SYS command:

break	Specifies when MS-DOS is to check for the CTRL+C or CTRL+BREAK key combination
buffers	Sets the amount of RAM that MS-DOS reserves for information transferred to and from a disk
country	Sets the country language conventions for your system
device	Loads an installable device driver into the operating system
devicehigh	Loads device drivers into the upper memory area
dos	Sets the area of RAM where MS-DOS will be located, and specifies whether to use the upper memory area
drivparm	Sets characteristics of a disk drive
fcbs	Sets the number of file control blocks (FCBs) that MS-DOS can open concurrently
files	Sets the number of files that MS-DOS allows to be open at one time
install	Runs a terminate-and-stay-resident (TSR) program while MS-DOS reads the CONFIG.SYS file
lastdrive	Sets the number of valid drive letters
rem	Indicates the use of descriptive comments in your CONFIG.SYS file
shell	Indicates that a command interpreter other than COMMAND.COM should be used, or that COMMAND.COM should be set up differently
stacks	Sets the amount of RAM that MS-DOS reserves for processing hardware interrupts
switches	Specifies the use of conventional keyboard functions even though an enhanced keyboard is installed

The **drivparm**, **fcbs**, **install**, **shell**, **stacks**, and **switches** commands are described in Chapter 14, "Commands." The **country** command is described in Chapter 13, "Customizing for International Use." For information about using the **dos** and **devicehigh** commands, see Chapter 12, "Optimizing Your System." The **rem** command is described in Chapter 10, "Working with Batch Programs."

The **break**, **buffers**, **device**, **files**, and **lastdrive** commands are described in the following sections in this chapter.

Installing Device Drivers

In Brief

To install a device driver, include a **device** command in your CONFIG.SYS file, as in the following example:

```
device=c:\dos\ansi.sys
```

To use an installable device driver, you include a **device** command in your CONFIG.SYS file. This loads, or *installs*, the driver into memory. For example, to use the device driver MOUSE.SYS in the C:\MOUSE directory, you would include the following command in your CONFIG.SYS file:

```
device=c:\mouse\mouse.sys
```

When MS-DOS reads this command, it loads MOUSE.SYS into memory. The driver becomes part of the MS-DOS system software and remains in memory.

In many cases, when you install a program that works with an installable device driver, it adds the appropriate command to your CONFIG.SYS file. The **device** command is described in Chapter 14, "Commands."

Increasing Memory for File Transfer

In Brief

To increase the memory reserved for file transfer, include the **buffers** command in your CONFIG.SYS file, as in the following example:

```
buffers=30
```

When MS-DOS starts, an area in your main memory is reserved for temporarily holding information from disks. The memory is divided into units called *buffers*, each of which is the same size as a sector on a disk (usually .5K). Each buffer can hold a sector of information from a disk. Buffers hold parts of files that are waiting to be stored or used by a program, in addition to directory and file-table information about the disk.

As your directory structure becomes more complex, MS-DOS works more efficiently with additional buffers. However, the more buffers there are, the less space there is for other programs and data in memory.

To set the number of buffers reserved for file transfers, include the **buffers** command in your CONFIG.SYS file. The amount of space reserved by the **buffers** command depends on the size of the disk sectors. For example, you would use the following command to reserve 20 buffers for file-transfer operations (10K of buffer space if each sector equals .5K):

```
buffers=20
```

NOTE Disk-caching programs, such as SMARTDRV.SYS, perform much of the work of buffers. If you use a disk-caching program, you need fewer buffers. For information about SMARTDRV.SYS, see Chapter 12, "Optimizing Your System."

Increasing the Number of Open Files

In Brief

To increase the number of files that can be open simultaneously, include the **files** command in your CONFIG.SYS file, as in the following example:

```
files=30
```

When MS-DOS starts, space in memory is reserved for a table that contains information about the files currently open. The more files you expect to have open at one time, the more space needed for this table. You can have up to 255 files open at one time. To specify the maximum number of files you expect to have open, use the **files** command.

For example, to reserve enough space for 30 files, include the following command in your CONFIG.SYS file:

```
files=30
```

If you run database or spreadsheet programs, or if you run MS-DOS with Microsoft Windows™ graphical environment or with network software, you are likely to need 30 open files. However, the larger the number you specify, the more space taken up in memory and the less space available for programs and data.

If you do not include a **files** command in your CONFIG.SYS file, space is reserved for eight open files. If you specify more than 255 files, an error message appears when MS-DOS runs the CONFIG.SYS file.

Increasing CTRL+C Checking

In Brief

To increase the number of times MS-DOS checks for the CTRL+C or CTRL+BREAK key combination (which stops a command), include the **break** command in your CONFIG.SYS file, as in the following example:

```
break=on
```

Unless you specify otherwise, MS-DOS checks whether CTRL+C or CTRL+BREAK has been pressed only while it reads from the keyboard or writes to the screen or printer. For example, if you press CTRL+C or CTRL+BREAK while MS-DOS is saving a file to a disk, MS-DOS does not carry out CTRL+C or CTRL+BREAK until the next time it displays something on your screen.

Include the following **break** command in your CONFIG.SYS file if you want these key combinations checked more frequently:

```
break=on
```

NOTE Some programs ignore or redefine CTRL+C and CTRL+BREAK. Pressing these keys while you are using such a program may have another effect, or no effect at all.

The **break** command can also be used in a batch program or at the command prompt. For information about using CTRL+C or CTRL+BREAK, see Chapter 2, "Command-Line Basics."

Increasing the Number of Logical Drives

In Brief

To increase the number of logical drives that your system recognizes, include the **lastdrive** command in your CONFIG.SYS file, as in the following example:

```
lastdrive=z
```

The default value is one more drive than the number of drives you actually have.

When MS-DOS starts, it reserves space in memory for a table that contains information about each logical drive that you expect your system to use. Regardless of how many physical disk drives your system has, you can reserve space for up to 26 logical drives. You might need to increase the number of logical drives if your system is part of a network, or if you use the **subst** command to equate a

directory with a disk drive. For more information about physical and logical drives, see "Adding Disk Drives" later in this chapter.

To change the number of logical drives, include a **lastdrive** command in your CONFIG.SYS file. For example, the following command reserves space for 10 logical drives (A through J):

```
lastdrive=j
```

The **lastdrive** command prepares MS-DOS to recognize extra logical drives. However, you must still assign these drives (to physical devices, network shares, existing directories, and so on) before you can use them. You cannot reserve space for fewer drives than you actually have. The default value is one more drive than the number of drives you actually have.

Sample Configuration Files

If you have a mouse and you use spreadsheet or database programs, you might have the following commands in your CONFIG.SYS file:

```
buffers=20
files=30
device=c:\dos\mouse.sys
break=on
```

The **buffers** command reserves 20 buffers (10K of buffer space if each sector equals .5K) for transferring information to and from disks. The **files** command reserves enough room to have 30 files open at one time. The **device** command loads the device driver MOUSE.SYS from the C:\DOS directory. The **break** command checks frequently for the CTRL+C or CTRL+BREAK key combination.

If you use a network, and your system includes an 80386 or higher processor, expanded memory, and 1MB of extended memory, your CONFIG.SYS file might look like this:

```
buffers=20
files=30
rem   The following commands install the mouse, network, and memory
rem   drivers, as well as the Smartdrive disk-caching program. The
rem   /a switch tells Smartdrive to use expanded memory.
device=c:\mouse.sys
device=c:\net\network.sys
device=c:\bin\himem.sys
device=c:\bin\emm386.exe
device=c:\bin\smartdrv.sys /a
break=on
rem The following command reserves space for 26 drives.
lastdrive=z
```

In addition to the four commands shown in the previous example, this CONFIG.SYS file has commands to load additional device drivers and reserve space for additional drives.

A **rem** command indicates a comment in a CONFIG.SYS file. MS-DOS does not carry out **rem** commands.

The **device** commands loads installable device drivers. In this sample file, there are drivers for communicating with the mouse, managing the computer's network link, managing extended memory, emulating expanded memory in extended memory, and managing disk caches. For more information about installable drivers, see Chapter 12, "Optimizing Your System."

The **lastdrive** command reserves space for 26 logical drives. In other words, letters from A to Z are available as labels for drives.

Configuring Your Ports

To configure your parallel and serial ports, you use the **mode** command. You can use the **mode** command alone or with a device name and one or more parameters or switches.

Other programs that you run may change the configuration of certain devices. For example, most word-processing programs include printer drivers. These drivers may supersede any printer driver you configure by using the **mode** command.

Configuring Your Printer

In Brief

To connect your printer to a serial port, use the **mode** command, as in the following example:

```
mode lpt1 = com1
```

MS-DOS redirects to COM1 the output it typically sends to LPT1.

By default, your printer is attached to the first parallel port (LPT1). To connect your printer to a different port, use the **mode** command with the names of the two ports. For example, use the following command to redirect printer output from LPT1 to COM1:

```
mode lpt1 = com1
```

You can only redirect a parallel port to a serial port. You cannot redirect a parallel port to another parallel port, and you cannot redirect a serial port to any other port.

For printers that support Epson-compatible escape sequences, the default is 80 characters across a line and 6 lines per inch of paper. If your printer prints 132 characters per line or 8 lines per inch, use the **mode** command to set MS-DOS to the new values. For example, the following command configures MS-DOS for a printer connected to port LPT1 with 132 characters per line and 8 lines per inch:

```
mode lpt1:132,8
```

You can add a *retry setting* to indicate how a print job should be handled when a printer does not accept information. For example, use the following command to continue sending a print job to LPT1 until the printer accepts the job:

```
mode lpt1:,,p
```

The two commas are placeholders for the first two parameters (characters-per-line and lines-per-inch), which will not change. MS-DOS continues to send information to the printer until the printer returns a completion code or an error code to MS-DOS, or until you press CTRL+C or CTRL+BREAK to stop printing.

If you use the **mode** command with no parameters, you receive information about your LPT, COM, and CON ports, such as whether any of your LPT ports have been redirected. For LPT1, the retry setting is included.

You can receive information about a specific port by using the **mode** command, the name of the port, and the **/status** switch. For example, use the following command to receive information about LPT1:

```
mode lpt1 /status
```

For more information about the **mode** command, see Chapter 14, "Commands."

Configuring a Serial Port

In Brief

To change the way MS-DOS communicates with a serial port, use the **mode** command with the name of the serial port, as in the following example:

```
mode com1:96,n,8,1,p
```

This command configures a serial port for use with a 9600-baud modem. The *n* indicates that there is no parity checking. The *8* specifies that there are 8 data bits per character. The *1* specifies one stop bit. The *p*, a retry setting, directs MS-DOS to continue sending the print job until the printer accepts the job.

To set up a serial port, you use the **mode** command with the name of the port. To specify how the port should be set up, use one or more of the following parameters:

baud=*b* Sets the transmission rate at which MS-DOS communicates with the port. (The default value, which depends on your system, is usually 1200 baud.) You can abbreviate this parameter by omitting **baud**= and specifying a value for *b*.

parity=*p* Sets the error-checking mode of the port (the default is even parity). You can abbreviate this parameter by omitting **parity**= and specifying a value for *p*.

data=*d* Indicates how many data bits the port expects (the default is 7 data bits). You can abbreviate this parameter by omitting **data**= and specifying a value for *d*.

stop=*s* Indicates how many stop bits the port expects (the default is 2 stop bits if baud=110; otherwise the default is 1 stop bit). You can abbreviate this parameter by omitting **stop**= and specifying a value for *s*.

retry=*r* Indicates how a device that isn't ready to receive data should be handled (the default is to stop sending data if the device is not ready). You can abbreviate this parameter by omitting **retry**= and specifying a value for *r*.

To change these settings for any of your serial communications ports, you can use the **mode** command with the name of the port. For example, to set your COM2 port to work with a 2400-baud modem with even parity, seven data bits, and one stop bit, you would use the following command:

```
mode com2:  baud=24
```

You do not have to specify the parity, data, and stop bits, because they are the same as the default values.

You can type the name of the parameter you want to change, or you can list the values you want in order, using commas between each value. For example, the following two commands are equivalent:

```
mode com1: baud=96 parity=n data=8 stop=1 retry=p
mode com1:96,n,8,1,p
```

For a complete list of values you can use for each parameter, see the **mode** command in Chapter 14, "Commands."

Adding Disk Drives

MS-DOS has a built-in device driver that controls all standard floppy disk drives reported by your system's ROM BIOS. (ROM BIOS stands for read-only memory Basic input/output system.) If you add a floppy disk drive and it does not work, then the ROM BIOS is not reporting that drive. In that case, you must install a device driver called DRIVER.SYS. You can use this device driver to control up to four additional floppy disk drives. DRIVER.SYS cannot be used to control hard disk drives. For more information about installable device drivers, see Chapter 15, "Device Drivers."

DRIVER.SYS involves both *physical drives* and *logical drives*. Physical drives are hardware components that are numbered, beginning with 0. Your first floppy disk drive is always physical drive 0, the second is always physical drive 1, and so on. If you have a hard disk drive, it is always physical drive 128 whether or not you have a second floppy disk drive. MS-DOS supports 128 physical floppy drives.

Logical drives are labels that MS-DOS uses to keep track of where it sends data. Logical-drive labels are letters A to Z. Every physical drive has a corresponding logical drive. However, every logical drive does not have a corresponding physical drive. Your first floppy disk drive (physical drive 0) is always represented as drive A; the second (physical drive 1) is always represented as B. You can have logical drives that do not correspond to physical drives if you use network drives, extended MS-DOS partitions, or RAM drives, or if you substitute a drive letter for a directory.

In general, you use only logical-drive letters, not physical-drive numbers. MS-DOS supports only 26 drive letters at a time. The only time you need to know the physical-drive number of a device is when you define the drive by using DRIVER.SYS or when you redefine the drive by using a **drivparm** command in your CONFIG.SYS file. For information about the **drivparm** command, see Chapter 14, "Commands."

Installing the Driver

In Brief

To install DRIVER.SYS, include a **device** command in your CONFIG.SYS file. For example, if the DRIVER.SYS file is in the C:\DOS directory, the following command installs DRIVER.SYS for a new 1.2-MB, 5.25-inch floppy disk drive:

```
device=c:\dos\driver.sys /d:2 /c /f:1
```

When you define a new disk drive, you indicate which physical drive it is. MS-DOS assigns it the next available logical drive letter. You cannot use DRIVER.SYS to change a previously defined logical drive. However, you can assign a second logical-drive letter to a drive to change its characteristics.

If you add a floppy disk drive that is not supported by your system's ROM BIOS, you need to install DRIVER.SYS to support that floppy disk drive. Include a **device=driver.sys** command in your CONFIG.SYS file. To give the drive a physical drive number and specify what kind of drive it is, include one or more of the following switches:

/d:*number* Assigns a physical drive number to a new drive. When you install a drive, its physical drive number is determined by the drive's physical position and the settings of your system's hardware DIP switches. Drive numbers range from 0 through 127. Drive numbers 0 and 1 are reserved for the first two floppy disk drives. Each **device=driver.sys** command must have a **/d** switch.

For example, **/d:3** specifies that the physical drive number is 3.

/c Indicates that the drive can detect when a drive door is open. If you use this switch, it is assumed that the disk drive supports change-line error detection. Check the documentation for your disk drive to see whether the drive supports change-line error detection.

/f:*number* Specifies the storage capacity of the disk drive you want to add. You can use any of the following values: 0=160K, 180K, 320K, or 360K; 1=1.2 MB; 2=720K; 7=1.44 MB; and 9=2.88 MB. The default value is 2.

/h:*number* Specifies the number of heads (sides) that a drive has. You can specify values from 1 through 99 (the default is 2).

/s:*number* Specifies the number of sectors per track on a disk drive. You can specify values from 1 through 99 (the default is 9).

/t:*number* Specifies the number of tracks per side on a disk drive. You can specify values from 1 through 999 (the default is 80).

The following table shows typical switch values you can use for each type of disk drive:

Drive type	/f	/h	/s	/t
360K or less	0	1 or 2	8 or 9	40
1.2-MB, 5.25-inch	1	2	15	80
720K, 3.5-inch	2	2	9	80
1.44-MB, 3.5-inch	7	2	18	80
2.88-MB, 3.5-inch	9	2	36	80

Suppose you want to add a 720K drive to your system. The drive has 2 sides (that is, two heads), 9 sectors per track, and 80 tracks per side. The drive is configured as drive 2. You would add the following **device** command to your CONFIG.SYS file:

```
device=c:\dos\driver.sys /d:2 /f:2
```

This command specifies that the DRIVER.SYS file is in the C:DOS directory. The **/d** switch specifies that the drive is configured as drive 2, using hardware switches. The **/f** switch specifies that the drive is a 720K drive. Because the **/f** switch includes a set of default values for heads, sectors, and tracks, you don't have to include switches that specify these values.

Assigning Two Drive Letters to a Drive

In Brief

To assign two logical-drive letters to the same physical device, include a **device** command in your CONFIG.SYS file. For example, if the DRIVER.SYS file is in the C:\DOS directory, the following command assigns the next available drive letter to the 1.2MB disk drive that has drive letter A:

```
device=c:\dos\driver.sys /d:0 /f:1
```

If you include in your CONFIG.SYS file a **device=driver.sys** command that uses the number of an existing drive, MS-DOS assigns an additional drive letter to the disk drive. If you assign two drive letters to a drive, MS-DOS prompts you to switch disks when you copy from one of the drive's logical-device letters to the other.

For example, use the following command to assign the next available drive letter to drive A (physical device 0):

```
device=c:\dos\driver.sys /d:0 /f:2
```

After this command is carried out, drive A has two drive letters associated with it: A, and the next available letter (D, for example). If a disk drive has two letters associated with it, you can copy files to and from that drive. For example, if drive A is also called drive D, you can use the following command to copy files to a different disk, using only one physical device:

```
copy a:*.* d:
```

MS-DOS prompts you to switch between the source disk and the destination disk.

Modifying Your Screen and Keyboard

You can use the ANSI.SYS installable driver to control the appearance of your screen and the functions of the keys on your keyboard. With the ANSI.SYS device driver, you can do the following:

- Change the character that a key displays
- Assign a command to a key

- Position the cursor anywhere on your screen
- Change the position of the command prompt
- Change the text format, text color, and background color of your screen

You can modify your keyboard and screen by using ANSI escape sequences. This section describes the most common ANSI escape sequences. For information about others, see Chapter 15, "Device Drivers."

NOTE Some programs may override changes you make by using ANSI escape sequences.

Understanding ANSI Escape Sequences

An ANSI *escape sequence* is a command you send to your console device (monitor or keyboard). This command is called an escape sequence because it begins with the escape (ESC) character and is not considered part of the typical output. It was developed by the American National Standards Institute (ANSI).

NOTE Unlike other MS-DOS commands, you cannot type an ANSI escape sequence at the MS-DOS prompt. For information about running an escape sequence, see "Running an ANSI Escape Sequence" later in this chapter.

ANSI escape sequences have a very different format from other commands you use with MS-DOS. An ANSI escape sequence begins with the ESC character and the left bracket ([). Following these two characters are the parameters. Last, there is a single-letter command name, which is case-sensitive. Do not put a space between any of the characters, as a space indicates the end of the command. If an ANSI escape sequence has more than one parameter, each parameter is separated by a semicolon (;), as in the following example:

```
ESC[3;12H
```

Understanding ASCII Codes

In many cases, ANSI escape sequences require ASCII codes as parameters. ASCII codes represent characters. Originally, ASCII consisted of 128 codes that represented the English alphabet, punctuation, and certain control characters. Currently, most systems recognize 256 codes: the original 128 ASCII codes plus an additional 128 codes called the extended character set. The extended character set includes a number of European characters, graphics characters, and scientific characters.

Each key on your keyboard has an ASCII code associated with it. When you press a key, MS-DOS determines which key it is and then assigns the corresponding ASCII code to it. Different codes are assigned to the key alone and to the key combined with SHIFT. In addition, most keys have codes for CTRL and ALT key combinations. The ASCII codes for keys that display a character (A, S, D, F, and so on) are numbers. For example, the ASCII code for uppercase A is 65.

Some keys don't display a character and have no corresponding ASCII code. (F1, DEL, and the UP ARROW key are examples of keys that have no ASCII code.) You can specify such a key by using its scan-code sequence, which consists of a signal and a numeric code. The signal for a scan code is a zero and semicolon (0;). The scan code immediately follows the signal. For example, the scan code for F1 is 59; the complete scan-code sequence for F1, including the scan-code signal, is 0;59.

Suppose you press SHIFT+2. MS-DOS assigns ASCII 64 to that key combination. Typically, ASCII 64 is an at sign (@). For a list of the ASCII codes MS-DOS typically assigns to each key, see the Appendix, "Keyboard Layouts and Their Code Pages."

NOTE The ASCII code that MS-DOS assigns to a key is affected by the current code page, the keyboard driver, and the display driver you are using.

Running an ANSI Escape Sequence

After installing ANSI.SYS, you can run an ANSI escape sequence in the following ways:

- Use the **prompt** command
- Put the ANSI escape sequence in an unformatted text file, and then use a **type** command to run it
- Put an ANSI escape sequence in an **echo** command in a batch program

Running an Escape Sequence by Using the Prompt Command

The **prompt** command is the most convenient way to run a single ANSI escape sequence because you can type the escape sequence directly from the keyboard and edit it by using the Doskey program. For information about Doskey, see Chapter 7, "Advanced Command Techniques" or the **doskey** command in Chapter 14, "Commands."

If you press ESC on your keyboard, MS-DOS cancels what you've typed at the command prompt. Because an ANSI escape sequence begins with ESC, you need a way to type ESC from the keyboard without canceling your command. The **prompt** command provides a way to do this. To specify ESC with the **prompt** command, type a **$e** combination. You can type **e** in uppercase or lowercase. For example, you can run the **m** escape sequence (which changes the foreground color to red and the background color to green) by using the following **prompt** command:

```
prompt $e[31;42m
```

Your command prompt is deleted when you use the **prompt** command to run an ANSI escape sequence. You can restore your command prompt by using the **prompt** command again, or you can add the characters that restore your command prompt (**pg**) to the ANSI escape sequence. For example, to run the **m** escape sequence and restore your prompt, type the following **prompt** command:

```
prompt $e[31;42m$p$g
```

This command changes the foreground color to red and the background color to green, and then creates a command prompt that shows the current drive and directory. Notice that there are no spaces in the command. If there is a space between the **$p** and **$g** in a **prompt** command, MS-DOS displays the space as part of the prompt.

Running an Escape Sequence from a Text File or Batch Program

To put an ANSI escape sequence in an unformatted text file or batch file, you can use any text-editing program that saves files as unformatted text and provides a way to create the ESC character. For example, in Microsoft Word, you can create an ESC character by pressing ALT+27. In MS-DOS Editor, you press CTRL+P and then ESC to create an ESC character.

If you have a number of escape sequences to run, it may be convenient to store them in an unformatted text file or batch file. There must be no spaces or carriage returns between the escape sequences in your file. Consequently, a file containing a series of escape sequences looks like one continuous line of characters. In this form, the file is very hard to read and edit. To avoid this problem, you could put each escape sequence on its own line while you are working in the file. Before you run the file, you must remove the carriage returns.

If the escape sequences are stored in a batch file, each line must begin with an **echo** command. To run a batch program, type the name of the batch file. To run a text file, use the **type** command followed by the name of the file.

Changing the Character That a Key Displays

In Brief

To change the character that a key displays, use the set keyboard strings (**p**) escape sequence. For example, the following command changes SHIFT+6 (^) to the ASCII 172 character (¼):

```
ESC["^";172p
```

This form of the **p** escape sequence has two parameters: the character or ASCII code assigned to the key you want to change and the character or ASCII code you want the key to have.

You can change the ASCII code that MS-DOS assigns to a key by using the ANSI set keyboard strings (**p**) escape sequence. Specify an ASCII code by typing its number or its character enclosed in quotation marks. For example, to change SHIFT+2 to display a plus sign (+) instead of an at sign (@), use the following **p** escape sequence:

```
ESC["@";"+"p
```

Notice that there are no spaces in the command and that the **p** follows the last parameter with no semicolon. Rather than type characters, you can type ASCII codes, as follows:

```
ESC[64;43p
```

When you use ASCII codes instead of characters, you don't use quotation marks. Once this command has been carried out, you can press SHIFT+2 to see a plus sign.

To return the key to its original code, type its original code as the first and second parameter, as follows:

```
ESC[64;64p
```

There are 256 ASCII codes. Many of the codes are not usually assigned to keys. To assign an unassigned code to a key, use the character's ASCII code in a **p** escape sequence. For example, if you want SHIFT+2 to display a check mark (✓)—which is ASCII 251—rather than an at sign (@), use the following escape sequence:

```
ESC["@";251p
```

The at sign is on most keyboards, so you have two ways to specify it: by typing an at sign (@) or by typing its ASCII code. The check mark is not on most keyboards, so you must specify it by typing its ASCII code.

Once a key has a different ASCII code, you can no longer type its character to generate it. To return SHIFT+2 to its usual assignment, use its original ASCII code as both the first and second parameter, as in this escape sequence:

```
ESC[64;64p
```

Assigning Commands to a Key

In Brief

To assign a command or sequence of commands to a key, use the ANSI set keyboard strings (**p**) escape sequence. For example, the following escape sequence assigns the command C:\WORK\MONDAY to CTRL+W (ASCII 23):

```
ESC[23;"c:\work\monday";13p
```

This form of the set keyboard strings (**p**) escape sequence can have several parameters. The first parameter always specifies the character or ASCII code of the key you want to change. The remaining parameters specify the operations you want the key to perform. In this example, two separate operations are assigned to CTRL+W: run the command C:\WORK\MONDAY, and generate an ASCII 13 character, which is equivalent to pressing ENTER on the U.S. keyboard.

Not only can you use the set keyboard strings (**p**) escape sequence to assign a single new character to a key, you can also use it to assign any number of characters to a key. For example, you can assign the word *percent* to the percent key (SHIFT+5) by using the following escape sequence:

```
ESC["%";"percent"p
```

After this command has been carried out, you can press SHIFT+5 and see the word *percent.*

You can use this capability of the set keyboard strings (**p**) escape sequence to assign one or more MS-DOS commands to a key. When you run a command from the command prompt, you type the command and press ENTER. When you assign a command to a key, you must specify these two operations: typing the command, and pressing ENTER. Enclose the command to be typed in quotation marks. Use the ASCII code 13 (carriage-return character) to generate ENTER. For example, use the following escape sequence if you want to view the short form of a directory, one screen at a time, by pressing SHIFT+7 (the ampersand key):

```
ESC["&";"dir /b /p";13p
```

You include two parameters for each command you want to assign to the key: the text of the command, and a carriage-return character (ENTER). Notice that you can type spaces inside a set of quotation marks.

To set F1 (scan-code sequence 0;59) so that it changes the current directory and starts the WORD.EXE program, use the following escape sequence:

```
ESC[0;59;"cd \edit";13;10;"word";13p
```

The command has six parameters: the code for the key, two commands enclosed in quotation marks, two carriage-return codes (ASCII 13), and one linefeed code (ASCII 10).

NOTE Many programs override key assignments.

Moving the Cursor

In Brief

To move the cursor, use the **A**, **B**, **C**, **D**, **H**, and **f** escape sequences.

For example, the cursor up (**A**) escape sequence moves the cursor up from the current position. The following escape sequence moves the cursor up two lines:

```
ESC[2A
```

When a command finishes running, the cursor returns to the position immediately to the right of the command prompt. If you run escape sequences by using a **prompt** command, you can specify where the cursor will appear after the command finishes running. If you run an ANSI escape sequence with a text file or batch program, you can change the position of the cursor in order to display text at a certain location. After your ANSI escape sequence runs, however, the cursor returns to the command prompt.

You use the following ANSI escape sequences to control cursor movement:

A	Moves the cursor up from its current position
B	Moves the cursor down from its current position
C	Moves the cursor right from its current position
D	Moves the cursor left from its current position
H	Moves the cursor to the line and character you specify (you can use **f** for the same results)
f	Moves the cursor to the line and character you specify (you can use **H** for the same results)

The **A** through **D** escape sequences move the cursor relative to its initial location. The **H** and **f** escape sequences move the cursor to the line and character you specify, regardless of where the cursor is currently located. If the cursor is already at the edge of the screen, and you run an escape sequence to move the cursor off your screen, the escape sequence is ignored, and the cursor remains where it is.

Positioning the Command Prompt

If you run a cursor-position (**H**) escape sequence with the **prompt** command, the cursor position you specify becomes the permanent position of the command prompt. For example, the following command moves the cursor to the upper-left corner of the screen (the "home" position), which becomes the permanent location of the command prompt:

```
prompt $e[H$p$g
```

This escape sequence moves the cursor to the home position and displays the current drive, current directory, and a greater-than sign (>) as the command prompt. Until you type the **prompt** command again, MS-DOS displays the command prompt in the upper-left corner of your screen.

When you use the cursor-position (**H**) escape sequence, the new prompt location is specified in screen coordinates. If you move the command prompt by using other escape sequences, you specify where the command prompt will be displayed in relation to its original position.

For example, the following command moves the command prompt to the fourth column on a line:

```
prompt $e[3C$p$g
```

Creating a Screen

If you combine cursor-movement commands with text, you can create a custom screen. For example, the following ANSI escape sequences in a batch program display an introductory screen for another program:

```
echo off
cls
echo ESC[2;H*WelcomeESC[2BESC[7D*to the*ESC[2BESC[6D*Database
Utility*ESC[12;1H(For Help, press F1)
prompt $e[10;1H Enter a Command:
```

The escape sequences position the cursor and display text. The screen produced by this batch program looks like this:

```
*Welcome*

  *to the*

    *Database Utility*

Enter a command:

(For Help, press F1)
```

The command prompt remains on line 10 of the screen until you reposition it.

Changing Screen Attributes

In Brief

To change screen attributes, use the ANSI set graphics mode (**m**) escape sequence. For example, the following command makes text bold:

```
ESC[1m
```

This command makes text magenta and bold:

```
ESC[1;35m
```

The **m** escape sequence can have as many parameters as you want.

There are three kinds of screen attributes: text format, text color, and background color. The text-format attribute specifies whether text is bold, underscored, blinking, or hidden. Text-color attributes specify text color. Background-color attributes specify screen color. If you specify an attribute that your system doesn't support, MS-DOS ignores the setting and no change occurs.

You can use the ANSI set graphics mode (**m**) escape sequence to change the appearance of your screen. After you set an attribute, any new text displayed has that attribute. For example, the following command makes text bold:

```
ESC[1m
```

By default, text is white and the background is black. To return to the default setting, use 0 as the parameter, as in the following example:

```
ESC[0m
```

You can set as many attributes as you like by using one escape sequence. For example, the following command sets text to blink (5) and to appear red (31) against a blue background (44):

```
ESC[5;31;44m
```

The order in which you type the parameters is not important. However, each parameter must be separated from the others by a semicolon (;).

Changing Text Format

The following is a list of text-format attributes:

1	Bold
4	Underscored (monochrome monitors only)
5	Blinking
8	Hidden

The hidden-text attribute prevents the display of text, unless you subsequently change the background color.

The first ANSI escape sequence in the following example displays the word *WARNING* in bold, blinking letters at the current cursor position:

```
ESC[1;5mWARNING
ESC[0m
```

The second escape sequence resets all attributes so that new text will not be bold and blinking.

The following **prompt** command makes the greater-than sign (>) in your command prompt bold:

```
prompt $p$e[1m$g$e[0m
```

Text following the prompt will not be bold, since the 0 parameter is used after the **$g**.

Changing Text Color

The following attributes specify text color:

30	Black
31	Red
32	Green
33	Yellow
34	Blue
35	Magenta
36	Cyan
37	White
7	Black text on a white background

The first ANSI escape sequence in the following example displays the word *WARNING* in red, bold, blinking letters:

```
ESC[31;1;5mWARNING
ESC[0m
```

The second escape sequence resets the attributes so that new text will have the default settings.

The following **prompt** command makes the drive and directory in your command prompt bold and blue, and the greater-than sign bold and red:

```
prompt $e[1;34m$p$e[31m$g$e[0m
```

Changing Background Color

The following attributes specify background color:

40	Black
41	Red
42	Green
43	Yellow
44	Blue
45	Magenta
46	Cyan
47	White
7	Black text on a white background

The first escape sequence in the following example displays the word *WARNING* in red, bold, blinking letters on a yellow background:

```
ESC[43;31;1;5mWARNING
ESC[0m
```

The second escape sequence resets the attributes to the default settings.

If you use the **ESC[7m** attribute, MS-DOS displays black text on a white background. Even if you change the text color, the background remains white until you explicitly change the background color or reset the attributes so that new text will have the default settings.

NOTE The screen attributes meet the ISO 6429 standard.

Chapter 12

Optimizing Your System

12

To *optimize* your system is to customize it so that it uses its resources most efficiently for the tasks you usually perform. Typically, optimizing involves improving one or more aspects of your system's performance, but sometimes sacrificing something else. In the MS-DOS environment, optimizing your system usually means balancing speed against memory. Typically, you have one of two goals:

- To make more memory available for programs. You might want to free memory, even at the cost of some speed, if there is not enough memory available to run certain programs.
- To improve your system's speed as much as possible, while retaining enough memory to run your programs. You can improve speed both by using your hard disk more efficiently and by installing MS-DOS utility programs.

This chapter explains how to use your system's resources to best advantage, whether you want to free memory or improve your system's speed.

For an overview of system resources and how they relate to system performance, see the following section, "Understanding System Resources."

If you are already familiar with memory and storage concepts, you might want to skip to the actual procedures for the type of optimization you want to do. If you want to make more memory available, see "Making More Memory Available" later in this chapter. If you want to speed up your system, see "Speeding Up Your System."

If you are an advanced user, you might want to see "Running Programs in the Upper Memory Area" later in this chapter, for additional information about conserving memory.

For a quick summary of optimization procedures, see "Optimization Summary" at the end of this chapter.

Understanding System Resources

In the MS-DOS environment, the most important system resources are memory and disk space. The available resources can affect all of the following:

- Which programs you can run
- How fast programs run
- How much data a program can work with at one time
- How much data you can store from one session to the next

The rest of this section describes how memory and disk space can affect your system's performance.

Understanding Memory

Memory provides temporary storage for programs and data. It exists on the main system board of your computer or on add-in memory boards. All programs must be loaded into memory in order to run.

In general, the more memory you have, the more data you can store in memory at one time. Some programs require more memory than others. You can increase the amount of memory on your system by plugging a memory board into a slot inside your computer. For example, you might add a 2-megabyte memory board to a system that has 1 MB of memory on its main system board; the system would then have 3 MB of memory.

Your system can have as many as three kinds of memory:

- Conventional
- Extended
- Expanded

In addition, most system have an upper memory area.

To find out what kind of memory your system has, and how much, use the **mem** command. For information about the **mem** command, see Chapter 14, "Commands."

Programs that run with MS-DOS normally use your system's conventional memory. In order for programs to use extended or expanded memory, or the upper memory area, you must install a memory manager that provides access to that memory. The rest of this section explains each kind of memory and describes the memory managers that come with MS-DOS.

Conventional Memory

Conventional memory is the basic type of memory found on all computers. Most computers have at least 256 kilobytes of conventional memory and can accommodate up to 640K. Programs can use conventional memory without the special instructions needed to use other types of memory.

MS-DOS uses some conventional memory. The device drivers and commands listed in your CONFIG.SYS and AUTOEXEC.BAT files use additional conventional memory. The memory left over is available for other programs.

Extended Memory (XMS)

One way to add more memory to your system is to install *extended memory*. Extended memory is available only on systems with 80286 or higher processors. (Many 80286 and 80386 computers come with 640K of conventional memory and 384K of extended memory.)

Most programs that use conventional memory cannot use extended memory because the numbers or *addresses* that identify locations in extended memory to programs are beyond the addresses most programs can recognize. Only the addresses in the 640K of conventional memory are recognized by all programs.

Programs need special instructions to recognize the higher addresses in extended memory. Extended memory is fast and efficient for programs that can use it. However, many programs are not designed to use extended memory.

To use extended memory efficiently, you should install a program called an *extended-memory manager*. An extended memory manager prevents different programs from using the same part of extended memory at the same time. The extended-memory manager also makes it easier for programs to use extended memory. MS-DOS includes the extended-memory manager HIMEM.SYS. HIMEM conforms to the Lotus/Intel/Microsoft/AST eXtended Memory Specification (XMS) version 2.0, which specifies a standard way for programs to use extended memory cooperatively.

MS-DOS can run in extended memory, leaving more conventional memory available for programs. Extended memory is also the best choice for memory

expansion if you use Microsoft Windows version 3.0 or later, since Windows works best with extended memory.

NOTE If you have an 80386 or 80486 system with extended memory and you use programs that can take advantage of expanded memory, you might want to install EMM386.EXE. EMM386 is a device driver that can use extended memory to simulate expanded memory. For information about expanded memory, see the following section, "Expanded Memory (EMS)." For information about EMM386 and expanded memory, see "Using EMM386 as an Expanded-Memory Emulator" later in this chapter.

Expanded Memory (EMS)

Another way to add memory in excess of 640K to your system is to install *expanded memory*. Most computers can accommodate expanded memory, which consists of two parts: an expanded-memory board, which must be installed on your computer; and a program called an *expanded-memory manager*, which comes with the expanded-memory board.

A program designed to use expanded memory does not have direct access to the information in expanded memory. Instead, expanded memory is divided into 16K segments called *pages*. When a program requests information that is in expanded memory, the expanded-memory manager *maps* or copies the appropriate page to an area called a *page frame*. (The page frame exists in the upper memory area, discussed in the next section.) A program gets the information from the page frame.

Expanded-memory boards and managers conform to the Lotus/Intel/Microsoft Expanded Memory Specification (LIM EMS) version 3.2 or 4.0, which specifies how programs make use of expanded memory.

Some programs are unable to use expanded memory because they were not designed to interact with an expanded-memory manager. However, because expanded memory was introduced before extended memory, more programs are designed to use expanded memory than to use extended memory.

Because an expanded-memory manager allows programs access to a limited amount of information at one time, expanded memory can be slower and more cumbersome for programs to use than extended memory.

Upper Memory Area

Most systems have 384K of space called the *upper memory area*. This area is immediately adjacent to the 640K of conventional memory. The upper memory area is not considered part of the total memory of your computer because

programs cannot store information in this area. This area is normally reserved for running your system's hardware, such as your monitor.

Information can be mapped (or copied) from another type of memory to parts of the upper memory area left unused by your system. These unused parts are called *upper memory blocks*. (One use of this mapping is for running programs that use expanded memory. For more information on expanded memory, see the previous section.)

If you have a system with an 80386 or 80486 processor and extended memory, MS-DOS can use the upper memory area to free up more conventional memory on your computer. MS-DOS has commands that enable you to store certain device drivers and programs outside of conventional memory, usually in extended memory. MS-DOS will then map these device drivers and programs into the upper memory area, where they can run successfully. The number of device drivers and programs you can run in the upper memory area depends on how much of the upper memory area is left unused by your system and the expanded memory page frame, if you are using one.

For information about how to run device drivers and programs in the upper memory area, see "Running Programs in the Upper Memory Area" later in this chapter.

MS-DOS Memory Managers

To use your computer's extended memory, expanded memory, or upper memory area, you must install a *memory manager*. A memory manager is a device driver that provides access to a particular type of memory. (You do not need to install a memory manager to use conventional memory, since MS-DOS has a built-in conventional-memory manager.)

MS-DOS includes the following installable memory managers:

- HIMEM, which provides access to extended memory. For information about HIMEM, see "Using the HIMEM Extended-Memory Manager" later in this chapter.
- EMM386, which uses extended memory to simulate expanded memory. For information about this use of EMM386, see "Freeing Expanded Memory" later in this chapter. EMM386 can also provide access to the upper memory area. For information about this use of EMM386, see "Running Programs in the Upper Memory Area" later in this chapter.

MS-DOS does not include an expanded-memory manager, since each expanded-memory board requires its own memory manager. To use expanded memory, you must install the memory manager that came with your expanded-memory board. For information about installing your expanded-memory manager, see the documentation that accompanied your memory board.

You install a memory manager by using a **device** command in your CONFIG.SYS file. Although memory managers take up some conventional memory, they make up for it by providing access to much larger amounts of extended memory, expanded memory, or the upper memory area.

NOTE MS-DOS also includes the device drivers SMARTDrive and RAMDrive. Although you start these programs by using the **device** command, they are not memory managers. Instead, they are optimization programs that use some memory in order to speed up your system. For information about SMARTDrive and RAMDrive, see "Speeding Up Your System" later in this chapter.

Understanding Disk Space

A disk provides both long-term and temporary storage for program and data files. The most common types of storage media are floppy disks and hard disks. Files are stored on a disk magnetically, much as information is stored on a cassette tape.

After you install all the programs and data files you need, there should be some disk space still available. There are two reasons you might need free disk space:

- To save documents and other data files
- To enable programs to store temporary files and data while they're running

You should keep track of how much disk space is available on your system. The amount of free disk space can affect your ability to store files and run programs. You can use the **chkdsk** and **dir** commands to check the amount of free disk space. For information about these commands, see Chapter 14, "Commands."

Making More Memory Available

If you're having trouble running programs because there isn't enough memory, your main optimization goal should be to make more memory available to those programs.

To run a program, your system must contain as much physical memory as that program requires. For example, if a program requires 512K of memory, it isn't going to run on a system that has only 256K of memory, no matter how much memory you free.

If your system does contain sufficient memory, a program still might not run. The cause is often that memory-resident programs are taking up some memory, and there is not enough memory left over. Usually, the problem is caused by insufficient conventional memory. However, with a few programs, the problem is caused by insufficient expanded or extended memory. This section explains how to make more memory of all types available to programs. It covers the following topics:

- Using the HIMEM extended-memory manager
- Freeing conventional memory
- Freeing extended memory
- Freeing expanded memory

Using the HIMEM Extended-Memory Manager

HIMEM is an extended-memory manager included with MS-DOS. It provides access to extended memory and ensures that no two programs can use the same part of extended memory at the same time.

You must install HIMEM if you want to use your system's extended memory.

NOTE The version of HIMEM that comes with MS-DOS version 5.0 supersedes the version of HIMEM that comes with Microsoft Windows version 3.0. If you have both MS-DOS 5.0 and Windows 3.0, use the version of HIMEM that comes with MS-DOS.

Advantages

The following are the advantages of using HIMEM:

- Makes extended memory available to programs that use extended memory according to XMS (the Extended Memory Specification).
- Prevents system errors that can result when programs make conflicting memory requests.
- In conjunction with EMM386, enables you to run MS-DOS in extended memory to conserve conventional memory. (For information about running MS-DOS in extended memory, see "Running MS-DOS in Extended Memory" later in this chapter.)
- Enables you to use parts of the upper memory area to conserve conventional memory, if you have an 80386 or 80486 system. (To do this, you also need to install EMM386. For information about running programs in the upper memory area, see "Running Programs in the Upper Memory Area" later in this chapter.)
- Enables EMM386 to use your system's extended memory to emulate expanded memory for programs that need expanded memory, if you have an 80386 or 80486 system.
- Is compatible with Microsoft Windows version 3.0 or later.

Disadvantages

The following are the disadvantages of using HIMEM:

- Uses a small amount of conventional memory.
- Might not be compatible with older programs that allocate extended memory directly, rather than by using an extended-memory manager. For example, Microsoft Windows/386 version 2.*x* will not run with the version of HIMEM included with MS-DOS version 5.0.

Recommendations

The following are some basic recommendations for using HIMEM:

- Install HIMEM if you have an 80286, 80386, or 80486 system with extended memory.

- Make sure the **device** command for HIMEM appears in your CONFIG.SYS file before any commands that start device drivers or programs which use extended memory. For example, since EMM386 uses extended memory, the **device** command for EMM386 must come after that for HIMEM.

▶ To install HIMEM:

1. Use a text editor such as MS-DOS Editor to open your CONFIG.SYS file. (Your CONFIG.SYS file is usually located in the root directory of your startup disk.)
2. Add a **device** command for HIMEM.SYS to the beginning of your CONFIG.SYS file. This **device** command must come before any **device** commands for device drivers that use extended memory.

   The command line for HIMEM specifies the location of the HIMEM program file, how HIMEM should manage memory, and your system type. The following command runs HIMEM, using default values:

   ```
   device=c:\dos\himem.sys
   ```
3. Save the changes to your CONFIG.SYS file.
4. Restart your computer by pressing CTRL+ALT+DEL.

For information about HIMEM parameters, see Chapter 15, "Device Drivers."

Freeing Conventional Memory

All programs require conventional memory in order to run. If a program fails to run because of insufficient memory, the problem is most often because of a shortage of conventional memory.

You can make more conventional memory available to your programs by minimizing how much memory MS-DOS, installable device drivers, and other memory-resident programs use. A program can use only the conventional memory that is available when you start it. If memory-resident programs are already using memory, the program cannot use that memory.

There are several ways to free conventional memory for use by programs:

- Run MS-DOS in extended memory instead of in conventional memory, if your system has extended memory.
- Streamline your CONFIG.SYS and AUTOEXEC.BAT files so that they don't start unnecessary memory-resident programs.

- Run device drivers and other memory-resident programs in the upper memory area instead of in conventional memory, if you have an 80386 or 80486 computer.

The following sections explain the first two methods. Running programs in the upper memory area can be a complex process and is explained in a separate section, "Running Programs in the Upper Memory Area," later in this chapter.

Running MS-DOS in Extended Memory

Normally, MS-DOS runs in conventional memory. This makes less conventional memory available to programs. However, if your system has extended memory, MS-DOS can run in extended memory. When it does, it uses the first 64K of extended memory, called the *high memory area* (HMA). Because few programs use the HMA, it makes sense to run MS-DOS there.

NOTE If your system has extended memory, the MS-DOS Setup program normally installs MS-DOS so that it will automatically run in the HMA.

Advantages

The following are the advantages of running MS-DOS in extended memory:

- Frees conventional memory.
- Works on any computer that has extended memory.
- Uses the HMA, a part of extended memory that few programs use.
- Loads most of HIMEM into the HMA, freeing more conventional memory.

Disadvantages

The following are the disadvantages of running MS-DOS in extended memory:

- Requires that your system have extended memory.
- Prevents programs from using the HMA. This is not usually a serious problem, since few programs require use of the HMA.

Recommendation

The following is the basic recommendation for running MS-DOS in extended memory:

- Run MS-DOS in the HMA if your system has extended memory.

To load MS-DOS into the HMA:

1. Use a text editor such as MS-DOS Editor to open your CONFIG.SYS file. (Your CONFIG.SYS file is usually located in the root directory of your startup disk.)
2. Make sure that your CONFIG.SYS file contains the following commands:

   ```
   device=himem.sys
   dos=high
   ```

   These commands first load the HIMEM extended-memory manager and then load MS-DOS into extended memory.
3. Save the changes to your CONFIG.SYS file.
4. Restart your computer by pressing CTRL+ALT+DEL.

Streamlining Your CONFIG.SYS and AUTOEXEC.BAT Files

When you start your system, the commands in your CONFIG.SYS and AUTOEXEC.BAT files can start device drivers and other programs that use memory. You can make more memory available to programs by removing unnecessary commands from these files.

To effectively streamline your CONFIG.SYS and AUTOEXEC.BAT files, you should know the purpose of each of the commands in those files.

CAUTION Use care when changing your CONFIG.SYS and AUTOEXEC.BAT files. If you incorrectly change some values or disable some commands, your system may not function properly.

Recommendations for Streamlining Your CONFIG.SYS File

As explained in Chapter 11, "Customizing Your System," your CONFIG.SYS file is a text file that starts device drivers and specifies your MS-DOS configuration. For example, a typical CONFIG.SYS file might specify the location of the MS-DOS file COMMAND.COM, start an extended-memory manager, and

specify how many files a program can have open at once. MS-DOS runs the commands in your CONFIG.SYS file before those in your AUTOEXEC.BAT file.

The following basic recommendations can help you conserve conventional memory by streamlining your CONFIG.SYS file:

- Include **device** commands only for device drivers that you really need. You can also disable **device** commands for any unnecessary device drivers by using the **rem** command. For information about disabling commands, see "Modifying Your CONFIG.SYS and AUTOEXEC.BAT Files" later in this section.

- If your system has expanded memory, include a **device** command for the expanded-memory manager that came with your memory board.

- If your system has extended memory, include a **device** command for the HIMEM.SYS extended-memory manager. Also include the following command:

  ```
  dos=high
  ```

  The **dos=high** command saves conventional memory by running MS-DOS in extended memory.

- If your CONFIG.SYS file contains a **device** command for SMARTDrive, RAMDrive, or the Fastopen program, disable that command to conserve conventional memory. (SMARTDrive, in particular, can use a lot of conventional memory.) If you use RAMDrive, make sure your RAM disk is in expanded or extended memory, not conventional memory.

- If your CONFIG.SYS file contains a **buffers** command, reduce the number of buffers. (Each buffer takes up about 500 bytes.) Because some programs might not run properly if you reduce the number too much, do not specify fewer than 10 or 15 buffers, unless you are using another caching scheme such as SMARTDrive.

- Add a **stacks** command to limit the number and size of interrupt stacks that MS-DOS uses. By default, MS-DOS uses 0 interrupt stacks for IBM PC, IBM PC/XT, IBM PC-Portable, and compatible machines; and 9 stacks for IBM PC/AT, IBM PS/2, and compatible machines. You can conserve memory by setting **stacks** to zero, as follows:

  ```
  stacks=0,0
  ```

  On a few systems, setting **stacks** to zero might cause problems with Microsoft Windows version 3.0. If you set **stacks** to zero and your system occasionally

locks up while you are running Windows in 386 enhanced mode, disabling the **stacks** command might solve the problem.

- If your CONFIG.SYS file includes the **lastdrive** command, set **lastdrive** to a letter such as J or K, rather than Z. (Each letter uses about 100 bytes more than the preceding letter.) If you use a network, this might limit the number of network drives you can use simultaneously.

- If your CONFIG.SYS file contains an **fcbs** command, set **fcbs** to 1.

The order of the **device** and **devicehigh** commands in your CONFIG.SYS file can be important: It affects both the efficient use of memory and the proper operation of the various programs that CONFIG.SYS starts.

The following list shows the order in which you should start device drivers from your CONFIG.SYS file:

1. HIMEM.SYS.
2. Your expanded-memory manager, if your system has physical expanded memory.
3. Any device drivers that use extended memory.
4. EMM386.EXE.

   If you are using EMM386 both to simulate expanded memory and to provide access to the upper memory area, the EMM386 command line should include the **ram** switch rather than the **noems** switch. Starting EMM386 with the **noems** switch prevents EMM386 from simulating expanded memory. Do not use EMM386 if you are using an expanded-memory manager.
5. Any device drivers that use expanded memory.
6. Any device drivers that use the upper memory area. For information about loading device drivers into the upper memory area, see the **devicehigh** command in Chapter 14, "Commands."

NOTE This list is intended to show only the optimal order in which your CONFIG.SYS file should start device drivers. It is not intended to be a list of the commands that your CONFIG.SYS file should contain. The contents of your system's CONFIG.SYS file depend on the type of system, the amount and type of memory, the hardware configuration, and the programs you use.

Recommendations for Streamlining Your AUTOEXEC.BAT File

As explained in Chapter 11, "Customizing Your System," your AUTOEXEC.BAT file is a special MS-DOS batch program located in the root directory of your hard disk, usually drive C. MS-DOS runs the commands in your AUTOEXEC.BAT file immediately after it runs the commands in your CONFIG.SYS file.

Typically, an AUTOEXEC.BAT file starts memory-resident programs such as network programs and sets up environment variables. In addition, your AUTOEXEC.BAT file might define your command prompt.

The following basic recommendations can help you conserve conventional memory by streamlining your AUTOEXEC.BAT file:

- Disable commands that start memory-resident programs you don't need. For information about disabling commands, see the next section, "Modifying Your CONFIG.SYS and AUTOEXEC.BAT Files."
- If you use a mouse only with Microsoft Windows, which has a built-in device driver, disable any commands that start and enable mouse device drivers such as MOUSE.COM.

Modifying Your CONFIG.SYS and AUTOEXEC.BAT Files

Before you change your CONFIG.SYS or AUTOEXEC.BAT files to streamline your system, it's important to back up the existing version of the files. Then, if your changes cause any problems, you can easily restart your computer by using the backup versions, and then correct the modified files.

▶ To modify your CONFIG.SYS and AUTOEXEC.BAT files:

1. Use the **sys** command to create a startup disk. (For information about the **sys** command, see Chapter 14, "Commands.")
2. Copy both your CONFIG.SYS file and your AUTOEXEC.BAT file to the startup disk. Remove the startup disk from the drive.
3. Use a text editor such as MS-DOS Editor to open and edit your CONFIG.SYS or AUTOEXEC.BAT file.
4. Disable any commands that load unnecessary device drivers and utility programs.

   It is better to disable a command than to delete it, because if you accidentally disable a command you really need, you can restore it easily. To disable a

command, insert a **rem** command at the beginning of the command line. For example, suppose you want to disable the following CONFIG.SYS command:

```
device=c:\device\mouse.sys
```

You would add the **rem** command to the command line, as follows:

```
rem device=c:\device\mouse.sys
```

5. Save the file.
6. When you have finished editing both files, restart your computer by pressing CTRL+ALT+DEL.

   If your system doesn't start properly, insert in drive A the floppy disk you created in step 1, and start your computer again. If you know which command(s) are causing the problem, edit the appropriate file (CONFIG.SYS or AUTOEXEC.BAT) on your hard disk and restart your computer. Or, to start over, copy the backup version of the files from the floppy disk to your hard disk.

Freeing Extended Memory

A few programs require additional extended memory in order to run. If you are having trouble running such a program, do the following:

- Make sure your system contains as much physical extended memory as the program needs.
- Make sure your CONFIG.SYS file contains a **device** command for the HIMEM.SYS extended-memory manager (or another memory manager that conforms to the XMS specification). Most programs need an extended-memory manager in order to use extended memory.
- If your CONFIG.SYS file contains a **device** command for SMARTDrive, RAMDrive, or EMM386, make sure that those programs are not using all of your extended memory. You can reduce the amount of extended memory you allocate for each device driver by changing the **device** command for that driver. You can also disable those **device** commands by using the **rem** command. For information about disabling commands, see "Modifying Your CONFIG.SYS and AUTOEXEC.BAT Files" earlier in this chapter.
- Make sure your CONFIG.SYS and AUTOEXEC.BAT files don't start unnecessary programs that use extended memory. For information about modifying these files, see "Streamlining Your CONFIG.SYS and AUTOEXEC.BAT Files" earlier in this chapter.

- If the program doesn't start and displays a message such as "High Memory Area (HMA) already in use," free the high memory area for that program.

  Few programs require use of the HMA. If your program does require the HMA and your CONFIG.SYS file contains the command **dos=high**, then MS-DOS is using the HMA. To free the HMA for use by your program, disable the **dos=high** command. Doing so will cause MS-DOS to run in conventional memory rather than in the HMA. (If your CONFIG.SYS file contains the command **dos=high,umb**, do not disable the command; instead, change it to **dos=umb**.)

Freeing Expanded Memory

Some programs require additional expanded memory in order to run. If you are having trouble running such a program, do the following:

- Make sure your system contains as much physical expanded memory as the program needs.

  If you have an 80386 computer with extended memory, you can use EMM386 to provide expanded memory for programs. For information about EMM386, see the next section, "Using EMM386 as an Expanded-Memory Emulator."

- If your system contains physical expanded memory, make sure your CONFIG.SYS file contains a **device** command for the expanded-memory manager that came with your memory board. This command is required in order for programs to use expanded memory.

- If your CONFIG.SYS file contains a **device** command for SMARTDrive or RAMDrive, make sure that those programs aren't using all of your expanded memory. You can reduce the amount of expanded memory you allocate for SMARTDrive or RAMDrive by changing the **device** command for that driver. You can also disable those **device** commands by using the **rem** command. For information about disabling commands, see "Modifying Your CONFIG.SYS and AUTOEXEC.BAT Files" earlier in this chapter.

- Make sure your CONFIG.SYS and AUTOEXEC.BAT files don't start unnecessary programs that use expanded memory. For information about modifying these files, see "Streamlining Your CONFIG.SYS and AUTOEXEC.BAT Files" earlier in this chapter.

- If you are already using EMM386 as an expanded-memory emulator, make more expanded memory available to programs by allowing EMM386 more extended memory. EMM386 can then use the additional extended memory to provide more expanded memory for your programs.

Using EMM386 as an Expanded-Memory Emulator

An *expanded-memory emulator* is a program that can use extended memory to simulate expanded memory. Programs can then use that simulated expanded memory as if it were physical expanded memory. The EMM386 device driver, which comes with MS-DOS, can function as an expanded-memory emulator on 80386 and 80486 computers. (EMM386 can also function as an upper-memory-area manager; for information about this use of EMM386, see "Installing EMM386 to Manage the Upper Memory Area" later in this chapter.)

NOTE EMM386 is for use on 80386 and 80486 computers only. To use a program that requires expanded memory on an 80286 or 8086 computer, you must configure your system so that it provides as much physical expanded memory as the program needs.For information about expanded memory, see "Understanding Memory" earlier in this chapter.

The version of EMM386 that comes with MS-DOS version 5.0 supersedes the version that comes with Microsoft Windows version 3.0. If you have both MS-DOS 5.0 and Windows 3.0, use the version of EMM386 that comes with MS-DOS.

Do not use EMM386 if you are using another expanded-memory manager.

Advantages

The following are the advantages of using EMM386 to simulate expanded memory:

- Provides expanded memory for systems that have only extended memory.
- Can improve the speed of some programs that use expanded memory, if your system has no physical expanded memory.

Disadvantages

The following are the disadvantages of using EMM386 to simulate expanded memory:

- Works only on 80386 and 80486 computers.
- Uses extended memory. EMM386 uses about 80K of extended memory to run. Also, any memory used to emulate expanded memory is no longer available as extended memory.

Recommendations

The following are some basic recommendations for using EMM368 to emulate expanded memory:

- Use EMM386 as an expanded-memory emulator only if you have an 80386 or 80486 system with extended memory and you want to use programs that require expanded memory.
- If you use EMM386 as an expanded-memory emulator, allocate it only as much extended memory as a program needs. For example, if you want to run a program that requires 256K of expanded memory, you would allocate 256K of extended memory to EMM386.
- If you use EMM386 both as an expanded-memory emulator and as an upper-memory-area manager, use the **ram** switch with the **device** command that starts EMM386. Otherwise, programs will not be able to use the expanded memory that EMM386 provides.
- If you use Microsoft Windows version 3.0 or later, use EMM386 as an expanded-memory emulator only if you run programs that need expanded memory outside Microsoft Windows. When running in 386 enhanced mode, Windows can simulate expanded memory for programs that need it. For information about 386 enhanced mode, see the *Microsoft Windows User's Guide*.

▶ To install EMM386 as an expanded-memory emulator:

1. Use a text editor such as MS-DOS Editor to open your CONFIG.SYS file. (Your CONFIG.SYS file is usually located in the root directory of your hard disk.)

2. Add a **device** command for EMM386 to your CONFIG.SYS file. (If your CONFIG.SYS file already contains a **device** command for EMM386, edit that command.)

   The **device** command for EMM386 must come after the **device** command for HIMEM and before any commands for device drivers that use expanded memory. The command specifies the location of the EMM386.EXE file; it also specifies that you want EMM386 to emulate expanded memory and indicates the amount of extended memory to allocate to EMM386. EMM386 then provides that amount of expanded memory to programs that need it. The following **device** command specifies that EMM386 should use extended memory to simulate expanded memory and allocates 640K of extended memory for that purpose:

   ```
   device=c:\dos\emm386.exe 640
   ```

3. Disable or remove any other **device** commands for expanded-memory managers.
4. Save the changes to your CONFIG.SYS file.
5. Restart your system by pressing CTRL+ALT+DEL.

Speeding Up Your System

You can use several methods to improve your system's speed and the speed of the programs you use. This section explains how to do the following:

- Speed up your system without using more memory
- Use the **buffers** command
- Use the Fastopen program
- Use the SMARTDrive disk-caching program
- Use the RAMDrive memory-disk program

Speeding Up Your System Without Using More Memory

You can use the following methods to speed up your system without taking up additional memory. These methods improve your system's speed by improving the efficiency of your hard disk:

- Delete unnecessary files. For more information, see "Deleting Unnecessary Files" later in this section.
- Use the **chkdsk /f** command to recover lost disk space, and then delete the files **chkdsk** creates. For more information, see "Using the Chkdsk Command" later in this section.
- Make sure MS-DOS is searching for files in the most efficient order. For more information, see "Helping MS-DOS Find Files Quickly" later in this section.
- Reorganize files on your hard disk. For more information, see "Reorganizing Your Hard Disk to Improve Speed" later in this section.
- Adjust your hard-disk controller's *interleave*. The interleave affects how many times the disk must revolve in order for an entire disk track to be read. For more information, see "Adjusting Your Hard-Disk Interleave" later in this section.

Deleting Unnecessary Files

As explained in "Understanding Disk Space" earlier in this chapter, disk space can be a valuable system resource. If you need more disk space, an easy solution is to delete unnecessary files. There are three categories of files you might want to delete:

- Program and data files that you no longer use.
- Temporary files that were left on your hard disk when a program ended unexpectedly.
- MS-DOS files that were installed automatically and that you don't plan to use. The table at the end of this section gives information about the MS-DOS files you can delete.

CAUTION Do not delete any MS-DOS files other than the ones listed in the table in this section.

It is important to delete unnecessary files before you compact your hard disk (see "Using a Disk-Compaction Program" later in this chapter). In general, keep as much disk space free as possible. To delete unnecessary files, you can use the **del** command. You can use the following guidelines for deciding which files to delete:

- Delete any temporary files created by your programs.

  Many programs create temporary files while they are running. Some programs store those files in a separate directory that is specified in your AUTOEXEC.BAT file by using the **set** command. (Most often, you designate such a directory by using the the **set** command with the TEMP or TMP environment variable.

  You should periodically clean out your TEMP directory. (This is not necessary if your TEMP directory is on a RAM disk.) To avoid deleting a temporary file that is currently in use, you should delete files in your TEMP directory only when you are not running any programs.

- If your system is very short on disk space, delete some MS-DOS files.

  If you plan to delete MS-DOS files, you should first use the MS-DOS Setup program to install MS-DOS on floppy disks. This makes it easy for you to restore individual files later.

The following table provides more information about the MS-DOS files you can delete.

Filename(s)	Description	When to delete
EMM386.EXE	Memory manager	If your computer is not an 80386 or 80486; or if your computer is an 80386 or 80846 and you do not use programs that require expanded memory or run programs in the upper memory area.
RAMDRIVE.SYS	RAMDrive memory-disk program, used to speed up your system	If you do not need a RAM disk, or if your system has only conventional memory.
SMARTDRV.SYS	SMARTDrive disk-caching program, used to speed up your system	If your system does not have a hard disk, or if your system has only conventional memory.
NLSFUNC.EXE, GRAFTABL.COM, KEYB.COM, *.CPI, COUNTRY.SYS, DISPLAY.SYS, KEYBOARD.SYS, PRINTER.SYS	Files that provide international support and code-page support	If you are in the U.S. and do not need international (foreign language) support.
EXE2BIN	Programming tool	If you do not plan to do any programming.

CAUTION Never delete the files COMMAND.COM, IO.SYS, or MSDOS.SYS. (The IO.SYS and MSDOS.SYS files are usually hidden files.) If you delete any of these files, your system will not start.

Using the Chkdsk Command

You can use the **chkdsk** command to recover lost *allocation units* that are taking up space on your hard disk. An allocation unit is the smallest piece of a hard disk that can be allocated to a file. Allocation units can get lost when a program ends unexpectedly, leaving temporary files on the hard disk without saving or deleting them properly. Over time, lost allocation units can accumulate and take up disk space.

When you use the **/f** switch with the **chkdsk** command, **chkdsk** converts lost allocation units to visible files that you can examine and delete.

CAUTION Before using **chkdsk /f**, make sure you aren't running any programs. You may need to disable memory-resident programs in your CONFIG.SYS and AUTOEXEC.BAT files and restart your system. You might lose data if you use this command while programs are running.

You can use the **chkdsk /f** command to do the following:

- Make sure there are no lost allocation units on your disk
- Check your hard disk before compacting your disk (running a defragmentation utility)
- Check your hard disk after a program ends unexpectedly

Make sure you quit all programs before using **chkdsk**. If you use the Fastopen program, the SMARTDrive program, or other memory-resident programs, disable the corresponding commands in your CONFIG.SYS file and restart your system, to ensure that those programs don't interfere with the disk-compaction process.

▶ To clean up lost allocation units by using chkdsk:

1. Quit all programs.
2. Change to the hard disk you want to clean up (for example, to clean up files on drive D, you would type **d:** at the prompt).
3. Type **chkdsk /f**. The **/f** switch finds and recovers any lost allocation units.

4. If **chkdsk** finds any lost allocation units, it prompts you to convert them to files. If you want to inspect the contents of the lost allocation units before deleting them, type **y** for yes. (If you are sure the lost allocation units don't contain information you want, type **n** for no. The **chkdsk** command deletes the information, and you can skip the remaining steps in this procedure.)

   If you answer yes, **chkdsk** converts any lost file allocation units to visible files with filenames similar to FILE0001.CHK. It puts these files in your root directory. The **chkdsk** command also displays information about the disk it just checked. For information about the **chkdsk** command, see Chapter 14, "Commands."

5. Use the **type** command to examine the .CHK files. (You can also use the View File Contents command on the File menu in MS-DOS Shell.)

   For example, to examine the file FILE0001.CHK, you would type **type file0001.chk**. For information about the **type** command, see Chapter 14, "Commands."

   Sometimes a .CHK file contains information you want to keep. For example, if a text-editing program ends before you save your edits, you might find your lost edits in a recovered .CHK file.

6. Delete any .CHK files you don't want.

Helping MS-DOS Find Files Quickly

When you type a command or start a program, MS-DOS must find the executable file before it can carry out the command or start the program. If you type the full path and filename of the file, MS-DOS can find and carry out the command or run the program almost immediately. If you type only the filename, MS-DOS searches for the program file as follows:

1. MS-DOS looks for the program file in your current directory.
2. If the file is not in your current directory, MS-DOS looks for the file in the directories specified by your **path** command. It searches the directories in the order they appear in the **path** command. Typically, the **path** command is included in your AUTOEXEC.BAT file. For information about the **path** command, see Chapter 14, "Commands."

This search can take time, particularly if your path contains many directories or if your directories contain many files. The fewer directories and filenames MS-DOS must search through, the faster the response will be.

For example, if you type **myeditor** at the command prompt, MS-DOS looks for a file named MYEDITOR.COM, MYEDITOR.EXE, or MYEDITOR.BAT. It first looks in your current directory. If it doesn't find the file there, it looks in the first

directory listed in your path. MS-DOS continues searching through each directory in your path until it finds the filename or until there are no more directories to search.

If your disk has one or two directories that contain frequently used program files, you might want to list those directories first in your **path** command. For example, suppose all your MS-DOS batch (.BAT) programs are in the directory C:\BELFRY, and the programs you use most frequently are in the directory C:\PROGRAMS. An efficient **path** command might look like the following:

```
path=c:\belfry;c:\programs;c:\dos;c:\;c:\util
```

You should keep the number of files in each directory to 150 or less. This reduces the time MS-DOS spends searching.

Reorganizing Your Hard Disk to Improve Speed

Over time, as programs read from and write to your hard disk, information stored on your disk can become *fragmented.* Fragmentation occurs when a file, instead of being stored in contiguous sectors of the disk, is broken into fragments that are stored in different locations on the disk. Although fragmentation doesn't affect the validity of the information—your files are still complete when you read them into a program—it takes much longer to read files from the disk. It also takes longer for programs to write files back to the disk.

There are two ways to improve the organization of your hard disk:

- Run a disk-compaction program.

  The disk-compaction program reorganizes information on your hard disk so that pieces of information in each file are stored as close together as possible. This makes reading from and writing to your hard disk more efficient.

- Back up, reformat, and restore your hard disk.

  Reformatting your hard disk "wipes it clean." You can then restore your files in an efficient order. However, the process of backing up, reformatting, and restoring your hard disk can be very time-consuming.

Using a Disk-Compaction Program

As explained in the preceding section, a disk-compaction program reorganizes the files on your hard disk so that they are no longer fragmented. This makes it faster to read files from and write files to the hard disk. (Disk-compaction programs are sometimes called *defragmentation utilities* or *disk organizers*.)

CAUTION Run the disk-compaction program directly from MS-DOS, after quitting all other programs, including memory-resident programs.

Advantages

The following are the advantages of using a disk-compaction program:

- Makes it faster to read from and write to files on a hard disk. This, in turn, speeds up system performance.
- Can significantly decrease the time it takes for programs to start.
- Is easy to implement.

Disadvantages

The following are the disadvantages of using a disk-compaction program:

- Takes a long time to run most disk-compaction programs (from 5 minutes to several hours, depending on your system).
- Can't be run from within another program.
- Isn't included with MS-DOS. However, various disk-compaction programs are available from your software dealer.

Recommendations

- Compact your hard disk(s) regularly to help keep your system's performance from degrading because of file fragmentation.
- Compact your hard disk immediately before installing new programs on it. (This is less important if you have been compacting the disk regularly.)

- Delete any unnecessary files, and then use **chkdsk /f** before compacting your hard disk.
- Make sure you quit all other programs before running a disk-compaction program.

To compact a hard disk:

1. Delete any unnecessary files from the hard disk by using the steps outlined in "Deleting Unnecessary Files" earlier in this chapter.
2. Quit all programs.

   If you use Fastopen, SMARTDrive, or other memory-resident programs, disable the corresponding commands in your CONFIG.SYS file and restart your computer to ensure that those programs don't interfere with the disk-compaction process.
3. Use the **chkdsk /f** command to clean up lost allocation units.

   Never use **chkdsk /f** from within a program. For information about the **chkdsk /f** command, see "Using the Chkdsk Command" earlier in this chapter.
4. Run your disk-compaction program by following the manufacturer's instructions.

Reducing File Fragmentation by Reformatting Your Hard Disk

If your system is reading and writing information more slowly than usual, the files on your disk might be fragmented. If you don't have a disk-compaction program and you suspect that the files on your hard disk are badly fragmented, you might want to compact your hard disk by reformatting it. This involves backing up the files on your hard disk, reformatting the disk, and restoring the files to the reformatted disk. However, you should reformat your hard disk to reduce file fragmentation only if you suspect the information on the disk is badly fragmented. Unlike running a disk-compaction program, this process is too time-consuming to do frequently.

CAUTION You must back up all your files before reformatting your hard disk. Otherwise, your data will be lost, since the **format** command deletes all files on your hard disk.

Advantages

The following are the advantages of reformatting your hard disk:

- Makes it faster to read from and write to files on your hard disk, which speeds up system performance.
- Can significantly decrease the time it takes for programs to start.
- Doesn't require you to purchase additional software.

Disadvantages

The following are the disadvantages of reformatting your hard disk:

- Can take several hours to complete.
- Requires you to back up all the files on your hard disk.

▶ To reorganize your hard disk by reformatting it:

1. Delete any unnecessary files from the hard disk by using the steps outlined in "Deleting Unnecessary Files" earlier in this chapter.
2. Quit all programs.
3. Use the **chkdsk /f** command to clean up lost file clusters.

   Never use **chkdsk /f** from within a program. For information about the **chkdsk /f** command, see "Using the Chkdsk Command" earlier in this chapter.

   Before running **chkdsk /f**, make sure you aren't running any memory-resident programs that use the hard disk. You may need to disable memory-resident programs in your CONFIG.SYS and AUTOEXEC.BAT files and restart your system.
4. Use the **backup** command to back up the files on your hard disk. For information about the **backup** command, see Chapter 14, "Commands."
5. After all your files are backed up, use the **format** command to reformat your hard disk. For information about the **format** command, see Chapter 14, "Commands."
6. Use the **restore** command to restore your saved files onto your newly formatted hard disk. For information about the **restore** command, see Chapter 14, "Commands."

Adjusting Your Hard-Disk Interleave

A hard-disk *interleave* determines how many times the disk must revolve in order for an entire disk track to be read. The optimal interleave for a hard disk depends on the type of hard disk and disk controller you have. The interleave is particularly important if you are using the SMARTDrive program. An improper interleave can reduce disk speed by as much as 200 to 300 percent. It is a good idea to adjust your hard-disk interleave so that it is optimal for your disk type and controller. You can purchase a program that checks and adjusts your hard-disk interleave.

Advantages

The following are the advantages of adjusting your hard-disk interleave:

- Can drastically improve system speed, particularly if you're using SMARTDrive.
- Doesn't use additional memory or disk space.
- Needs to be done only once (although you might experiment to find the best interleave for your system).

Disadvantages

The following are the disadvantages of adjusting your hard-disk interleave:

- Requires you to purchase software. MS-DOS doesn't include a program for adjusting the interleave of your hard disk.
- May require that you reformat your hard disk. However, a few utility programs are capable of safely resetting the interleave without affecting data on the hard disk.

Using the Buffers Command

The **buffers** command in your CONFIG.SYS file specifies the number of buffers that MS-DOS reserves for file transfers. For information about the **buffers** command, see Chapter 11, "Customizing Your System."

The greater the number of buffers (up to about 50), the faster your system runs. However, past a certain value, increasing the number of buffers only uses more memory without increasing speed.

When optimizing your system for speed, you want to specify the greatest number of buffers that will be useful for your system. This number depends on the size of your hard disk. The following are the most effective buffer sizes for different sizes of hard disks:

Hard-disk size	Buffer size
Less than 40 MB	20
40 through 79 MB	30
80 through 119 MB	40
More than 120 MB	50

The following command specifies 40 buffers — an optimal number for a system with a 110-MB hard disk:

```
buffers=40
```

NOTE When calculating the default number of buffers, MS-DOS bases the number on how much conventional memory your system has, rather than on the size of your hard disk. The default number that MS-DOS calculates is a minimum number. The numbers in the preceding list are larger in order to increase system speed.

Using a Secondary Buffer Cache

A secondary buffer cache (sometimes called a secondary cache) can be useful if you are not using the SMARTDrive disk-caching program. MS-DOS uses the secondary cache to store the contents of files that programs are currently using. When a program requests part of a file stored on disk, MS-DOS provides the program with the information it requested. If there is a secondary cache, MS-DOS then stores the next portion of the file in the secondary cache. When the program requests the next portion of the file, MS-DOS supplies the information more quickly from the cache than it could from the hard disk.

A secondary buffer cache speeds up word-processing programs and language compilers more effectively than it does other programs. The secondary cache can also make it faster to load programs.

You specify a secondary buffer cache as part of the **buffers** command. Don't specify a secondary cache if you have installed a disk-caching program such as SMARTDrive. Normally, if you specify a secondary buffer cache, you should allocate 8 buffers to it.

The following command specifies 30 buffers and a secondary buffer cache of 8:

```
buffers=30,8
```

Using the Fastopen Program

The Fastopen program can speed up access to files and directories. It also keeps track of the locations of the files and directories you open, making subsequent access to those files much faster. The Fastopen program is particularly useful if you use programs that repeatedly open and close files, such as database programs.

Advantages

The following are the advantages of using the Fastopen program:

- Improves speed for programs that repeatedly open and close files, such as database programs and language compilers.
- Can be used on systems with only 640K of memory, since it doesn't require expanded or extended memory.
- Can use expanded memory.

Disadvantages

The following are the disadvantages of using the Fastopen program:

- Uses some conventional memory.
- Cannot use extended memory.
- Doesn't improve performance of programs that don't repeatedly open and close files. For programs other than database programs and compilers, the **buffers** command or the SMARTDrive program might improve speed more effectively than the Fastopen program.

Recommendations

The following are some basic recommendations for using the Fastopen program:

- Use the Fastopen program if you use database programs or compilers and have memory to spare.
- Give the Fastopen program access to one file for every megabyte of hard-disk space. For example, if you have a 40-MB hard disk, Fastopen can be set to work with up to 40 files at once.
- Experiment with the Fastopen program. If you don't notice any improvement in speed, the programs you are using probably don't perform a type of disk access that Fastopen can speed up. If this is the case, stop using Fastopen and free the memory it uses.
- If your system has physical expanded memory, run the Fastopen program in expanded memory by using the **fastopen** command with the **/x** switch.
- If you have an 80386 or 80486 system, try running the file FASTOPEN.EXE (the program file for Fastopen) in the upper memory area. (To do so, include a **loadhigh** command for FASTOPEN.EXE in your AUTOEXEC.BAT file.)

Starting the Fastopen Program

There are four ways to start the Fastopen program:

- Type the **fastopen** command at the command prompt.
- Include the **fastopen** command in your AUTOEXEC.BAT file.
- Add an **install** command for the file FASTOPEN.EXE to your CONFIG.SYS file.
- To run FASTOPEN.EXE in the upper memory area, include a **loadhigh** command for FASTOPEN.EXE in your AUTOEXEC.BAT file.

To start Fastopen from your CONFIG.SYS file, you would include a command like the following:

```
install=c:\dos\fastopen.exe c:=40 /x
```

This command specifies that FASTOPEN.EXE should put its cache in expanded memory, should work with files on drive C, and can work with as many as 40 files at once — the recommended number for a 40-MB hard disk.

If you have an 80386 or 80486 system, you can run FASTOPEN.EXE in the upper memory area by using the **loadhigh** command in conjunction with the **fastopen** command. (This means that you can load FASTOPEN.EXE into the upper memory area either from your AUTOEXEC.BAT file or from the command prompt, but not from your CONFIG.SYS file.) For example, the following command loads FASTOPEN.EXE into the upper memory area from either your AUTOEXEC.BAT file or the command prompt:

```
loadhigh c:\dos\fastopen.exe c:=30
```

To use the upper memory area, you must set up your CONFIG.SYS file properly. For information about how to do this, see "Running Programs in the Upper Memory Area" later in this chapter.

For more information about the **fastopen** command, see Chapter 14, "Commands."

Using the SMARTDrive Disk-Caching Program

SMARTDrive is a disk-caching program for computers that have a hard disk and extended or expanded memory. Disk-caching programs can reduce the amount of time your computer spends reading data from your hard disk.

SMARTDrive sets aside some expanded or extended memory for its own use. This area of memory is called the SMARTDrive *cache*; SMARTDrive uses it to store information read from the hard disk. When a program attempts to read that information from the hard disk, SMARTDrive supplies the information directly from its cache instead. SMARTDrive always copies new or modified information to the hard disk, so there is no danger of losing data when you turn off your computer.

You install SMARTDrive by adding a **device** command to your CONFIG.SYS file.

NOTE The version of SMARTDrive that comes with MS-DOS version 5.0 supersedes the version that comes with Microsoft Windows version 3.0. If you have both MS-DOS 5.0 and Windows 3.0, use the version of SMARTDrive that comes with MS-DOS.

Advantages

The following are the advantages of using the SMARTDrive program:

- Improves speed on all systems that have expanded or extended memory.
- Is fairly easy to adjust.

Disadvantages

The following are the disadvantages of using the SMARTDrive program:

- Uses some conventional memory.
- Requires either extended or expanded memory.

Recommendations

The following are some basic recommendations for using the SMARTDrive program:

- Use SMARTDrive if your system has a hard disk and at least 512K of extended memory or 256K of expanded memory. Many programs run much faster with SMARTDrive.
- If your system has extended memory, specify that SMARTDrive should use extended memory.
- If your system has expanded memory, specify that SMARTDrive should use expanded memory. You do this by adding the **/a** switch to the **device** command for SMARTDrive in your CONFIG.SYS file.
- If your system has both extended and expanded memory, specify that SMART-Drive should use whichever type of memory is more abundant on your system.
- Allow SMARTDrive as large a cache as possible, up to 2 MB. (The larger the cache, the more memory SMARTDrive uses.) If a program won't run because there is not enough expanded or extended memory, gradually reduce the size of the cache until the program does run properly. For information about determining the optimal size for the SMARTDrive cache, see "Specifying the Size of the SMARTDrive Cache" later in this chapter.
- If possible, compact your hard disk regularly. SMARTDrive runs best if the files on your hard disk are not fragmented.
- If you have an 80386 computer with extended memory, try running SMART-Drive in the upper memory area.
- Do not use SMARTDrive in conjunction with other disk-caching programs. In addition, Microsoft Windows/386™ version 2.*x* does not run with the version of SMARTDrive included with MS-DOS version 5.0.

Installing SMARTDrive

When you install MS-DOS, the Setup program copies the SMARTDRV.SYS file to your MS-DOS directory. To install SMARTDrive, you add a **device** command for SMARTDrive to your CONFIG.SYS file. The command specifies the following:

- The location of the SMARTDRV.SYS file.
- The size of the SMARTDrive cache.
- Whether SMARTDrive should use extended or expanded memory (optional). By default, SMARTDrive uses extended memory.

The following is a typical **device** command for SMARTDrive:

```
device=c:\dos\smartdrv.sys 1024
```

This command specifies that SMARTDRV.SYS is in the C:\DOS directory and that the cache size should be 1024K (1 MB). It runs SMARTDrive in extended memory, since SMARTDrive runs in extended memory by default.

Specifying the Size of the SMARTDrive Cache

The size of the SMARTDrive disk cache affects its efficiency. In general, the larger the cache, the less often the SMARTDrive program needs to read data from the disk. The most efficient size for the cache is about 2 MB; increasing the cache size over 2 MB probably won't increase system performance.

The SMARTDrive cache size is the first numeric parameter in the **device** command for SMARTDrive.

NOTE SMARTDrive also has an optional second numeric parameter that limits how much Microsoft Windows can reduce the cache size. This parameter is important if you use Microsoft Windows version 3.0 or later. For additional information about running SMARTDrive with Windows, see the *Microsoft Windows User's Guide*.

The following command specifies that SMARTDrive should create a 1024K cache in extended memory:

```
device=c:\dos\smartdrv.sys 1024
```

When creating its cache, SMARTDrive rounds the specified cache size down to the nearest multiple of the size of the disk track. (If your system has more than one hard disk, SMARTDrive uses the largest track size.) For example, if the track size

on your hard disk is 10K and you have set up a 256K cache, then 25 tracks can fit in the cache. In this case, SMARTDrive creates a cache that uses 250K, not 256K, of memory and stores no more than 25 tracks at a time. In general, SMARTDrive runs most efficiently with disks that have small tracks.

NOTE On most systems, the SMARTDrive program requires 15K or more of conventional memory in order to run. The amount depends on the size of the largest track on your hard disk(s) and the size of the cache you specify.

Because the optimal cache size for SMARTDrive depends on the programs you run and your system configuration, there is no single best setting. You should experiment to find the best cache size for your system.

Recommendations for Cache Size

The following are some basic recommendations for setting the size of the disk cache:

- Set the cache size to between 256K and 2048K, and set it as large as possible within this range.
- Don't set the cache size smaller than 256K. If you do, SMARTDrive probably won't be able to cache enough information to be effective.
- Avoid setting the cache size larger than 2048K, because this might not be the best use of your system's memory. Although a larger cache is faster, you get less improvement in speed as the cache size increases above 2048K. For example, increasing a 256K cache to 512K might improve your system speed by 20 percent. However, increasing a 2048K cache by the same amount, from 2048K to 2304K, might improve speed by only 2 percent.

Putting the SMARTDrive Cache in Extended or Expanded Memory

The SMARTDrive program can use either extended or expanded memory for its disk cache. By default, it uses extended memory, and you should give SMARTDrive extended memory unless your system has only expanded memory. To have SMARTDrive use expanded memory instead, add the **/a** switch to the end of the **device** command for SMARTDrive in your CONFIG.SYS file.

Putting the Cache in Extended Memory

If your computer has extended memory, you'll probably want to put the SMARTDrive cache in extended memory and give it as much memory as possible (up to 2 MB).

NOTE To put the SMARTDrive cache in extended memory, the **device** command for SMARTDrive must appear in your CONFIG.SYS file after the command for HIMEM.

The following commands install HIMEM and SMARTDrive and give SMARTDrive extended memory:

```
device=c:\dos\himem.sys
device=c:\dos\smartdrv.sys  1024
```

The second command specifies that the SMARTDRV.SYS file is in the directory C:\DOS. It also assigns 1024K (1 MB) of extended memory to SMARTDrive, so the cache size is 1024K.

Putting the Cache in Expanded Memory

If your computer has expanded memory and you can spare some for SMARTDrive, you'll want to put the SMARTDrive cache in expanded memory. (If you use other programs that need expanded memory, be sure to leave enough expanded memory for them.)

NOTE To use expanded memory, the **device** command for SMARTDrive must appear in your CONFIG.SYS file after the command that starts your system's expanded-memory manager.

The following command creates the SMARTDrive cache in expanded memory:

```
device=c:\dos500\smartdrv.sys 2048 /a
```

This command specifies that the SMARTDRV.SYS file is in the C:\DOS500 directory. It also assigns 2048K (2 MB) of expanded memory to SMARTDrive, so the cache size is 2048K.

NOTE It is not a good idea to put the SMARTDrive cache in expanded memory provided by EMM386.EXE. EMM386 uses extended memory to emulate expanded memory that other programs can then use. Although SMARTDrive can also use this emulated expanded memory for its cache, it might not speed up your system as much as if you gave SMARTDrive real physical memory.

Using the RAMDrive Memory-Disk Program

The RAMDrive memory-disk program is a memory-resident program that enables you to use part of your system memory to emulate a very fast, temporary disk drive. This memory area is called a *RAM disk* because it exists in random-access memory (RAM). RAM disks are much faster than hard disks, because your computer can read information faster from memory than from a physical disk. You can use a RAM disk just as you would a physical disk drive.

One important difference between a real disk drive and a RAM disk is that, since the RAM disk exists only in memory, information on a RAM disk is lost when you turn off or restart your computer.

Advantages

The following are the advantages of using the RAMDrive program:

- Provides you with a very fast disk drive.
- Provides additional disk space for temporary storage. This can be extremely useful on a system without a hard disk.

Disadvantages

The following are the disadvantages of using the RAMDrive program:

- Uses additional memory, which can decrease your system's speed and capacity.
- Doesn't store information permanently. When you turn off your computer, the information stored on your RAM disk is lost. Therefore, the RAM disk is best for storing temporary files or copies of programs, not for saving data files that might change.
- Re-creates the RAM disk whenever you restart your computer. Therefore, you must recopy information to your RAM disk each time you start your computer.

Recommendations

The following are some basic recommendations for using the RAMDrive program:

- Use RAMDrive only if you really need a RAM disk. In many cases, using the same amount of memory for the SMARTDrive disk-caching program will improve your system speed more than RAMDrive would.

- If your system has plenty of memory but doesn't have a hard disk drive, use RAMDrive and assign it as much memory as possible.

- If you often run programs that use many small temporary files, use RAMDrive instead of SMARTDrive, since RAMDrive might improve system speed more than SMARTDrive in this case. Then set your TEMP environment variable to the RAM disk. For information about the TEMP environment variable, see "Using the TEMP Environment Variable with RAMDrive" later in this chapter.

- If you have an 80386 or 80486 computer, load RAMDrive by using the **devicehigh** command instead of the **device** command in your CONFIG.SYS file; the **devicehigh** command loads RAMDrive into the upper memory area, thereby conserving conventional memory. For information about using the upper memory area, see "Running Programs in the Upper Memory Area" later in this chapter.

- If you run programs from your RAM disk, list your RAM disk first in your **path** command. For example, if your RAM disk is drive E, add **e:** to the beginning of the **path** command.

- If you use the EMM386 program as an expanded-memory emulator, do not put the RAM disk in expanded memory. Although RAMDrive can also use this emulated expanded memory, it won't be as efficient as it would if it were using real physical memory.

CAUTION Do not use the RAM disk to store data files. All information on your RAM disk is lost whenever you turn off or restart your computer, your system fails, or there is a power outage.

Installing RAMDrive

When you install MS-DOS, the Setup program copies the RAMDRIVE.SYS file to your DOS directory. To install RAMDrive, you add a **device** or **devicehigh** command for RAMDrive to your CONFIG.SYS file. This command specifies the following:

- The location of the RAMDRIVE.SYS file

- The amount of memory to allocate to RAMDrive

- Whether RAMDrive should use conventional, expanded, or extended memory

The following is a typical **device** command for RAMDrive:

```
device=c:\dos\ramdrive.sys 512 /e
```

This command specifies that RAMDRIVE.SYS is in the C:\DOS directory and assigns 512K of extended memory to RAMDrive. The **/e** switch specifies that RAMDrive should use extended memory.

IMPORTANT If RAMDrive is to use extended memory, your CONFIG.SYS file must contain a **device** command for the HIMEM.SYS memory manager. If RAMDrive is to use expanded memory, your CONFIG.SYS file must contain a **device** command for the expanded-memory manager that came with your memory board. The **device** command for RAMDrive must come after the one for the memory manager.

The following command specifies that the RAMDRIVE.SYS file is in the C:\DOS directory and assigns to RAMDrive 4096K of expanded memory. The **/a** switch specifies that RAMDrive should use expanded memory:

```
device=c:\dos\ramdrive.sys 4096 /a
```

Running Programs from Your RAM Disk

If you start certain programs frequently, you might want to run them from your RAM disk rather than from a physical disk. Starting a program from a RAM disk can be substantially faster than starting it from a physical disk. This method can be particularly useful if you usually start programs from a floppy disk.

▶ To run a program from your RAM disk:

1. Install RAMDrive as described in the preceding section.

   The drive letter of your RAM disk should be the letter after that of your last physical drive. For example, if your last physical disk drive is C, your RAM disk would be D. If you have three hard disk drives, named C, D, and E, your RAM disk would be F.

2. Copy the program's executable file(s) to the RAM disk.
3. Start the program from the RAM disk as you would from a physical disk drive. (Depending on the program, you might need to first make the RAM disk your current drive.)

You will need to recopy the program to the RAM disk each time you restart your computer. If you use the program frequently, you might want to add a **copy** command to your AUTOEXEC.BAT file. For example, to copy Microsoft Excel to a RAM disk named F whenever you restart your computer, you would add the following command to your AUTOEXEC.BAT file:

```
copy c:\excel\excel.exe f:\
```

Using the TEMP Environment Variable with RAMDrive

Many programs use temporary files to store data while they're running. Some of these programs store temporary files in a directory specified by the TEMP environment variable. For information about how a particular program stores temporary files, see the documentation that came with the program.

You set the TEMP variable by using the **set** command. Typically, the **set** command is used in the AUTOEXEC.BAT file. For example, the following command sets the TEMP variable to the TEMPFILE directory on drive C:

```
set temp=c:\tempfile
```

Setting the TEMP variable affects only those programs that check for the value of TEMP.

NOTE A few older programs check for the TMP environment variable instead. If you use older programs, you might want to set both the TEMP and TMP variables.

The location you specify for temporary files can affect the speed of programs that use the TEMP variable. For example, if the TEMP variable specifies a relatively slow hard disk drive, programs that store temporary files on that drive might run at less than optimal speed.

Because it's much faster to read information from memory than from a hard disk, a program that uses temporary files usually runs faster if it stores its temporary files on a RAM disk. Therefore, you should set the TEMP variable to your RAM disk. Since most programs delete their temporary files when they're finished using them, you don't need to worry about saving copies of those temporary files before turning off your computer.

You should set the TEMP variable to a subdirectory, not to a root directory. With MS-DOS, you can create only a limited number of files in the root directory of a disk, but you can create as many files as you need within a subdirectory.

Make sure that the disk to which you set the TEMP variable has enough free space for the temporary files your program creates. For more information about how the program uses temporary files, consult the documentation that came with the program.

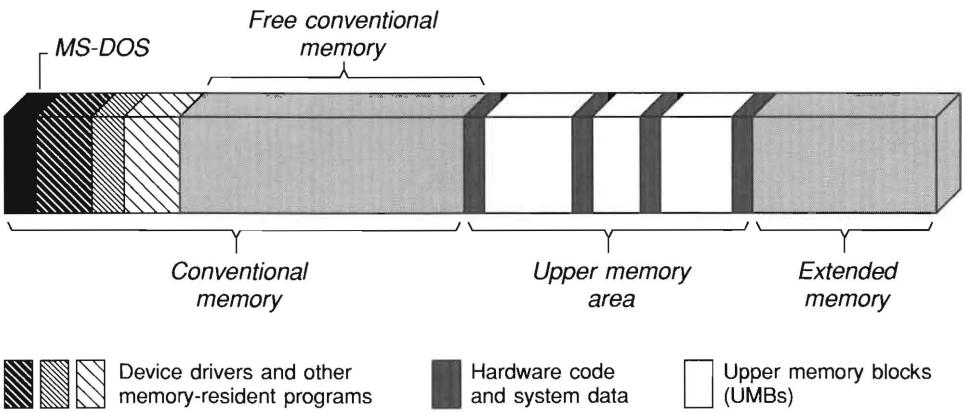
Running Programs in the Upper Memory Area

This section explains an advanced technique you can use to make more conventional memory available. Many of the procedures in this section are technically complex and should be performed only by advanced users.

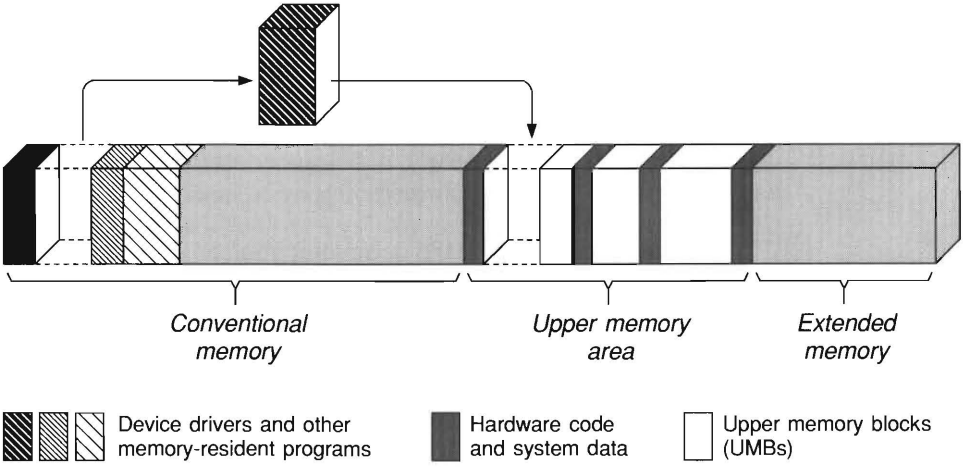
If you have an 80386 or 80486 computer, you can conserve conventional memory by running device drivers and other memory-resident programs in the upper memory area.

As explained in "Understanding Memory" earlier in this chapter, the *upper memory area* is the region of your computer's memory that is normally set aside for the system's use. On most systems, some parts of the upper memory area are left unused after all the hardware drivers have started. These areas are called *upper memory blocks* (UMBs). You can use UMBs for running installable device drivers and other memory-resident programs. This lets you move those programs out of conventional memory, thereby making more conventional memory available for running programs.

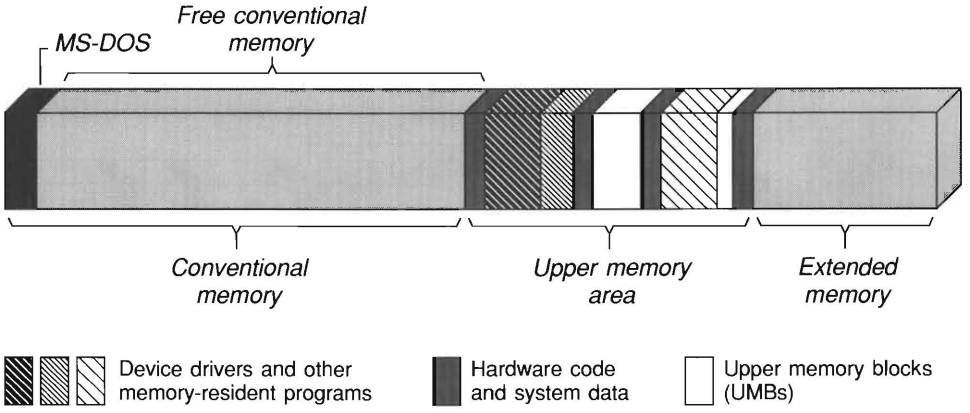
The following illustration shows the contents of a typical computer's conventional memory and upper memory area. Note that a fair amount of conventional memory is being used by device drivers and other memory-resident programs.



The next illustration shows how a device driver or other memory-resident program can be moved to the upper memory area, freeing conventional memory for use by other programs.



The next illustration shows the same computer's memory after some drivers and programs have been moved to the upper memory area. Notice that those programs are running in upper memory blocks that were previously unused. There is now much more conventional memory available for other programs that require it.



Advantages

The following are the advantages of running programs in the upper memory area:

- Makes more conventional memory available to programs.
- Enables you to use parts of the upper memory area that would otherwise go unused.

Disadvantages

The following are disadvantages of running programs in the upper memory area:

- EMM386, the MS-DOS upper-memory-area manager, runs only on 80386 and 80486 computers.

  This means that you cannot use EMM386 to gain access to the upper memory area on 8086, 8088, and 80286 computers. With these computers, you must use an upper-memory-area manager other than EMM386. Check with your computer's manufacturer for information about memory-management programs for your computer that provide access to the upper memory area.

- The EMM386 upper-memory-area manager uses about 8K of conventional memory.
- It may be difficult to determine which programs will fit into the upper memory area.
- Some programs cannot run successfully in the upper memory area.

Recommendations

The following are some basic recommendations for using the upper memory area:

- Try running programs in the upper memory area if you have an 80386 or 80486 computer.
- Before getting started, make sure MS-DOS version 5.0 is set up properly and your system is working properly. This will make it easier to eliminate the cause of any problems you encounter while running programs in the upper memory area.

- Move device drivers and other memory-resident programs to the upper memory area one at a time.
- Experiment to find out which device drivers and other programs fit into the available portions of the upper memory area.

The following is an overview of the steps you must perform in order to run programs in the upper memory area:

1. Prepare for the process of moving programs to the upper memory area.

   Not all programs work properly in the upper memory area. Therefore, it is not unusual for your system to lock up while you are moving programs to the upper memory area. There are several things you can do to make this process safer and simpler. For information about how to do this, see "Preparing to Run Programs in the Upper Memory Area" later in this chapter.
2. Set up your CONFIG.SYS file so that MS-DOS can gain access to the upper memory area. For information about how to do this, see "Setting Up Your CONFIG.SYS File for the Upper Memory Area" later in this chapter.
3. Find out how much of the upper memory area is available for running programs, and which device drivers and other memory-resident programs can fit into that available memory. For information about how to do this, see "Getting Information About the Upper Memory Area" later in this chapter.
4. One at a time, move device drivers and other memory-resident programs to the upper memory area. After changing the command that loads the driver or program into the upper memory area, restart your system and make sure the driver or program is working properly in the upper memory area. For more information, see "Moving Programs to the Upper Memory Area" later in this chapter.

   Repeat this step for each driver and other memory-resident program you want to run in the upper memory area.
5. After determining which programs run well in the upper memory area, you might want to optimize your use of the upper memory area. For information about how to do this, see "Optimizing Your Computer's Use of the Upper Memory Area" later in this chapter.

Preparing to Run Programs in the Upper Memory Area

Before you run programs in the upper memory area, you should prepare your system to do so. Some programs do not run properly in the upper memory area. Unfortunately, the only way to discover whether a program can run in the upper memory area is to try it. If you load a program into the upper memory area and the program cannot run there, your system might not restart, or it might lock up when you try to use that program.

You can make the process of running programs in the upper memory area both safer and simpler by preparing for it.

▶ Before you start running programs in the upper memory area:

1. Make sure your system's hardware and memory work properly.
2. Make sure MS-DOS is installed properly.
3. Make sure your device drivers and other memory-resident programs work properly as they are currently installed.
4. Make a system disk by using the **format /s** command.
5. Copy your CONFIG.SYS and AUTOEXEC.BAT files to the startup disk for use as backup files.

Setting Up Your CONFIG.SYS File for the Upper Memory Area

After performing the steps in the preceding section, you are ready to set up your CONFIG.SYS file so that MS-DOS can gain access to the upper memory area. MS-DOS needs both the HIMEM and the EMM386 memory manager in order to use the upper memory area.

▶ To set up your CONFIG.SYS file:

1. Make a backup copy of your CONFIG.SYS file, if you have not done so already. (The startup disk you created in the previous section should also contain a backup copy of your CONFIG.SYS file.)
2. Open your CONFIG.SYS file by using MS-DOS Editor or another text editor.
3. Make sure that the file contains a **device** command for the HIMEM memory manager. For example:

```
device=c:\dos\himem.sys
```

Normally, if your system has extended memory, the MS-DOS Setup program installs HIMEM automatically. If your CONFIG.SYS file does not contain a **device** command for HIMEM, you should add one. For information about the HIMEM memory manager, see "Using the HIMEM Extended-Memory Manager" earlier in this chapter.

4. Add a **dos=umb** command. This command specifies that MS-DOS should maintain a link between conventional memory and the upper memory area.

   If your CONFIG.SYS file contains a **dos=high** command, you can just add the **umb** switch to that command. For example:

   ```
   dos=high,umb
   ```

5. Add a **device** command for the EMM386 program. This command must contain either the **noems** or the **ram** switch. For example:

   ```
   device=c:\dos\emm386.exe noems
   ```

   If your CONFIG.SYS file already contains a **device** command for EMM386, just add the **ram** switch to the end of the command. For information about the **device** command for EMM386, see the rest of this section.

6. Make sure that the **device** commands for HIMEM and EMM386 appear before any other **device** commands. The **device** command for HIMEM should come before that for EMM386.

7. Save the changes to your CONFIG.SYS file.

8. Restart your system by pressing CTRL+ALT+DEL.

Your CONFIG.SYS file should now be set up so that you can run device drivers and programs in the upper memory area.

Installing EMM386 to Manage the Upper Memory Area

The EMM386 memory manager provides access to unused portions of an 80386 or 80486 computer's upper memory area. This enables you to run device drivers and other programs in that memory. (EMM386 can also use your system's extended memory to simulate expanded memory for use by programs. For information about this use of EMM386, see "Freeing Expanded Memory" earlier in this chapter.)

To use EMM386 as an upper-memory-area manager, add a **device** command for EMM386 to your CONFIG.SYS file. (If your CONFIG.SYS file already contains a **device** command for EMM386, just edit that command.)

The **device** command for EMM386 must come after the **device** command for HIMEM, and before any **devicehigh** commands. The command must include one of the following switches:

Switch	Description
noems	Runs EMM386 to manage the upper memory area only. Use this switch if your programs do not require expanded memory. This switch provides the maximum available amount of the upper memory area for running device drivers and programs. However, it prevents EMM386 from simulating expanded memory.
ram	Runs EMM386 to both manage the upper memory area and simulate expanded memory. Use this switch if you have any programs that require expanded memory. When started with the ram switch, EMM386 sets aside a part of the upper memory area for use as an EMS page frame. The **ram** switch provides less of the upper memory area for running device drivers and programs than does the **noems** switch. However, it allows programs to use the simulated expanded memory that EMM386 provides.

NOTE Microsoft Windows will not be able to allocate expanded memory to programs that need it if you specify the **noems** switch. If you use such programs, use the **ram** switch instead of the **noems** switch.

Sample Device Commands for EMM386

The following **device** command runs EMM386 to manage the upper memory area by using the **noems** switch:

```
device=c:\dos\emm386.exe noems
```

Because this command contains the **noems** switch, EMM386 will provide access to all available portions of the upper memory area, and will not function as an expanded-memory emulator.

The following **device** command uses EMM386 both to provide access to the upper memory area and to emulate expanded memory:

```
device=c:\dos\emm386.exe 1024 ram
```

This command starts EMM386 to provide access to the upper memory area and to use 1024K of your computer's extended memory as expanded memory. The command includes the **ram** switch rather than the **noems** switch, since the **noems** switch would prevent programs from using the simulated expanded memory that EMM386 can provide. For information about using EMM386 to emulate expanded memory, see "Freeing Expanded Memory" earlier in this chapter.

For more information about EMM386 switches, see Chapter 15, "Device Drivers."

Getting Information About the Upper Memory Area

Once you have set up your CONFIG.SYS file as explained in the preceding section, you can find out how much of the upper memory area is available. You also need to find out how much memory your device drivers and other memory-resident programs require.

▶ To get information about your system's upper memory area:

- Type the following command at the prompt:

```
mem /c | more
```

MS-DOS displays three columns of information about the programs currently using your system's memory. The following sample display shows the contents of memory on a system that has 640K of conventional memory:

```
Conventional Memory :

  Name                 Size in Decimal        Size in Hex
-------------        ---------------------    -------------
  MSDOS               23808      ( 23.3K)        5D00
  HIMEM                1184      (  1.2K)         4A0
  EMM386               9232      (  9.0K)        2410
  DISPLAY             27792      ( 27.1K)        6C90
  MOUSE               14816      ( 14.5K)        39E0
  VT52                 4192      (  4.1K)        1060
  ANSI                 4208      (  4.1K)        1070
  RAMDRIVE             1184      (  1.2K)         4A0
  SMARTDRV            22576      ( 22.0K)        5830
  COMMAND              2880      (  2.8K)         B40
  DOSKEY               4144      (  4.0K)        1030
  FREE                   64      (  0.1K)          40
  FREE                 3616      (  3.5K)         E20
  FREE                  368      (  0.4K)         170
  FREE               534864      (522.3K)       82950

Total FREE :         538912      (526.3K)
```

```
Upper Memory :

  Name                 Size in Decimal        Size in Hex
-------------        ---------------------    ------------
  SYSTEM              199640      (192.0K)       30020
  FREE                   368      (  0.4K)         170
  FREE                 71632      ( 70.0K)       117D0
  FREE                 58944      ( 57.6K)        E640

Total FREE :          130944      (127.9K)

Total bytes available to programs (Conventional+Upper):     669856    (654.2K)
Largest executable program size :                           534864    (522.3K)
Largest available upper memory block :                       71632     (70.0K)

   3145728 bytes total contiguous extended memory
         0 bytes available contiguous extended memory
    273408 bytes available XMS memory
           MS-DOS resident in High Memory Area
```

The contents of the display are as follows:

- The Conventional Memory section contains information about the programs and device drivers loaded in conventional memory. The Upper Memory section contains information about the programs and device drivers running in the upper memory area.
- The Name column shows the name of each program or device driver. Available areas of memory are listed as "FREE."
- The Size in Decimal column shows the number of bytes of memory that each program module is using. (The number in parentheses is the same number in kilobytes.)
- The Size in Hex column shows the hexadecimal equivalent of the number shown in the Size in Decimal column.

You will use the information in the Name and Size in Decimal columns during the procedure in the following section.

NOTE If you are running Microsoft Windows 3.0 or later in 386 enhanced mode, the **mem** command does not report the contents of the upper memory area.

Moving Programs to the Upper Memory Area

After following the instructions in the preceding sections, you can begin moving programs from conventional memory to the upper memory area. In order to run a program in the upper memory area, that program must fit into the largest available UMB.

NOTE You should perform this procedure for one device driver or program at a time, so that you can tell whether each runs successfully in the upper memory area.

▶ To move a device driver or other program to the upper memory area:

1. Type the following command at the MS-DOS prompt:

   ```
   mem /c ¦ more
   ```

2. Note the size given in the line "Largest available upper memory block," near the bottom of the display.

3. Look in the Conventional Memory section for the largest device driver or program that will fit into that UMB.

   Ensure that the size of the device driver or program is equal to or smaller than the size of the largest available UMB.

   Note that certain items in conventional memory, such as MS-DOS system data, cannot be moved to the upper memory area. Also note that the HIMEM and EMM386 memory managers cannot run in the upper memory area.

4. After deciding which device driver or program you want to run in the upper memory area, change that program's startup command so that the program will be loaded into the upper memory area the next time you restart your system.

   For a device driver, edit your CONFIG.SYS file and change the driver's **device** command to a **devicehigh** command. For information about loading device drivers into the upper memory area, see "Running Device Drivers in the Upper Memory Area" later in this section.

   For a memory-resident program, insert the **loadhigh** command at the beginning of the command that starts the program. (Most memory-resident programs are started by a command in your AUTOEXEC.BAT file.) For information about using **loadhigh** with memory-resident programs, see "Running Memory-Resident Programs in the Upper Memory Area" later in this section.

5. Save your CONFIG.SYS or AUTOEXEC.BAT file.
6. Restart your system by pressing CTRL+ALT+DEL.
7. Type the following command at the prompt:

   ```
   mem /c | more
   ```

   Make sure that the driver or other program is running in the upper memory area. If the program appears in the Conventional Memory section, the program is still running in conventional memory. This is probably because the program did not fit into the largest available UMB. (Some device drivers and programs require more memory when they are loaded than when they are running. Such programs might not fit into a UMB even if the size of the program file for that program is smaller than the largest UMB.)
8. Test the program, and your system in general, to make sure everything works properly. If your system locks up during startup or while you are using that program, it is likely that the program cannot run successfully in the upper memory area. (If this happens with a device driver, see "Running Device Drivers in the Upper Memory Area" later in this section, for possible ways to fix this problem.)

If the program is running successfully in the upper memory area, repeat this procedure for the next program or device driver you want to run in the upper memory area.

The following sections explain how to use the **devicehigh** and **loadhigh** commands, respectively.

Running Device Drivers in the Upper Memory Area

Device drivers are programs that provide access to your computer's hardware. For example, HIMEM.SYS is a device driver that provides access to extended memory. All device drivers use some conventional memory; this makes less conventional memory available for programs. You can conserve conventional memory by running certain device drivers in the upper memory area.

Normally, you run a device driver by using the **device** command in your CONFIG.SYS file; this command runs the device driver in conventional memory. However, sometimes you might want to conserve conventional memory by running the device driver in the upper memory area instead. To run a device driver in the upper memory area, you use the **devicehigh** command in your CONFIG.SYS file.

Recommendations for Using the Upper Memory Area for Device Drivers

The following are some basic recommendations for running device drivers in the upper memory area:

- If you use the RAMDrive memory-disk program, run it in the upper memory area.
- If you use a console driver, run it in the upper memory area.
- If you use other device drivers, try running them in the upper memory area. Not all device drivers can run successfully in the upper memory area. In general, if the device driver is unable to run in the upper memory area, it will not start, and it might cause your system to lock up. (If this happens, insert a startup disk in drive A; restart your system by pressing CTRL+ALT+DEL or by turning the power off and back on again.)
- Do not run HIMEM and EMM386 in the upper memory area. Trying to do so will not cause any problems; however, neither will it work. HIMEM and EMM386 must be running in conventional memory in order to provide the upper memory area for other drivers and memory-resident programs.

▶ To run a device driver in the upper memory area:

- In your CONFIG.SYS file, change the **device** command to a **devicehigh** command for that device driver.

The **devicehigh** command is similar to the **device** command, except that it loads the specified driver into the upper memory area. The following **devicehigh** command runs RAMDrive in the upper memory area. The **/a** switch specifies that RAMDrive should use 512K of expanded memory for its RAM disk:

```
devicehigh=c:\dos\ramdrive.sys 512 /a
```

When MS-DOS encounters the **devicehigh** command in your CONFIG.SYS file, it attempts to load the specified device driver into the upper memory area. If there is not enough of the upper memory area available, MS-DOS loads the driver into conventional memory instead. For more information about the **devicehigh** command, see Chapter 14, "Commands."

Running Device Drivers that Allocate Additional Memory After Starting

Some device drivers allocate additional memory after starting. Such device drivers may not be able to run properly in the upper memory area if you start them by using the **devicehigh** command.

When you start a driver by using **devicehigh**, MS-DOS places the driver in the largest UMB that will accommodate the driver's executable file. However, if the driver tries to allocate additional memory after starting, it will probably be unable to do so, and it might cause your system to lock up. To avoid this, use the **devicehigh** command with the **size=** parameter. This form of the **devicehigh** command enables you to specify how much of the upper memory area the driver needs.

To find out how much memory the driver needs, type the following command at the command prompt:

```
mem /c ¦ more
```

Look in the Conventional Memory section for the device driver. MS-DOS displays the hexadecimal size of that driver in the Size in Hex column. You specify that size in the **devicehigh** command.

For example, the following **devicehigh** command runs the MOUSE.SYS driver in the upper memory area and specifies that the driver might need an upper memory block as large as 39E0h bytes:

```
devicehigh size=39E0 C:\drivers\mouse.sys
```

The **size=** parameter takes effect only if necessary.

Running Memory-Resident Programs in the Upper Memory Area

Most memory-resident programs use some conventional memory; this makes less conventional memory available for other programs. You can conserve conventional memory by running certain memory-resident programs in the upper memory area. A typical use of the upper memory area would be to run the Doskey program.

To run a memory-resident program in the upper memory area, you use the **loadhigh** command. You can use this command either in your AUTOEXEC.BAT file or at the command prompt.

Recommendations for Using the Upper Memory Area for Memory-Resident Programs

The following are some basic recommendations for running memory-resident programs in the upper memory area:

- If you use the Doskey, Share, Nlsfunc, Keyb, Graphics, Mirror, or Fastopen program, run the associated executable file in the upper memory area.
- If your AUTOEXEC.BAT file starts a memory-resident program, try running that program in the upper memory area.

▶ To run a memory-resident program in the upper memory area:

1. Locate the command that starts the program. For most memory-resident programs, this command will be in your AUTOEXEC.BAT file.
2. Insert the **loadhigh** command at the beginning of the command that starts the program. (The **loadhigh** command can be abbreviated to **lh**.)

   For example, suppose the Doskey program is started by the following command in your AUTOEXEC.BAT file:

   ```
   c:\dos\doskey.com
   ```

   To run DOSKEY.EXE in the upper memory area, insert the **loadhigh** command so that the line appears as follows:

   ```
   loadhigh c:\dos\doskey.com
   ```

When MS-DOS encounters the **loadhigh** command, it attempts to load the specified program into the upper memory area. If the program does not fit in one of the available upper memory blocks, MS-DOS loads the program into conventional memory instead. If the program cannot run properly in the upper memory area, it might terminate or cause your system to lock up. If either of these situations occurs, you should run the program in conventional memory.

For more information about the **loadhigh** command, see Chapter 14, "Commands."

Optimizing Your Computer's Use of the Upper Memory Area

When you load a program or driver into the upper memory area, MS-DOS uses the largest remaining UMB, even if that program would fit into a smaller UMB. Therefore, the order in which you load programs into the upper memory area is important. It will probably take some experimentation to figure out the most efficient way to use the available parts of the upper memory area.

Recommendations

The following are some basic recommendations for optimizing your computer's use of the upper memory area:

- In general, load device drivers and programs in order of size, from largest to smallest.
- Experiment with different combinations and orders of programs. The optimal order will depend on the sizes of the programs you are loading and the sizes of the available UMBs.

Troubleshooting the Upper Memory Area

Some device drivers and programs cannot run in the upper memory area. This section describes problems you might encounter and some possible solutions.

You Receive an Error Message

If you see an error message for a device driver or program that you are attempting to run in the upper memory area, remove the **devicehigh** or **loadhigh** command for that device driver or program. In the case of a device driver, edit your CONFIG.SYS file by changing the **devicehigh** command to **device**. In the case of a program you started by using a **loadhigh** command in your AUTOEXEC.BAT file, open the file, remove the **loadhigh** command for that program, and save your changes. After you edit the appropriate file, restart your computer to assure that the problem is fixed. MS-DOS will load the device driver or program into conventional memory.

Your Computer Locks Up

If your computer locks up during or after MS-DOS attempts to use the upper memory area for device drivers or programs, note any error messages that are displayed. Use this information to try to identify the device driver or program that is causing the problem. If you can determine the problem, remove the **devicehigh** or **loadhigh** command for that device driver or program and restart your computer.

If you cannot determine which device driver or program is causing the problem, you might be able to find out by testing each one. Before you do this, create a system disk by using the **format /s** command, if you haven't already done so.

NOTE A device driver that expands beyond the limits of its UMB could also cause your system to lock up. For more information, see the following section.

▶ To determine which program is not working in the upper memory area:

1. Insert your system disk in drive A, and restart your computer.
2. Remove the **devicehigh** or **loadhigh** command for one of the device drivers or programs you were attempting to run in the upper memory area.

   In the case of a device driver, edit your CONFIG.SYS file by changing the **devicehigh** command to **device**. In the case of a program you started by using a **loadhigh** command in your AUTOEXEC.BAT file, open the file and remove the **loadhigh** command for that program. Save the file when you are done.
3. Remove your system disk from drive A, and restart your computer.

   If your computer starts successfully, you have found the problem.
4. Repeat this procedure until your computer starts successfully.

   Each time, remember to move just one device driver or program to conventional memory. That way, when you determine which one was causing the problem, you can reinstate the **devicehigh** and **loadhigh** commands for the other device drivers and programs.

A Device Driver Small Enough for a UMB Does Not Load There

Some device drivers do not fit into a UMB even though the file size seems small enough to fit into the largest available UMB. The reason is that these device drivers expand while they are being loaded into memory. Therefore, they need more memory than is indicated by the size of the device-driver file.

In most cases, MS-DOS simply loads such a device driver into conventional memory. It is possible, however, that your system will lock up if such a device driver is loaded into a UMB.

To determine how much memory such a device driver actually needs, use the **mem /c** command. Load the device driver into conventional memory, and then type **mem /c** at the command prompt. Locate the device driver in the Conventional Memory section of the display, and note the number in the Size in Hex column. This is the amount of memory the device driver needs. You can provide the driver with a large enough UMB by specifying that number in the **size=** parameter of the **devicehigh** command. For more information about this form of the **devicehigh** command, see Chapter 14, "Commands."

A Program Becomes Erratic When Loaded into the Upper Memory Area

Some programs do not run properly in the upper memory area. These include:

- Programs designed with the assumption that there will always be memory above them. Such programs work in conventional memory because there is always at least the upper memory area above them. However, when they run in the upper memory area, this might not always be the case.
- Programs that do not correctly recognize addresses in the upper memory area.

If a program performs erratically in the upper memory area, you should run it in conventional memory. If you started the program by using a **loadhigh** command in your AUTOEXEC.BAT file, open the file and remove the **loadhigh** command for that program. Save the file when you are done, and restart your computer.

Conflict in the Upper Memory Area

Some hardware programs might attempt to use the upper memory area after EMM386 has determined this memory is available for running device drivers and programs. To avoid this conflict, you can use the **x** switch when you load EMM386. This switch prevents EMM386 from allocating a specified range of the upper memory area for its use.

For example, to prevent EMM386 from using the addresses D800h through DFFFh for UMBs, you would include the following command in your CONFIG.SYS file:

```
device=c:\dos\emm386.exe noems x=d800-dfff
```

For more information about the **x** switch, see Chapter 15, "Device Drivers."

Nothing Is Running in the Upper Memory Area

If you think your computer is set up to run device drivers and programs in the upper memory area, but nothing appears in the Upper Memory section when you use the **mem /c** command, check each item in the following list.

NOTE If you are running Microsoft Windows version 3.0 or later in 386 enhanced mode, the **mem** command does not report the contents of the upper memory area.

- Make sure your CONFIG.SYS file contains the **dos=umb** command.
- Make sure the **device** command for EMM386 in your CONFIG.SYS file contains the **noems** or **ram** switch.
- Make sure your CONFIG.SYS file contains a **devicehigh** command for each device driver you want to run in the upper memory area.
- Make sure your AUTOEXEC.BAT file contains the **loadhigh** command before the name of each program that you want to run in the upper memory area.
- Make sure the **device** command for HIMEM appears before the **device** command for EMM386 in your CONFIG.SYS file.
- Make sure the **device** command for EMM386 appears before any **devicehigh** command in your CONFIG.SYS file.

Optimization Summary

The following table summarizes the methods you can use to free memory.

Method	When to use it	Memory it frees	Memory it uses
Make sure HIMEM is installed.	If your system has extended memory.	Makes extended memory available; also, frees conventional memory by allowing you to use extended memory for running certain programs.	A small amount of conventional memory.
Run MS-DOS in extended memory.	If your system has extended memory.	Conventional memory.	The portion of extended memory known as the high memory area (HMA).
Use EMM386 as an expanded-memory emulator.	If you have an 80386 or 80486 computer with extended memory, and your programs need expanded memory.	Expanded memory for use by programs (even if your system has no expanded memory).	Extended memory, and a small amount of conventional memory.
Simplify your CONFIG.SYS and AUTOEXEC.BAT files so that they do not start unnecessary memory-resident programs or utilities.	If you need to free memory.	Conventional, extended, or expanded memory, depending on the programs you remove.	Depends on the device drivers you remove. Removing a memory manager makes that type of memory unavailable; for example, removing the EMM386 upper-memory-area manager makes the upper memory area unavailable to programs.
Run device drivers such as RAMDrive, and programs such as Fastopen, in the upper memory area.	If you have an 80386 or 80486 computer.	Conventional memory.	The upper memory area, which is not normally needed by programs.

The following table summarizes the methods you can use to speed up your system without using additional memory.

Method	When to use it	What it speeds up
Run the **chkdsk /f** command.	If you suspect that lost allocation units are taking up disk space.	All programs, to some extent.
Help MS-DOS to find files quickly; for example, place frequently used directories at the beginning of the **path** command.	If your system takes a long time to respond when you type a command.	Starting programs.
Reformat or compact your hard disk.	Periodically, or if the information on your hard disk becomes fragmented.	All programs, to some extent. Can improve program startup time.
Adjust your hard-disk interleave.	If you're using SMARTDrive and your hard disk still processes information slowly.	All programs.

The following table lists the MS-DOS programs you can use to speed up your system.

Method	When to use it	What it speeds up
Use the **buffers** command to specify a secondary buffer cache.	If you use programs that a secondary cache can speed up. Don't set **buffers** to more than 20 if you are using SMARTDrive.	Compilers and other programs that read files in small sections.
Use the Fastopen program.	If you use databases or compilers, or if your system has only 640K of memory.	Mainly database managers and compilers; also speeds up other programs, though to a lesser extent.
Install SMARTDrive.	If your system has a hard disk and extended or expanded memory that isn't needed by programs. Don't use in conjunction with a secondary buffer cache.	All programs.
Install RAMDrive.	If your system has extended or expanded memory and you use programs that RAMDrive can speed up.	Programs that use temporary files, or programs that you run often.

Chapter 13

Customizing for International Use

13

You can change the national language and the country-related settings that MS-DOS uses. When MS-DOS was set up, the commands needed to convert your system to a language other than United States English may have been added to your CONFIG.SYS and AUTOEXEC.BAT files.

MS-DOS version 5.0 can use language conventions, keyboards, and character sets for the following countries, regions, or languages:

Belgium	Germany	Portugal
Brazil	Hungary	Spain
Canadian-French	International English	Sweden
Czechoslovakia (Czech)	Italy	Switzerland (French)
Czechoslovakia (Slovak)	Latin America	Switzerland (German)
Denmark	Netherlands	United Kingdom
Finland	Norway	United States
France	Poland	Yugoslavia

You can change country-related settings on your system, as follows:

- You can change the country-specific conventions for displaying dates, times, currency, character sort order, and filename characters. You do this by using the **country** command in your CONFIG.SYS file.
- You can change the characters and arrangement of your keyboard to fit the standard keyboard of another country. You do this by using the **keyb** command, which starts the Keyb program.
- You can change the character set (code page) so that you can type, display, and print characters from other languages.

MS-DOS can use up to 256 different characters when displaying, printing, and working with text. The exact set of characters used at one time is known as the *code page*. MS-DOS provides code pages for countries, languages, and regions.

By default, MS-DOS uses the code page that comes with your system. This code page is called the *hardware code page*. Your system has a hardware code page for your keyboard and your screen. Your printer may also have a hardware code page.

If all the characters you need are in your hardware code page, you can use the **country** and **keyb** commands to change to another language. If you want to use a language with characters not included in your hardware code page, you must use a *prepared code page*. Prepared code pages are alternate sets of 256 characters that are stored in code-page information (.CPI) files.

For each of the countries supported by MS-DOS, there are two code pages that you can use and switch between: a default code page and an alternate code page. If you set up a prepared code page, you must use the code-page numbers that are valid for your country settings.

The procedures in this chapter describe how to change your country settings. When you use these procedures, you need to specify a country code, keyboard code, and code-page number for the new country. Refer to the following table for the values you need to specify:

Country/Region or Language	Country Code	Keyboard Code	Default Code Page	Alternate Code Page
Belgium	032	be	850	437
Brazil	055	br	850	437
Canadian-French	002	cf	863	850
Czechoslovakia (Czech)	042	cz	852	850
Czechoslovakia (Slovak)	042	sl	852	850
Denmark	045	dk	850	865
Finland	358	su	850	437
France	033	fr	850	437
Germany	049	gr	850	437
Hungary	036	hu	852	850
International English	061		437	850
Italy	039	it	850	437
Latin America	003	la	850	437
Netherlands	031	nl	850	437
Norway	047	no	850	865

Country/Region or Language	Country Code	Keyboard Code	Default Code Page	Alternate Code Page
Poland	048	pl	852	850
Portugal	351	po	850	860
Spain	034	sp	850	437
Sweden	046	sv	850	437
Switzerland (French)	041	sf	850	437
Switzerland (German)	041	sg	850	437
United Kingdom	044	uk	437	850
United States	001	us	437	850
Yugoslavia	038	yu	852	850

Changing Conventions and Keyboards

When you change countries, you must always change the country settings and the keyboard arrangement. To change the MS-DOS country-specific conventions, you include a **country** command in your CONFIG.SYS file. To change the keyboard arrangement, you use a **keyb** command to start the Keyb program. You can use the **country** and **keyb** commands regardless of whether you have loaded any code pages.

Changing the Date and Time Format

In Brief

To set the country-specific conventions, include a **country** command in your CONFIG.SYS file. For example, if your COUNTRY.SYS file is in the C:\DOS directory, the following command sets the conventions of Finland (country code 358):

```
country=358,,c:\dos\country.sys
```

Notice that there are two commas between the country code and the path of COUNTRY.SYS.

By default, MS-DOS uses United States conventions for the following items:

- The date and time display
- The symbol used for currency

- The sort order used when alphabetizing files
- The characters used in filenames and directory names

To change the country-specific conventions, you include a **country** command in your CONFIG.SYS file. The conventions for each country are stored in the COUNTRY.SYS file. When you change conventions, MS-DOS uses the information in the COUNTRY.SYS file rather than the United States conventions.

When you use a **country** command, you must include a three-digit *country code* to specify which conventions you want to use. For example, the country code for Spain is 034. Usually, the country code is the same as the international long-distance telephone code.

If you used the MS-DOS 5.0 Setup program, your COUNTRY.SYS file is in the C:\DOS directory. If the Setup program installs a **country** command in your CONFIG.SYS file, the command points to your COUNTRY.SYS file. However, if no path to COUNTRY.SYS is specified, MS-DOS looks for the file in the root directory of your startup disk. If COUNTRY.SYS is in a directory other than the root directory, you must include its path in the **country** command. For example, suppose you want to use the conventions of Italy (country code 039). If your COUNTRY.SYS file is in the root directory of your startup disk, you would add the following command to your CONFIG.SYS file:

```
country=039
```

If your COUNTRY.SYS file is in the C:\DOS directory, you would add the following command to your CONFIG.SYS file:

```
country=039,,c:\dos\country.sys
```

If you are using the default code page, this form of the command must include two commas between the country code and the path of COUNTRY.SYS.

When you change the country setting in your CONFIG.SYS file, the default code page of the new country is used unless you specify otherwise. If you want to use the alternate code page, you can include the code-page number in the **country** command. For example, suppose you want to use the conventions of Italy. To use code page 437 instead of the default 850, you would add the following **country** command to your CONFIG.SYS file:

```
country=039,437,c:\dos\country.sys
```

If you want to use a code page (default or alternate) for a new country, you may need to set up prepared code pages for your devices. For more information, see "Using Code Pages" later in this chapter.

Changing the Keyboard

In Brief

To change the characters and their arrangement on your keyboard, use the **keyb** command. For example, if the KEYBOARD.SYS file is in the C:\DOS directory, the following command changes your keyboard layout to the characters and arrangement of a Swedish keyboard (keyboard code sv):

```
keyb sv,,c:\dos\keyboard.sys
```

Notice that there are two commas between the keyboard code and the path of KEYBOARD.SYS.

When you change from United States English to another language, you change the characters and their arrangement on your keyboard by using the **keyb** command to start the Keyb program. For example, to change from United States English to Latin American Spanish, you add four characters to your keyboard (one letter and three symbols). Many of the characters on the keyboard must be rearranged to make room for the extra characters. The Keyb program works with the IBM PC/XT, IBM PC/AT, IBM PS/2, and IBM PC-compatible keyboards.

There are three ways to start the Keyb program:

- Include the **keyb** command in your AUTOEXEC.BAT file so that it is carried out each time you start your system. For information about the AUTOEXEC.BAT file, see Chapter 11, "Customizing Your System."
- Type the **keyb** command at the command prompt.
- Use an **install** command in your CONFIG.SYS file so the command is carried out when MS-DOS reads the CONFIG.SYS file.

Regardless of how you start the Keyb program, you can always change keyboards by typing a **keyb** command at the command prompt.

If you change keyboards, you can switch back to a United States keyboard by pressing CTRL+ALT+F1. To return to the keyboard you were using, press CTRL+ALT+F2.

For more information about using keyboards, see the Appendix.

Typing the Keyb Command at the Command Prompt

When you type the **keyb** command at the command prompt or use it in a batch program, you include a two-letter keyboard code to specify which keyboard you want to use. MS-DOS looks in the KEYBOARD.SYS file to find out how to rearrange your keyboard for the country you specified.

The MS-DOS 5.0 Setup program places the KEYBOARD.SYS file in your C:\DOS directory. The **keyb** command may also find it in the current directory or root directory of your startup disk. If KEYBOARD.SYS is not in any of these directories, you must type its path in the **keyb** command. For example, suppose you want to use an Italian keyboard. Assume also that your *active code page* is 850. If your KEYBOARD.SYS file is in the root directory of your startup disk, you would type the following command:

```
keyb it
```

However, if your KEYBOARD.SYS file is in the C:\INTL directory, you must type the following command:

```
keyb it,,c:\intl\keyboard.sys
```

This form of the command must include two commas between the keyboard code and the path of KEYBOARD.SYS.

You can use code-page numbers in the **keyb** command to select the alternate code page associated with a country. For example, if you want to use code page 437 (the alternate code page for the Italian keyboard) and your KEYBOARD.SYS file is in the C:\DOS directory, you would type the following command:

```
keyb it,437,c:\dos\keyboard.sys
```

You can specify any keyboard with the **keyb** command. However, if the active code page is for a language other than the keyboard you specify, some of the characters on the keyboard may not be available. For more information about code pages, see the following section.

Using the Keyb Command in Your CONFIG.SYS File

Besides using a **keyb** command in your AUTOEXEC.BAT file or typing **keyb** at the command prompt, you can include an **install** command in your CONFIG.SYS file to load KEYB.COM. In this case, your country's default code page must be the same as the hardware code page. When loading the Keyb program from your CONFIG.SYS file, you use the same parameters you would use at the command prompt.

For example, if your KEYB.COM and KEYBOARD.SYS files are in the C:\DOS directory, you can change to an Italian keyboard by including the following command in your CONFIG.SYS file:

```
install=c:\dos\keyb.com it,,c:\dos\keyboard.sys
```

Using Code Pages

By default, MS-DOS uses the character set in the hardware code pages built into your keyboard, screen, and printer. If you use a language with characters not included in your hardware code pages, you need to install *prepared code pages.*

NOTE Monochrome and CGA monitors and many printers cannot use prepared code pages. See the documentation for your hardware device to determine whether prepared code pages are supported.

MS-DOS has six prepared code pages that you can use in addition to or instead of the hardware code pages built into your devices. Each prepared code page has the same set of standard ASCII characters—that is, the first 128 characters of each set are the same. However, each code page has a different set of national language characters. For example, the Portuguese code page has the same ASCII characters as the other code pages, but also has characters specific to Portuguese.

The following table describes the six MS-DOS prepared code pages:

Type	Number	Description
Canadian-French	863	Contains characters for English and Canadian-French
Multilingual (Latin I)	850	Contains characters for most of the languages, using the Latin alphabet, which MS-DOS supports
Nordic	850	Contains all characters for English, Norwegian, and Danish
Portuguese	850	Contains characters for English and Portuguese
United States	437	Contains characters for English and most other European languages
Slavic (Latin II)	852	Contains characters for the Slavic languages, using the Latin alphabet, which MS-DOS supports

Prepared code pages are stored in code-page information (.CPI) files. Before you use a code page, you must load it into memory. There may be more than one code page in memory, but only one code page can be active. If you don't install a prepared code page, MS-DOS uses your hardware code page. If you install one or more prepared code pages, you can switch between hardware code pages and any of the prepared code pages.

See the Appendix for tables listing the characters contained in each prepared code page.

Setting Up a Prepared Code Page

When you set up your system to use a prepared code page, be sure that the code page you specify is compatible with your country settings. For more information, see the table at the beginning of this chapter.

To use a prepared code page instead of your hardware code page, you must do the following:

- Prepare your screen, keyboard, and printer for code pages by installing device drivers. You do this by including **device** commands in your CONFIG.SYS file.
- Load the national language support program by using the Nlsfunc program. You can type **nlsfunc** at the command prompt or put the command in your AUTOEXEC.BAT file. However, you only need to load the national language support if you want to switch device code pages later by using the **chcp** command.

- Load into memory the code page(s) you want to use. You do this by using the **mode** command, either by typing **mode** at the command prompt or putting the command into your AUTOEXEC.BAT file.
- Make the prepared code page active by using the **chcp** command or the **mode** command.

These procedures are described in the following sections.

Preparing Your Keyboard and Screen for Code Pages

In Brief

To prepare your keyboard and screen to use one or more prepared code pages, include a **device** command in your CONFIG.SYS file. For example, if the DISPLAY.SYS file is in the C:\DOS directory, the following command installs the DISPLAY.SYS driver for an EGA or VGA monitor that has the 437 hardware code page:

```
device=c:\dos\display.sys con=(ega,437,1)
```

This command reserves space for one prepared code page, which you must load by using the **mode** command. See "Loading a Code Page into Memory" later in this chapter.

Every system has a hardware code page that determines which characters can be typed and displayed. If you only need the code page that your system already uses, you do not have to install additional code pages. If you want to use characters that are not contained in your hardware code page, you must include a **device** command in your CONFIG.SYS file to reserve space for one or more additional code pages.

MS-DOS has an installable device driver called DISPLAY.SYS that enables you to use prepared code pages with an EGA, VGA, or LCD monitor. Monochrome and CGA monitors can use only their own hardware code page (usually 437). If you have an EGA or VGA monitor, you can use up to six prepared code pages. If you have an LCD monitor, you can use only one prepared code page. For information about installable device drivers, see Chapter 15, "Device Drivers."

You can install the DISPLAY.SYS driver by including a **device** command in your CONFIG.SYS file. With this command, you specify the following:

- The kind of monitor you have. You can choose EGA or LCD. You can use EGA for both EGA and VGA monitors. If you omit this parameter, MS-DOS checks your hardware to find out what kind of monitor you have.
- The hardware code page your system uses. You must specify the name of the hardware code page if you want to make it active again after you have switched to a different code page. The most common value for this parameter is 437, the United States code page.
- The number of prepared code pages you want to use. For a VGA or EGA monitor, this number can be 1 through 6. For an LCD monitor, it must be 1. The default is 1.
- The number of fonts that are supported for each code page. The default is 2. To specify subfont support, add the following to the DISPLAY.SYS line in your CONFIG.SYS file (where *n* equals the number of code pages and *m* equals the number of fonts per code page):

```
display.sys con=(ega,437,(n,m))
```

NOTE If you install both DISPLAY.SYS and a third-party console driver (such as VT52.SYS), the third-party console driver must be installed first. Otherwise, the third-party console driver may disable DISPLAY.SYS.

For example, suppose the DISPLAY.SYS file is in the C:\DOS directory, and your VGA monitor has a 437 hardware code page. To add a code page, you would include the following command in your CONFIG.SYS file:

```
device=c:\dos\display.sys con=(ega,437,1)
```

All the parameters are within a set of parentheses, and each is separated by a comma. In this example, EGA is the type of monitor you have. Notice that you use EGA even though you have a VGA monitor. The hardware code page used with your monitor is 437. Because you specified a hardware code page, you can switch back to it later. The 1 is the number of prepared code pages you want to use. You don't specify which prepared code page you want because this command doesn't load the code page, it simply reserves space for it.

Preparing Your Printer for Code Pages

In Brief

You can prepare your printer to use one or more prepared code pages by including a **device** command in your CONFIG.SYS file. For example, if the PRINTER.SYS file is in the C:\DOS directory, the following command installs the PRINTER.SYS driver for an IBM Quietwriter III 5202 printer. In this case, the printer has a 437 hardware code page and is attached to the LPT1 port:

```
device=c:\dos\printer.sys lpt1=(5202,437,1)
```

This command reserves space for one prepared code page, which you must load by using the **mode** command. See "Loading a Code Page into Memory" later in this chapter.

MS-DOS has an installable device driver called PRINTER.SYS that enables you to use prepared code pages with certain types of printers. If you have one of the following printers attached to LPT1 (PRN), LPT2, or LPT3, you can use a prepared code page to change its character set:

- The IBM Proprinters II and III Model 4201 and IBM Proprinters II and III XL Model 4202 work with code pages stored in the 4201.CPI file.
- The IBM Proprinter X24E Model 4207 and the IBM Proprinter XL24E Model 4208 and compatibles work with code pages stored in the 4208.CPI file.
- The IBM Quietwriter III Printer Model 5202 and compatibles work with code pages stored in the 5202.CPI file.

Many printers have their own installable device drivers that override the MS-DOS driver. See your printer documentation for information about changing the code page. For more information about installable device drivers, see Chapter 15, "Device Drivers."

You can install the PRINTER.SYS driver by including a **device** command in your CONFIG.SYS file. When you use this command, you specify the following:

- The type of printer you have. You can choose from 4201, 4208, or 5202 (the printers that MS-DOS supports).
- The hardware code page your printer uses. You must specify the name of the hardware code page if you want to make it active again after you have switched to a different code page. The most common value for this parameter is 437, the United States code page. To find out which hardware code page your printer has, see your printer documentation.
- The number of prepared code pages you want to use. The maximum number of prepared code pages depends on the type of printer you have.

For example, suppose the PRINTER.SYS file is in the C:\DOS directory, your 4201 printer has the United States hardware code page (437), and you want to use one prepared code page. You would include the following command in your CONFIG.SYS file:

```
device=c:\dos\printer.sys lpt1=(4201,437,1)
```

All the parameters are within a set of parentheses, and each is separated by a comma. The 4201 is the type of printer you have (IBM Proprinters II and III Model 4201 or IBM Proprinters II and III XL Model 4202). The 437 is the hardware code page of your printer. Because you specified a hardware code page, you can switch back to it later. The 1 is the number of prepared code pages you want to use. You don't specify which prepared code page you want because this command doesn't load the code page, it simply reserves space for it.

Suppose you don't know which hardware code page your printer has, or you want to use a prepared code page and you don't need to return to the hardware code page. In this case, you can omit the hardware code page from your **device** command (but you must retain the commas), as in the following example:

```
device=c:\dos\printer.sys lpt1=(4201,,1)
```

Loading National Language Support for Code Pages

In Brief

Before MS-DOS can recognize and switch between prepared code pages for all devices at the same time, you must load the Nlsfunc program into memory. You use the following command:

```
nlsfunc
```

If you want to change the country code or default code page without restarting your system or if you want to change the code page for all devices at the same time, you must load the Nlsfunc (national language support function) program into memory. However, if you only want to change or use code pages for a single device, you do not need to use the Nlsfunc program.

You can load the Nlsfunc program from either your AUTOEXEC.BAT or CONFIG.SYS file. In your AUTOEXEC.BAT file, the **nlsfunc** command should precede any commands that load or switch code pages.

If you want to load the Nlsfunc program from your CONFIG.SYS file, use the **install** command. For example, if NLSFUNC.EXE is in the C:\DOS directory, use the following **install** command in your CONFIG.SYS file:

```
install=c:\dos\nlsfunc.exe
```

On the same line, you can specify a code page and the path for the COUNTRY.SYS file if you do not have a **country** command in your CONFIG.SYS file.

Loading a Code Page into Memory

In Brief

To load a code page into memory, use the **mode cp prep** command with CON or the name of the port your printer is attached to. For example, the following command loads code page 850 from the file C:\DOS\EGA.CPI for an EGA or VGA monitor:

```
mode con cp prep=((850)c:\dos\ega.cpi)
```

The following command loads code page 850 from the file C:\DOS\4201.CPI for an IBM Proprinter 4201 attached to LPT1:

```
mode lpt1 cp prep=((850)c:\dos\4201.cpi)
```

The **device** command you include in your CONFIG.SYS file installs the device driver necessary for using a prepared code page, but it doesn't load the code page. To load a prepared code page, you use a **mode codepage prepare** command (**codepage prepare** can be abbreviated as **cp prep**). The **mode** command gets the code page you want from the .CPI file in which it is stored and loads it into memory.

Once the code page is in memory, you can make it active, display information with it, or reload it by using other forms of the **mode** command.

When you use a **mode cp prep** command, you specify the following:

- The device for which you want to load the code page. You would type **con** to load the code page for the screen and keyboard. You would type **prn**, **lpt1**, **lpt2**, or **lpt3** to load the code page for a printer port.
- The prepared code page(s) you want to load. You can load as many code pages as you reserved space for in the **device** command (included in your CONFIG.SYS file).
- The file in which the code page is stored. All code-page files have a .CPI extension. The EGA/VGA code pages are stored in EGA.CPI; the LCD code pages are stored in LCD.CPI. The printer code pages are stored in 4201.CPI, 4208.CPI, and 5202.CPI.

For example, the following command loads code page 850 from C:\DOS\EGA.CPI into memory:

```
mode con cp prep=((850)c:\dos\ega.cpi)
```

The name of the device must precede the **cp prep** portion of the command. Use parentheses to group together all the other parameters. Within these parentheses, use another set of parentheses to group together the code pages you want to load.

If you reserved space for more than one code page in the **device** command in your CONFIG.SYS file, you can load more than one code page. For example, the following command loads code pages 850 and 865:

```
mode con cp prep=((850 865)c:\dos\ega.cpi)
```

Use a space to separate the two code pages.

To load code pages for your monitor and printer, use two **mode** commands. For example, the following commands load code page 865 for a monitor and a 5202 printer attached to LPT1:

```
mode con cp prep=((865)c:\dos\ega.cpi)
mode lpt1 cp prep=((865)c:\dos\5202.cpi)
```

Making a Code Page Active

In Brief

To make a code page active for all your devices, use the **chcp** command, as in the following example:

```
chcp 850
```

To make a code page active for one device, use the **mode cp select** command. For example, the following command makes code page 850 active for a monitor and keyboard:

```
mode con cp select=850
```

After you have installed the device driver, loaded the Nlsfunc program, and loaded the code page into memory, you make the code page active. To make a code page active for all devices, use the **chcp** (change code page) command. To make it active for one device, use the **mode cp select** command.

You cannot change to a prepared code page if a device uses only a hardware code page or if the code page has not been loaded for the device. In addition, you cannot change the code page used with the keyboard if it is not compatible with the keyboard code of the country. For example, the Danish keyboard (keyboard code dk) can only be used with code pages 850 and 865. You cannot change the code page used with the Danish keyboard to 437.

Using the Chcp Command to Change a Code Page

Using the **chcp** command, you can make a code page active for every device that can use it. For example, the following command makes code page 850 active for every device:

```
chcp 850
```

If the code page has not been loaded for one or more of your devices, MS-DOS displays a message similar to the following:

```
Code page 850 not prepared for all devices
```

If a device cannot use the code page, it retains its original code page. For example, if you try to make a code page active that has not been loaded for your printer, MS-DOS makes the code page active for your keyboard and monitor but not for your printer.

Using the Mode Command to Change a Code Page

To make a code page active for one device, you use a **mode cp select** command and specify the name of the device and the code page to which you want to change. For example, to change to code page 850 for a printer attached to LPT1, you would use the following command:

```
mode lpt1 cp select=850
```

Before this command can be successful, you must load code page 850 for your printer. If the code page is not loaded, MS-DOS displays a message indicating that the code page has not been prepared.

Viewing Information about Code Pages

In Brief

To view information about code pages for your keyboard, monitor, and printer, use the **mode** command without a parameter:

```
mode
```

To view information about code pages for your keyboard and monitor only, use the **keyb** command without a parameter:

```
keyb
```

To view the active code page, use the **chcp** command without a parameter:

```
chcp
```

When you use the **mode**, **keyb**, or **chcp** command without a parameter, you can view information about any code pages you are using.

The **mode** command lists the active code page, and the hardware code page. It also lists any prepared code pages for the console (monitor and keyboard); LPT1, LPT2, and LPT3; and any existing serial ports, such as COM1. For example, suppose you have loaded code pages for LPT1 and your keyboard and monitor. If you type the **mode** command without a parameter, the following type of information will be displayed:

```
Status for device LPT1:
-----------------------
LPT1: not rerouted
RETRY=NONE

No code page has been selected
Hardware code pages:
  code page 437
Prepared code pages:
  code page 850

Status for device CON:
----------------------
Columns=80
Lines=25

Active code page for device CON is 850
Hardware code pages:
  code page 437
Prepared code pages:
  code page 850
```

In this case, both the LPT1 printer and the console have hardware code page 437. Both devices also have code page 850 loaded for them, but only the keyboard and monitor have 850 as the active code page.

To view information about the keyboard and monitor or any LPT port separately, use a **mode cp** command. For example, the following command displays information about code pages for LPT2:

```
mode lpt2 cp
```

If you type **keyb** without parameters, MS-DOS displays a message indicating which code pages are in use with your keyboard and monitor. For example, suppose you are using a German keyboard and code page 850 for your keyboard and monitor. When you type **keyb** with no parameters, MS-DOS displays the following message:

```
Current keyboard code: GR  code page:850
Current CON code page: 850
```

To view the active code page, type the **chcp** command without parameters.

Sample Language Changes

If you don't need to change code pages, you only use two commands to change language character sets. If you do need to change code pages, the number of commands you use depends on how many code pages you want to use and whether you want to use them with your keyboard and monitor only, or with your keyboard, monitor, and printer.

If you always use one particular language and you need to use code pages for that language, it may be convenient to put all the commands you need in your AUTOEXEC.BAT file. Then, whenever you start your system, MS-DOS sets your keyboard, monitor, and printer for the language you want to use. If you set up your languages in your CONFIG.SYS and AUTOEXEC.BAT files, you can switch between keyboards and code pages when you need them.

Changing Languages Without Changing Code Pages

For many languages, you only need to run a **country** command and a **keyb** command. For example, if your MS-DOS files are in the C:\DOS directory, you use the following two commands to change MS-DOS to Italian conventions and an Italian keyboard:

```
country=039,,c:\dos\country.sys
keyb it,,c:\dos\keyboard.sys
```

The **country** command must be in your CONFIG.SYS file. You can type **keyb** at the command prompt, or put the command in your AUTOEXEC.BAT file or CONFIG.SYS file (if your default code page is the same as the hardware code page). To include the preceding **keyb** command in your CONFIG.SYS file, use the following command:

```
install=c:\dos\keyb.com it,,c:\dos\keyboard.sys
```

Using One Prepared Code Page

If the language you want to use requires a prepared code page, you must add at least two commands to your CONFIG.SYS file and two or more commands to your AUTOEXEC.BAT file.

For example, suppose your hardware code page is 437, but you want to use code page 850 with a Belgian keyboard and an EGA monitor. If the MS-DOS files you need are in the C:\DOS directory, you can use the following commands in your CONFIG.SYS file to change to Belgian conventions (032) and install the display driver that enables you to switch between code pages:

```
country=032,,c:\dos\country.sys
device=c:\dos\display.sys con=(ega,437,1)
```

The **country** command sets the Belgian conventions for date, time, currency, character sort order, and filename characters. The **device** command installs DISPLAY.SYS, indicates that you have an EGA or VGA monitor with a 437 hardware code page, and reserves space for one prepared code page.

In your AUTOEXEC.BAT file, you include the following commands to prepare and select code page 850:

```
cd \dos
nlsfunc
mode con cp prep=((850)c:\dos\ega.cpi)
keyb be,,c:\dos\keyboard.sys
chcp 850
```

You include the **nlsfunc** command so you can switch between code pages for all devices at the same time. The **mode** command loads code page 850 from the EGA.CPI file. The **keyb** command changes the arrangement of the keyboard to match a Belgian keyboard. The **chcp** command makes code page 850 active.

The following **nlsfunc** command could be included in the CONFIG.SYS file rather than in the AUTOEXEC.BAT file:

```
install=c:\dos\nlsfunc.exe
```

You could use the United States keyboard temporarily by pressing CTRL+ALT+F1. To return to the Belgian keyboard, press CTRL+ALT+F2.

Using Two Prepared Code Pages

Suppose you want to use code pages 850 and 863 with your VGA monitor and Canadian-French keyboard. If the hardware code page is 437, and the MS-DOS files you need are in C:\DOS, you add the following two commands to your CONFIG.SYS file to change to Canadian-French conventions and install the display driver:

```
country=002,,c:\dos\country.sys
device=c:\dos\display.sys con=(ega,437,2)
```

The **country** command sets the Canadian-French conventions, and the **device** command reserves space for two prepared code pages. Notice that the EGA value also works for VGA monitors.

To load both code pages and make 850 the active one, include the following commands in your AUTOEXEC.BAT file:

```
cd \dos
nlsfunc
mode con cp prep=((850 863)c:\dos\ega.cpi)
keyb cf,,c:\dos\keyboard.sys
chcp 850
```

The **mode** command loads both prepared code pages from the EGA.CPI file into memory. The **chcp** command makes code page 850 active. While you are working, you can switch to code page 863 by typing the following **chcp** command:

```
chcp 863
```

Using Prepared Code Pages with Your Printer

To add the Canadian-French code pages described in the previous section to an IBM Proprinter 4208 or compatible printer, you add another **device** command to your CONFIG.SYS file and another **mode** command to your AUTOEXEC.BAT file.

With the added **device** command, your CONFIG.SYS file includes these commands:

```
country=002,,c:\dos\country.sys
device=c:\dos\display.sys con=(ega,437,2)
device=c:\dos\printer.sys lpt1=(4208,437,2)
```

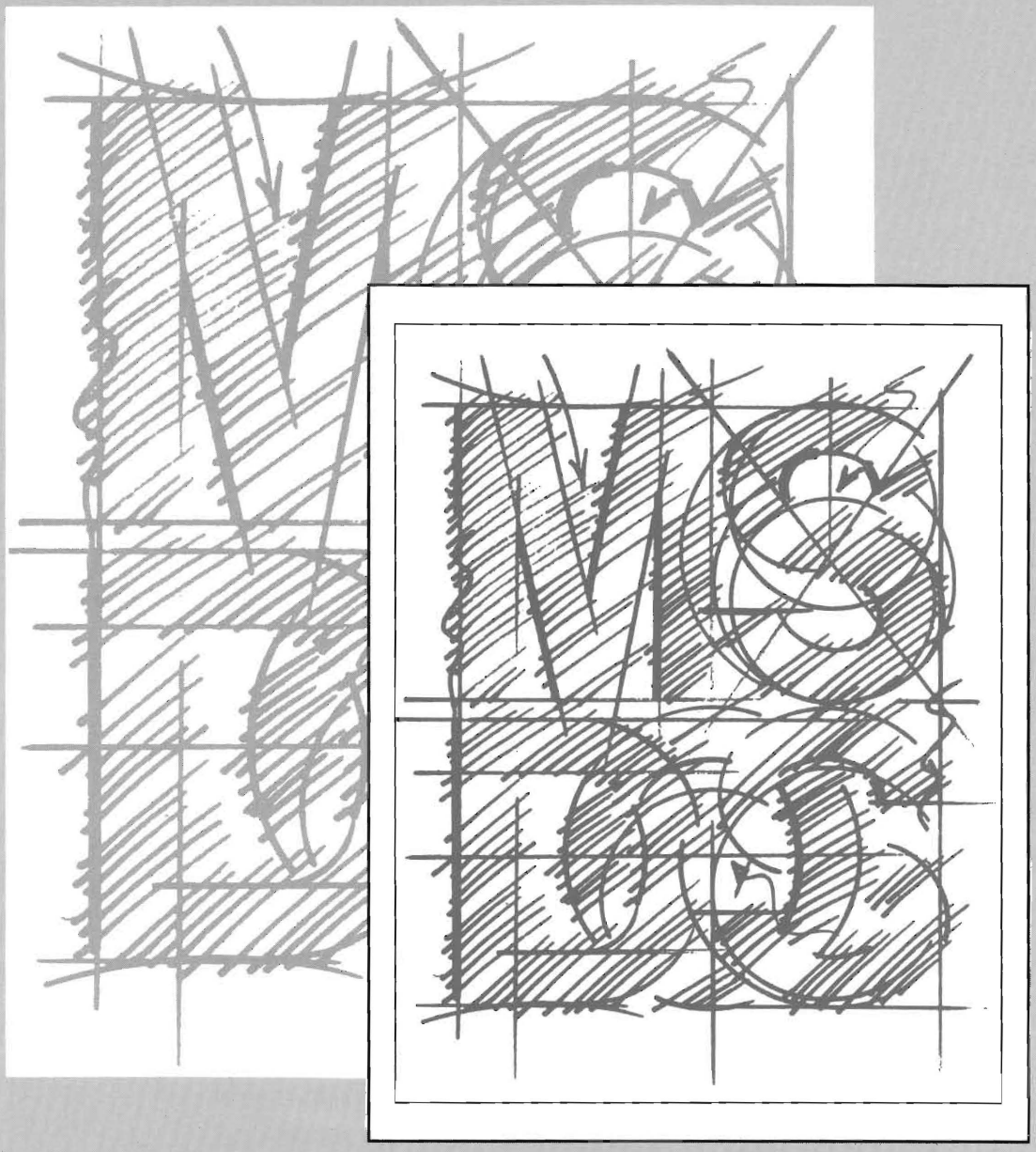
The second **device** command installs PRINTER.SYS and specifies an IBM Proprinter 4208 or compatible printer with a 437 hardware code page. Like the first **device** command listed, the second command reserves space for two prepared code pages.

With the added **mode** command, your AUTOEXEC.BAT file includes these commands:

```
cd \dos
nlsfunc
mode con cp prep=((850 863)c:\dos\ega.cpi)
mode lpt1 cp prep=((850 863)c:\dos\4208.cpi)
keyb cf,,c:\dos\keyboard.sys
chcp 850
```

The second **mode** command loads code pages 850 and 863 for the 4208 printer from the 4208.CPI file. The **chcp** command makes code page 850 active for all three devices.

Part 4
MS-DOS Reference



Chapter 14

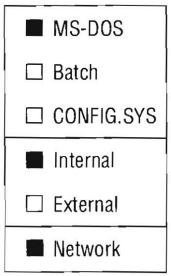
Commands

14

This chapter is a complete reference for the commands supplied with MS-DOS version 5.0. The chapter includes descriptions of the types of commands, an explanation of command syntax, and a description of each command.

Types of Commands

Each command entry is accompanied by a chart that summarizes the characteristics of the command. For example, the chart for the **copy** command is as follows:



If a check box is filled, its associated characteristic applies to the command. As this example shows, more than one characteristic can apply to a given command. A description of each characteristic follows:

MS-DOS

MS-DOS commands are basic instructions provided with MS-DOS version 5.0. For an introduction to using MS-DOS commands, see Chapter 2, "Command-Line Basics."

Batch

Batch commands are *internal commands* that you can use to direct how a batch program runs. For a description of internal commands, see the "Internal" topic later in this section. For an introduction to using batch commands, see Chapter 10, "Working with Batch Programs."

CONFIG.SYS

CONFIG.SYS commands are commands that you can use to customize your system. These commands are useful for such tasks as installing device drivers, setting limits on files and buffers, and carrying out MS-DOS commands during CONFIG.SYS processing. For an introduction to using CONFIG.SYS commands, see Chapter 11, "Customizing Your System."

Internal

Internal commands are stored in the COMMAND.COM file, which is loaded into memory when you start your system. They include the simpler, more commonly used commands you need on a regular basis. Because internal commands are part of COMMAND.COM, you never see their names in a directory listing. These commands remain resident in memory and are available to you at all times.

External

External commands exist as separate files on your disk. When you use the **dir** command to view the files on your MS-DOS system disk, you see the external commands in the list of filenames and directory names. The filename of an external command has a .COM, .EXE, or .BAT extension.

External commands sometimes perform in ways similar to programs. For that reason, some users may refer to them as utilities. However, in this book, they are called commands.

Network

Not all MS-DOS commands are designed to be used on a network. If this check box is filled, it indicates that the command *can* be used on a network.

MS-DOS Commands

append
assign
attrib
backup
break
chcp
chdir (cd)
chkdsk
cls
command
comp
copy
ctty
date
debug
(See Debug Commands)
del (erase)
dir
diskcomp
diskcopy
doskey
dosshell
edit
edlin
(See Edlin Commands)
emm386
exe2bin
exit
expand
fastopen
fc
fdisk
find
for
format
graftabl
graphics
help
join
keyb
label
loadhigh (lh)
mem
mirror
mkdir (md)
mode
(configure printer)
(configure serial port)
(display device status)
(redirect printing)
(set device code pages)
(set display mode)
(set typematic rate)
more
nlsfunc
path
print
prompt
qbasic
recover
rename (ren)
replace
restore
rmdir (rd)
set
setver
share
sort
subst
sys
time
tree
type
undelete
unformat
ver
verify
vol
xcopy

Batch Commands

call
echo
for
goto
if
pause
rem
shift

CONFIG.SYS Commands

break	**devicehigh**	**files**	**shell**
buffers	**dos**	**install**	**stacks**
country	**drivparm**	**lastdrive**	**switches**
device	**fcbs**	**rem**	

Debug Commands

a (assemble)
c (compare)
d (dump)
e (enter)
f (fill)
g (go)
h (hex)
i (input)
l (load)
m (move)
n (name)
o (output)
p (proceed)
q (quit)
r (register)
s (search)
t (trace)
u (unassemble)
w (write)
xa (allocate expanded memory)
xd (deallocate expanded memory)
xm (map expanded-memory pages)
xs (display expanded-memory status)

Edlin Commands

(*line*)	**e** (end)	**p** (page)	**t** (transfer)
a (append)	**i** (insert)	**q** (quit)	**w** (write)
c (copy)	**l** (list)	**r** (replace)	
d (delete)	**m** (move)	**s** (search)	

Syntax Conventions

Syntax represents the order in which you must type an MS-DOS command and any parameters and switches that follow it. Elements that appear in bold type must be typed exactly as they appear in the syntax line; elements that appear in italic are placeholders representing specific information that you supply.

Unless specified otherwise, you can type commands, parameters, and switches in either uppercase or lowercase letters. The word "type," as it is used in this book, means to press a key, a sequence of keys, or a key combination and then press the ENTER key.

The following is a sample syntax line:



The meaning of these elements is as follows:

Number	Element	Meaning
❶	**sample**	Specifies the name of the command.
❷	[ ]	Indicates an item that is optional. To include the optional information described within the brackets, type only the information, not the brackets themselves.
❸	|	Separates two mutually exclusive choices in a syntax line, as shown in the following example: **break [on|off]** Type only one of these choices (that is, **break on** or **break off**); do not type the pipe (|) itself. In this reference the pipe (¦) is used as a redirection symbol. To the computer, the two pipes (| and ¦) are equivalent—that is, you press the same key to type any pipe. Before using a pipe for redirection, you should set the TEMP environment variable in your AUTOEXEC.BAT file. For more information about pipes and command redirection, see Chapter 7, "Advanced Command Techniques."
❹	*drive*:	Specifies the name of a hard disk drive or floppy disk drive. To carry out an external command when its file is not on the disk in the current drive or in the search path, you must specify the correct drive. You never need to specify the drive of an internal command.
❺	*path*	Specifies the route the operating system is to follow through the directory structure to locate a directory or file. You need to specify a path with a filename only if the file is not in the current directory. For information about specifying *path*, see Chapter 5, "Working with Directories."
❻	*filename*	Specifies the name of a file. A filename can be up to eight characters long and can be followed by a period (.) and an extension of up to three characters (for example, YOURFILE.TXT). In this book, lowercase filenames are used in example blocks and uppercase filenames are used in other sections. You can type either uppercase or lowercase filenames. You cannot specify a device name or drive letter for *filename*.
❼	...	Indicates that the previous parameter or switch can be repeated several times in a command. Type only the information, not the ellipsis (...) itself.
❽	*options*	Specifies one or more optional command parameters or switches. A switch usually begins with a slash—for example, **/p**.

Other placeholders used in syntax lines in this manual include the following:

Placeholder	Meaning
source	Specifies the location of data to be transferred to a specified destination or used as input to a command. *Source* can consist of a drive letter and colon, a directory name, a filename, or a combination.
destination	Specifies a location to which the data specified by *source* is to be transferred. *Destination* can consist of a drive letter and colon, a directory name, a filename, or a combination.
string	Specifies a group of characters to be treated as a unit. A string can include letters, numbers, spaces, or any other characters and is usually enclosed in double quotation marks. Some commands, such as **find**, work with strings of text.

Online Help with Commands

MS-DOS version 5.0 includes online Help for MS-DOS commands. To get Help with the syntax, parameters, and switches of any MS-DOS command, type the command name followed by **/?** on the command line or type **help** followed by the command name. For example, for Help information about the **copy** command, type one of the following:

```
copy /?
help copy
```

MS-DOS displays information about the command syntax, parameters, and switches.

To see a list of all MS-DOS commands and a brief description of the purpose of each command, type **help** with no parameters or switches.

■ MS-DOS
□ Batch
□ CONFIG.SYS
□ Internal
■ External
■ Network

Append

Enables programs to open data files in specified directories as if these files were in the current directory.

The specified directories are called *appended directories* because, for the sake of opening data files, they can be found as if they were appended to the current directory.

Syntax

append [[*drive*:]*path*[;...]] [**/x**[**:on**|**:off**]] [**/path:on**|**/path:off**] [**/e**]

To display the list of appended directories, use the following syntax:

append

To cancel the existing list of appended directories, use the following syntax:

append ;

Parameters

[*drive*:]*path*

Specifies the drive (if other than the current drive) and directory that you want to append to the current directory. You can specify multiple entries of [*drive*:]*path*, separating the entries with semicolons.

; When used by itself (**append ;**), cancels the existing list of appended directories.

Switches

/x[**:on**|**:off**]

Specifies whether MS-DOS is to search appended directories when executing programs. If you use the **/x:on** switch, the program does search appended directories. If you use the **/x:off** switch, the program does not search appended directories. You can abbreviate **/x:on** to **/x**. If you want to specify **x:on**, you must do it the first time you use **append** after starting your system. After that, you can switch between **x:on** and **x:off**.

/path:on|**/path:off**

Specifies whether a program is to search appended directories for a data file when a path is already included with the name of the file the program is looking for. The default setting is **/path:on**.

/e Assigns the list of appended directories to an environment variable named APPEND. This switch can be used only the first time you use **append** after starting your system. If you use **/e**, you can use the **set** command to display the list of appended directories. For information about environment variables, see the **set** command.

Notes

Storing the list of appended directories in the environment

You can use the **/e** switch with **append** to assign the list of appended directories to an environment variable named APPEND. To do this, first use the **append** command with only the **/e** switch. Then use **append** again, this time including the directories you want to append. You cannot specify **/e** and [*drive*:]*path* on the same command line.

Specifying multiple appended directories

To append more than one directory, separate multiple entries with semicolons. If you use the **append** command with the [*drive*:]*path* parameters again, the specified directory or directories replace any directories specified in a previous **append** command.

Appended directories and the dir command

The **dir** command does not add filenames from appended directories to directory listings produced by the **dir** command.

Filename conflicts

If a file in an appended directory has the same name as a file in the current directory, programs open the file in the current directory.

Using append with programs that create new files

When a program opens a file in an appended directory, the file can be found as if it were in the current directory. If the program then saves the file by creating a new file with the same name, the new file is created in the current directory (not the appended directory). **Append** is appropriately used for data files that are not to be modified or that are to be modified without creating new copies of the files. Database programs often modify data files without making new copies. Text editors and word processors, however, usually save modified data files by making new copies. To avoid confusion, you might want to avoid using **append** with these programs.

Using the /x:on switch and the path command

When **/x:on** is specified, you can run a program located in an appended directory by typing the program name at the command prompt. Usually, you use the **path** command to specify directories that contain programs. However, when your program is in an appended directory, you do not need to use the **path** command to specify that directory. MS-DOS finds a program in an appended directory by following the usual order in which MS-DOS searches for a program; that is, first in the current directory, then in the appended directories, and then in the search path.

MS-DOS functions that always use appended directories

Even when the **/x:on** switch is not specified, appended directories are used when programs call the following MS-DOS Interrupt 21h functions:

- Open File (0Fh)
- Open File Handle (3Dh)
- Get File Size (23h)

When **/x:on** is specified, appended directories are used when programs call any of the Interrupt 21h functions in the preceding list or any of the Interrupt 21h functions in the following list:

- Find First Entry (11h)
- Find First File (4Eh)
- Execute Program (EXEC) (4Bh)

Using append with the assign command

If you use both the **append** and **assign** commands, you must use **append** first, even if the commands affect different drives. When you use the **assign** command to assign a different drive letter to an existing disk drive, you cannot use the **append** command to append directories that use the new drive letter.

Using append with network drives

You can use the **append** command to append directories that are located on network drives.

Examples

To allow programs to open data files in a directory named LETTERS on the disk in drive B and in a directory named REPORTS on the disk in drive A as if the files were in the current directory, type the following command:

```
append b:\letters;a:\reports
```

To append the same directories and keep a copy of the list of appended directories in the MS-DOS environment, type the following commands:

```
append /e

append b:\letters;a:\reports
```

These must be the first **append** commands you use after starting your system.

Related Command

For information about setting a search path for executable files, see the **path** command.

■ MS-DOS
□ Batch
□ CONFIG.SYS
□ Internal
■ External
■ Network

Assign

Redirects requests for disk operations on one drive to a different drive.

Some older programs can read and write files only on drives A and B. With the **assign** command, you can redirect disk operations for those programs so that you can read and write files on drives other than A and B.

Syntax

assign [*x*[**:**]=*y*[**:**][...]]

To redirect all drive letters to their original drives, use the following syntax:

assign

To display a list of the current assignments, use the following syntax:

assign /status

Parameters

x Specifies the drive from which you want to redirect read and write operations. This value must be a letter. The use of the colon (:) is optional.

y Specifies the existing drive to which you want to redirect read and write operations. This value must be a letter. The use of the colon (:) is optional.

Switch

/status

Lists current assignments. You can abbreviate this switch as **/sta** or **/s**.

Notes

Invalid uses of assign

You must not assign the drive letter of your hard disk to another drive. You should not use **assign** for a drive that is in use by a program.

You cannot use the drive letter of a hard disk drive that does not exist for either the *x* or the *y* parameter.

Avoid the use of **assign** in the following cases:

- With commands requiring drive information (**backup**, **join**, **label**, **restore**, **subst**)
- With the **diskcopy** and **format** commands, which ignore drive reassignments
- During typical use of MS-DOS, unless a program cannot read and write files on the specified drive

Using assign with the append command

If you use both the **assign** and **append** commands, you must use **append** first, even if the commands affect different drives.

Using assign for network drives

You can use the **assign** command for network drives.

Canceling a previous assignment as the result of a new assignment

Assigning a drive letter to a drive cancels previous assignments to it. Suppose you assign drive A to drive letter C, as the following example shows:

```
assign a=c
```

Later you assign drive B to drive letter C, as the following example shows:

```
assign b=c
```

As a result, drive A is no longer assigned to drive letter C.

Using the subst command instead of assign

You should use the **subst** command instead of **assign**. The following commands are equivalent:

```
assign a=c

subst a: c:\
```

Examples

Suppose you want to use drive C to read and write files, but your program requires you to put your program disk into drive A and your data disk into drive B. To reassign the drive letters A and B to drive C, type the following command:

```
assign a=c b=c
```

This command causes MS-DOS to look for your program and data files on drive C.

To reset all drive letters to their original drives, type the **assign** command without parameters, as follows:

```
assign
```

Related Command

For information about associating a drive letter with a path in a way that ensures compatibility with future versions of MS-DOS, see the **subst** command.

- ■ MS-DOS
- □ Batch
- □ CONFIG.SYS
- □ Internal
- ■ External
- ■ Network

Attrib

Displays or changes file attributes.

This command displays, sets, or removes the read-only, archive, system, and hidden attributes assigned to files. For an introduction to the **attrib** command, see Chapter 4, "Working with Files."

Syntax

attrib [+r|–r] [+a|–a] [+s|–s] [+h|–h] [[*drive*:][*path*]*filename*] [/s]

To display all attributes of all files in the current directory, use the following syntax:

attrib

Parameter

[*drive*:][*path*]*filename*

Specifies the location and name of the file or set of files you want to process.

Switches

+r Sets the read-only file attribute.

–r Clears the read-only file attribute.

+a Sets the archive file attribute.

–a Clears the archive file attribute.

+s Sets the file as a system file.

–s Clears the system file attribute.

+h Sets the file as a hidden file.

–h Clears the hidden file attribute.

/s Processes files in the current directory and all of its subdirectories.

Notes

Groups of files

You can use wildcards (? and *) with the *filename* parameter to display or change the attributes for a group of files. If a file has the system or hidden attribute set, you must clear that attribute before you can change any other attributes for that file.

Archive attributes

The archive attribute (**a**) is used to mark files that have changed since they were previously backed up. The **backup**, **restore**, and **xcopy** commands use these archive attributes. For information about archive attributes, see the **backup**, **restore**, and **xcopy** commands.

Examples

To display the attributes of a file named NEWS86 located on the current drive, type the following command:

```
attrib news86
```

To assign the read-only attribute to the file REPORT.TXT, type the following command:

```
attrib +r report.txt
```

To remove the read-only attribute from files in the \USER\JONES directory on a disk in drive B and from files in any subdirectories of \USER\JONES, type the following command:

```
attrib -r b:\user\jones\*.* /s
```

As a final example, suppose you want to give an associate a disk containing all files in the default directory on a disk in drive A except files with the .BAK extension. Because you can use **xcopy** to copy only those files marked with the archive attribute, you need to set the archive attribute for those files you want to copy. To do this, you use the following two commands to set the archive attribute for all files on drive A and then to clear the attribute for those files with the .BAK extension:

```
attrib +a a:*.*

attrib -a a:*.bak
```

Next, use the **xcopy** command to copy the files from the disk in drive A to the disk in drive B. The **/a** switch in the following command causes **xcopy** to copy only those files marked with the archive attribute:

```
xcopy a: b: /a
```

If you want **xcopy** to clear each file's archive attribute after it copies the file, use the **/m** switch instead of **/a**, as in the following example:

```
xcopy a: b: /m
```

Related Command

For more information about copying files and directories, see the **xcopy** command.

- ■ MS-DOS
- ☐ Batch
- ☐ CONFIG.SYS
- ☐ Internal
- ■ External
- ■ Network

Backup

Backs up one or more files from one disk onto another.

You can back up files onto either a hard disk or floppy disk(s). Files can also be backed up from one floppy disk onto another, even if the disks have different numbers of sides or sectors. MS-DOS displays the name of each file it backs up. For an introduction to the **backup** command, see Chapter 6, "Managing Disks."

Syntax

backup *source destination-drive*: [**/s**] [**/m**] [**/a**] [**/f**[:*size*]] [**/d**:*date* [**/t**:*time*]] [**/l**[:[*drive*:][*path*]*logfile*]]

Parameters

source

Specifies the location of files you want to back up. *Source* can consist of a drive letter and colon, a directory name, a filename, or a combination.

destination-drive:

Specifies the drive that contains the disk on which you want to store any backup files. The backup files are stored in the BACKUP.*nnn* and CONTROL.*nnn* files. That is, **backup** assigns the names BACKUP.001 and CONTROL.001 to the files it creates on the first backup disk you use, BACKUP.002 and CONTROL.002 to the files it creates on the second backup disk, and so on.

Switches

/s Backs up the contents of all subdirectories.

/m Backs up only files that have changed since the last backup, and turns off the archive attribute of the original files.

/a Adds backup files to an existing backup disk without deleting existing files. (The **/a** switch is ignored if the existing backup disk contains backup files that were created by using the **backup** command from MS-DOS version 3.2 or earlier.)

/f[:*size*]

Formats the backup disk to the size you specify. (The **format** command must be present in the current path.) With this switch, you direct **backup** to format floppy disks that do not match the default size of the drive. The **backup** command formats an unformatted destination disk even if you do not specify the **/f** switch. When **backup** finishes formatting, it begins backing up files onto the last disk it formatted. *Size* specifies the size in kilobytes of the disk to be formatted. If you do not specify *size*, the **/f** switch uses the default size of the drive. The following list shows the valid values for *size* and a brief description of each size:

160 or **160k** or **160kb**

160K, single-sided, double-density, 5.25-inch disk

180 or **180k** or **180kb**

180K, single-sided, double-density, 5.25-inch disk

320 or **320k** or **320kb**

320K, double-sided, double-density, 5.25-inch disk

360 or **360k** or **360kb**

360K, double-sided, double-density, 5.25-inch disk

720 or **720k** or **720kb**

720K, double-sided, double-density, 3.5-inch disk

1200 or **1200k** or **1200kb** or **1.2** or **1.2m** or **1.2mb**

1.2-MB, double-sided, quadruple-density, 5.25-inch disk

1440 or **1440k** or **1440kb** or **1.44** or **1.44m** or **1.44mb**

1.44-MB, double-sided, quadruple-density, 3.5-inch disk

2880 or **2880k** or **2880kb** or **2.88** or **2.88m** or **2.88mb**

2.88-MB, double-sided, 3.5-inch disk

/d:*date*

Backs up only files modified on or after the specified date. The date format depends on the setting you are using for the **country** command. For information about changing the date format, see Chapter 13, "Customizing for International Use."

/t:*time*

Backs up only files modified at or after the specified time. The time format depends on the setting you are using for the **country** command. For information about changing the time format, see Chapter 13, "Customizing for International Use." Do not use the **/t** switch without the **/d** switch.

/l[**:**[*drive***:**][*path*]*logfile*]

Creates a log file and adds an entry to that file to record the backup operation. If you do not specify a location for the log file, **backup** puts the file in the root directory of the source drive. If you do not specify *logfile*, **backup** names the file BACKUP.LOG. You should not specify a removable drive (such as a floppy disk drive) for this parameter; but once the backup is complete, you can copy the log file to a floppy disk.

Notes

Backing up onto a disk with files

Unless you use the **/a** switch, **backup** deletes old files (including read-only files) from a backup disk before adding new files to it.

Backup log file

If you use the **/l** switch and do not specify a name and location for the log file, the **backup** command adds a file named BACKUP.LOG to the root directory of the source drive. If the BACKUP.LOG file already exists, **backup** adds the current entry to the file. A backup log-file entry uses the following format:

- The date and time of the backup appear on the first line.
- Each filename appears on a separate line with the number of the backup disk that contains the file.

The backup log file can assist you later, when you need to identify the files you want to restore. The **restore** command always returns a file to the original directory or sub-directory recorded in the backup log, creating the subdirectory if necessary.

Labeling backup disks

It is important to label and number backup disks consecutively. As each disk is filled, **backup** prompts you for the next disk. When you restore files, you need to insert the backup disks into the disk drive in the same sequence. To check the sequence of backup disks (MS-DOS version 3.3 or later), use the **dir** command to check the disk number.

Backup and system files

The **backup** command cannot back up the system files IO.SYS, MSDOS.SYS, and COMMAND.COM. You can use the **sys** command to copy these files onto a floppy disk.

Using an old version of the restore command

You cannot use an old version of the **restore** command (MS-DOS version 3.2 or earlier) for files backed up with MS-DOS version 3.3 or later. If you attempt this, MS-DOS displays the following message:

```
Source does not contain backup files
```

This error occurs because the format of old backup files differs from the format of files backed up with MS-DOS versions 3.3 and later.

Using backup with networks or redirected drives or directories

If you are sharing files on a network, you can back up only those files to which you have access. You should not use **backup** with a drive that has been redirected with the **assign**, **join**, or **subst** command. If you do, the **restore** command may not be able to restore the files.

Backup exit codes

The following list shows each exit code and a brief description of its meaning:

0 The backup was successful.

1 No files were found to back up.

2 Some files were not backed up because of file-sharing conflicts.

3 The user pressed CTRL+C to stop the process.

4 The process stopped because of an error.

You can use the **errorlevel** parameter on the **if** command line in a batch program to process exit codes returned by the **backup** command. For an example of a batch program that processes exit codes, see the following "Examples" section. For more information about batch programs, see Chapter 10, "Working with Batch Programs."

Examples

Suppose you want to back up all the files in the \USER\SMITH directory on drive C onto a blank, formatted disk in drive A. To do so, type the following:

```
backup c:\user\smith\*.* a:
```

Suppose you need to back up all files in the \USER\SMITH directory on drive C onto a 720K floppy disk in drive B. If the floppy disk is unformatted, **backup** formats it before backing up any files. Because the **/s** switch is not specified in the following command, files in subdirectories are not backed up:

```
backup c:\user\smith\*.* b: /f:720k
```

To write a simple batch program named SMITH that supports the **backup** command's exit codes and the **/s** switch, you can type the following commands by using MS-DOS Editor:

```
echo off
rem Smith's backup command
backup c:\user\smith\*.* b: /s
if errorlevel 4 goto error
if errorlevel 3 goto abort
if errorlevel 2 goto conflict
if errorlevel 1 goto no_files
if errorlevel 0 goto success
:error
echo Backup stopped the process due to an error
goto exit
:abort
echo You just pressed CTRL+C to stop the backup
goto exit
:conflict
echo One or more files were not backed up due to a sharing conflict
goto exit
:no_files
echo Sorry, but there were no files to backup
goto exit
:success
echo The backup was successful
goto exit
:exit
```

For more information about using the **if** command in batch programs, see the **if** command.

Related Command

For information about restoring a backup file, see the **restore** command.

- ■ MS-DOS
- ☐ Batch
- ■ CONFIG.SYS
- ☐ Internal
- ☐ External
- ☐ Network

Break

Sets or clears extended CTRL+C checking.

You can press CTRL+C to stop a program or an activity (file sorting, for example). Typically, MS-DOS checks for CTRL+C only while it reads from the keyboard or writes to the screen or a printer. If you set **break** to **on**, you extend CTRL+C checking to other functions, such as disk read and write operations. For an introduction to the **break** command, see Chapter 11, "Customizing Your System."

Syntax

break [on|off]

To display the current **break** setting, use the following syntax:

break

In your CONFIG.SYS file, use the following syntax:

break=on|off

Parameter

on|off

Turns extended CTRL+C checking on or off.

Notes

Including break in CONFIG.SYS

The default setting for **break** is **off**. You can include the **break** command in your CONFIG.SYS file to enable extended CTRL+C checking every time you start your system.

Break=on slows down your system

The disadvantage of setting **break=on** is that it slightly decreases the speed of your system.

Examples

To specify that MS-DOS is to check for CTRL+C only while it is reading from the keyboard or writing to the screen or printer, type the following command:

```
break off
```

To specify that MS-DOS is to check for CTRL+C while it is reading from a disk or the keyboard or writing to a disk or the screen, type the following command:

```
break on
```

To turn on this extended CTRL+C checking every time you start your system, include the following command in your CONFIG.SYS file:

```
break=on
```

☐ MS-DOS
☐ Batch
■ CONFIG.SYS
☐ Internal
☐ External
☐ Network

Buffers

Allocates memory for a specified number of disk buffers when your system starts.

For an introduction to using the CONFIG.SYS file, see Chapter 11, "Customizing Your System."

Syntax

buffers=*n*[,*m*]

Parameters

n Specifies the number of disk buffers. The value of *n* must be in the range 1 through 99.

m Specifies the number of buffers in the secondary buffer cache. The value of *m* must be in the range 1 through 8.

Notes

Default settings

The default setting for the number of disk buffers depends upon the configuration of your system, as shown in the following table:

Configuration	Buffers (*n*)	Bytes
< 128K of RAM, 360K disk	2	—
< 128K of RAM, > 360K disk	3	—
128K to 255K of RAM	5	2672
256K to 511K of RAM	10	5328
512K to 640K of RAM	15	7984

The default setting for the number of buffers in the secondary cache (*m*) is 1.

If you specify an invalid value for *n* or *m*, **buffers** uses the default value instead.

Using the secondary buffer cache

Using the cache can speed up certain disk operations. For more information about using the cache, see Chapter 12, "Optimizing Your System."

How MS-DOS uses buffers

MS-DOS uses the memory reserved for each disk buffer to hold data during read and write operations. To achieve the best performance with programs such as word processors, specify a value between 10 and 20 for *n*. If you plan to create many subdirectories, you might want to increase the number of buffers to 20 or 30. Each buffer requires approximately 532 bytes of memory. Therefore, the more buffers you have, the less memory you have available for programs.

If MS-DOS is in the high memory area (HMA), the buffers are also in HMA. This leaves more conventional memory for your program.

Example

To create 20 disk buffers, include the following command in your CONFIG.SYS file:

```
buffers=20
```

- ☐ MS-DOS
- ■ Batch
- ☐ CONFIG.SYS
- ■ Internal
- ☐ External
- ☐ Network

Call

Calls one batch program from another without causing the parent batch program to stop.

For an introduction to using batch programs, see Chapter 10, "Working with Batch Programs."

Syntax

call [*drive*:][*path*]*filename* [*batch-parameters*]

Parameters

[*drive*:][*path*]*filename*

Specifies the location and name of the batch program you want to call. *Filename* must have a .BAT extension.

batch-parameters

Specifies any command-line information required by the batch program.

Notes

Using *batch-parameters*

Batch-parameters can contain any information that you can pass to a batch program, including switches, filenames, the replaceable parameters **%1** through **%9**, and variables such as %baud%. For more information about these variables, see Chapter 10, "Working with Batch Programs."

Using pipes and redirection symbols

Do not use pipes and redirection symbols with the **call** command.

Making a recursive call

You can create a batch program that calls itself; however, you must provide an exit condition. Otherwise, the parent and child batch programs can loop endlessly.

Examples

To run the CHECKNEW.BAT program from another batch program, include the following command in the parent batch program:

```
call checknew
```

Suppose the parent batch program accepts two replaceable parameters and you want it to pass those parameters to CHECKNEW.BAT. You can use the following command in the parent batch program:

```
call checknew %1 %2
```

■ MS-DOS
□ Batch
□ CONFIG.SYS
■ Internal
□ External
■ Network

Chcp

Displays the number of the active code page, or changes the active code page that MS-DOS is to use for all devices that support code-page switching to one of the two prepared system code pages associated with your current country setting.

For an introduction to using code pages and the **chcp** command, see Chapter 13, "Customizing for International Use." Tables of the code pages included with MS-DOS are shown in the Appendix, "Keyboard Layouts and Code Pages."

Syntax

chcp [*nnn*]

To display the number of the active code page, use the following syntax:

chcp

Parameter

nnn Specifies the prepared system code pages defined by the **country** command in the CONFIG.SYS file. The following list shows each code page that MS-DOS supports and its country or language:

437	United States
850	Multilingual (Latin I)
852	Slavic (Latin II)
860	Portuguese
863	Canadian-French
865	Nordic

Notes

Requirements for using the chcp command

Before you can use the **chcp** command, you must specify the location of the COUNTRY.SYS file by using the **country** command and load the Nlsfunc program into memory.

Assigning a new code page

After you assign a new code page, any program you start uses that new code page. However, any program (not including COMMAND.COM) that you started before assigning the new code page will probably attempt to use the original code page.

Examples

To view the active code-page setting, type the following command:

```
chcp
```

MS-DOS responds with a message similar to the following:

```
Active code page: 437
```

To change the active code page to 850 (Multilingual), type the following command:

```
chcp 850
```

MS-DOS alerts you if the specified code page has not been prepared for your system. The following error message appears:

```
Invalid code page
```

If a device (screen, keyboard, printer) is not prepared for a code page, MS-DOS displays an error message in the following format:

```
Code page 850 not prepared for device nnn
```

Related Commands

For more information about code pages, see the **country**, **nlsfunc**, **device**, and **mode** (set device code pages) commands.

- ■ MS-DOS
- ☐ Batch
- ☐ CONFIG.SYS
- ■ Internal
- ☐ External
- ■ Network

Chdir (cd)

Displays the name of the current directory or changes the current directory.

For an introduction to the **chdir** command, see Chapter 5, "Working with Directories."

Syntax

chdir [*drive*:][*path*]

chdir[..]

cd [*drive*:][*path*]

cd[..]

To display the names of the current drive and directory, use either of the following syntax lines:

chdir

cd

Parameters

[*drive*:][*path*]

Specifies the drive (if other than the current drive) and directory to which you want to change.

..

Specifies that you want to change to the parent directory.

Notes

Changing to the root directory

The *root directory* is the top of the directory hierarchy for a drive. To return to the root directory, type the following command:

```
cd \
```

Using the current directory from a different drive

If you are working in the \USER\JONES directory on drive C and you change to drive D, you can copy files to and from the \USER\JONES directory by specifying only the drive letter C.

Changing the directory on another drive

You can change the current directory on another drive by specifying the drive name on the command line when you use **chdir**.

Examples

Either of the following commands changes your current directory to the directory named PRIMETIM:

```
chdir \primetim
```

```
cd \primetim
```

Suppose you have a directory named SPECIALS with a subdirectory named SPONSORS. To change your current directory to \SPECIALS\SPONSORS, type the following command:

```
cd \specials\sponsors
```

Or, if your current directory is \SPECIALS, you can use the following command to change to the \SPECIALS\SPONSORS directory:

```
cd sponsors
```

To change from a subdirectory back to the parent directory, type the following command:

```
cd..
```

To display the name of the current directory, you can use **chdir** or **cd** without a parameter. For example, if your current directory is \USER\JONES on the disk in drive B, type **chdir** to see the following response:

```
B:\USER\JONES
```

If you are working on drive D and you want to copy all files in the \USER\JONES and \USER\LEWIS directories on drive C to the root directory on drive D, type the following commands:

```
chdir c:\user\jones
copy c:*.* d:\
chdir c:\user\lewis
copy c:*.* d:\
```

If, instead, you want to copy all files in the \USER\JONES and \USER\LEWIS directories to your *current* location on drive D, type the following commands:

```
chdir c:\user\jones
copy c:*.* d:
chdir c:\user\lewis
copy c:*.* d:
```

■ MS-DOS
☐ Batch
☐ CONFIG.SYS
☐ Internal
■ External
☐ Network

Chkdsk

Creates and displays a status report for a disk.

The status report shows logical errors found in the file allocation table and file system. If errors exist on the disk, **chkdsk** alerts you with a message. You should use **chkdsk** occasionally on each disk to check for errors. For an introduction to the **chkdsk** command, see Chapter 6, "Managing Disks."

Syntax

chkdsk [*drive*:][[*path*]*filename*] [**/f**] [**/v**]

To display the status of the disk in the current drive, use the following syntax:

chkdsk

Parameters

drive:

Specifies the drive that contains the disk that you want **chkdsk** to check.

[*path*]*filename*

Specifies the location and name of a file or set of files that you want **chkdsk** to check for fragmentation. You can use wildcards (* and ?) to specify multiple files.

Switches

/f Fixes errors on the disk.

/v Displays the name of each file in every directory as the disk is checked.

Notes

Format of status reports

MS-DOS displays **chkdsk** status reports in the following format:

```
Volume Serial Number is B1AF-AFBF

  72214528 bytes total disk space
     73728 bytes in 3 hidden files
     30720 bytes in 12 directories
  11493376 bytes in 386 user files
     61440 bytes in bad sectors
  60555264 bytes available on disk

      2048 bytes in each allocation unit
     35261 total allocation units on disk
     29568 available allocation units on disk

    655360 total bytes memory
    493456 bytes free
```

Fixing disk errors

The **chkdsk** command corrects disk errors only if you specify the **/f** switch. Since repairs usually change a disk's file allocation table and sometimes cause loss of data, **chkdsk** first prompts you with a confirmation message similar to the following:

```
10 lost allocation units found in 3 chains.
Convert lost chains to files?
```

If you press Y, MS-DOS saves each lost chain in the root directory as a file with a name in the format FILE*nnnn*.CHK. When **chkdsk** finishes, you can check these files to see if they contain any data you need. If you press N, MS-DOS fixes the disk but does not save the contents of the lost allocation units.

If you do not use the **/f** switch, **chkdsk** alerts you with a message if a file needs to be fixed but does not fix the error(s).

Using chkdsk with open files

If you specify the **/f** switch, **chkdsk** shows an error if open files are found on the disk. If you do not specify the **/f** switch and open files exist, **chkdsk** might report lost allocation units on the disk. This could happen if open files have not yet been recorded in the file allocation table. If **chkdsk** reports the loss of a large number of allocation units, consider repairing the disk. To avoid problems caused by open files, avoid using **chkdsk** from another program or from an MS-DOS command interpreter started from Microsoft Windows.

Using chkdsk with assigned drives and networks

The **chkdsk** command does not work on drives formed by the **subst**, **join**, or **assign** command. You cannot use **chkdsk** to check a disk on a network drive.

Physical disk errors

The **chkdsk** command finds only logical errors in the file system, not physical disk errors. For information about recovering physically damaged files, see the **recover** command.

Bad disk sectors

Bad sectors reported by **chkdsk** were marked as "bad" when your disk was first prepared for operation. They pose no danger.

Saving a chkdsk status report to a file

You can save a **chkdsk** status report by redirecting the output to a file. Do not use the **/f** switch when you redirect **chkdsk** output to a file. For more information about redirection, see Chapter 7, "Advanced Command Techniques."

Examples

If you want to check the disk in drive A and have MS-DOS fix any errors encountered, type the following command:

```
chkdsk a: /f
```

Chkdsk pauses and displays messages if it encounters errors. Then **chkdsk** prompts you to specify whether you want MS-DOS to correct the errors. **Chkdsk** finishes by displaying a report showing the status of the disk.

To redirect the output of **chkdsk** to a file named STATUS, type the following command:

```
chkdsk a: > status
```

Because the output is redirected, MS-DOS does not repair errors it encounters during the check; but it records all the errors in a report file. Afterward, you can use **chkdsk** with the **/f** switch without redirection to correct any errors noted in the status report.

■ MS-DOS
□ Batch
□ CONFIG.SYS

■ Internal
□ External

■ Network

Cls

Clears the screen.

The cleared screen shows only the command prompt and cursor.

For examples of using **cls** in a batch-program menu system, see Chapter 10, "Working with Batch Programs."

Syntax

cls

■ MS-DOS
☐ Batch
☐ CONFIG.SYS
☐ Internal
■ External
☐ Network

Command

Starts a new instance of the MS-DOS command interpreter, COMMAND.COM.

A *command interpreter* is a program that displays the command prompt at which you type commands. Use the **exit** command to stop the new command interpreter and return control to the old one.

Syntax

command [[*drive*:]*path*] [*device*] [**/e:***nnnnn*] [**/p**] [**/c** *string*] [**/msg**]

In your CONFIG.SYS file, use the following syntax:

shell=[[*dos-drive*:]*dos-path*]**command.com** [[*drive*:]*path*] [*device*] **/e:***nnnn* **/p**

Parameters

[*drive*:]*path*

Specifies where the command interpreter is to look for the COMMAND.COM file when the transient part of the program needs to be reloaded. This parameter must be included if COMMAND.COM is not located in the root directory. This parameter is used to set the COMSPEC environment variable.

device

Specifies a different device for command input and output. For more information about this parameter, see the **ctty** command.

[*dos-drive*:]*dos-path*

Specifies the location of COMMAND.COM.

Switches

/e:*nnnnn*

Specifies the environment size, where *nnnnn* is the size in bytes. The value of *nnnnn* must be in the range 160 through 32768. MS-DOS rounds this number up to a multiple of 16 bytes. The default value is 256. For more information about environment size and the CONFIG.SYS file, see Chapter 11, "Customizing Your System."

/p Should be used only when **command** is used with the **shell** command in the CONFIG.SYS file. The **/p** switch makes the new copy of the command interpreter permanent. In this case, the **exit** command cannot be used to stop the command interpreter. If you specify **/p**, MS-DOS runs your AUTOEXEC.BAT batch program when it carries out the corresponding **shell** command.

/c *string*

Specifies that the command interpreter is to perform the command specified by *string* and then stop.

/msg

Specifies that all error messages should be stored in memory. Usually, some messages are stored only on disk. This switch is useful only if you are running MS-DOS from floppy disks. You must specify the **/p** switch when you use the **/msg** switch. For more information about using the **/msg** switch, see the following "Notes" section.

Notes

Limits on environment size

If *nnnnn* is less than 160 or greater than 32768, MS-DOS uses the default value of 256 bytes and displays the following message:

```
Parameter value not in allowed range.
```

Changing your terminal device

You can specify a different device (such as AUX) for input and output by using the *device* parameter. For more information about *device*, see the **ctty** command.

Running multiple command interpreters

When you start a new command interpreter, MS-DOS creates a new command environment. This new environment is a copy of the parent environment. You can change the new environment without affecting the old one. The default size of the new environment is 256 bytes or the size of the current environment rounded up to the next 16 bytes, whichever is larger. Use the **/e** switch to override the default size. (Note that the current environment refers to the memory actually being used, not to the environment size specified with the previous **/e** switch.)

Transient and resident memory

MS-DOS loads the command interpreter into memory in two parts: the transient part (in memory) and the resident part (on disk). Some programs write over the transient part of COMMAND.COM when they run. When this happens, the resident part must locate the COMMAND.COM file on disk to reload the transient part. The COMSPEC environment variable identifies where COMMAND.COM is located on the disk. If COMSPEC is set to a floppy disk drive, MS-DOS might prompt you to insert a disk that contains COMMAND.COM.

Using the /msg switch

Usually, MS-DOS keeps many error messages in the resident part of COMMAND.COM instead of using memory to store them. When MS-DOS needs to display one of these messages, MS-DOS retrieves the message from the disk containing COMMAND.COM.

If you are running MS-DOS from floppy disks instead of from a hard disk, MS-DOS cannot retrieve such error messages unless you have the disk containing COMMAND.COM in drive A. If this disk is not present, MS-DOS displays one of the following short messages instead of the full message:

```
Parse error

Extended error
```

You can make sure MS-DOS displays complete error messages by using the **/msg** switch with **command**. This switch forces MS-DOS to keep these error messages in memory so that they are always available when needed.

Use the **/msg** switch with **command** if you have a floppy disk system, unless you cannot afford to lose the memory used to store the error messages.

You must also specify the **/p** switch when you use the **/msg** switch.

Examples

The following command specifies that the MS-DOS command interpreter is to start a new command interpreter from the current program, run a batch program named MYBAT.BAT, and then return to the first command interpreter:

```
command /c mybat.bat
```

The following command specifies that COMMAND.COM is located in the DOS directory on drive C:

```
c:\dos\command.com /e:1024
```

Since the full path for **command** is specified, MS-DOS sets the COMSPEC environment variable to C:\DOS\COMMAND.COM. This command also creates an environment of 1024 bytes for this command interpreter.

Related Command

The **shell** command is the preferred method of using **command** to permanently increase space for the environment table. For more information about this alternative, see the **shell** command.

■ MS-DOS
☐ Batch
☐ CONFIG.SYS
☐ Internal
■ External
■ Network

Comp

Compares the contents of two files or sets of files byte by byte.

Comp can compare files on the same drive or on different drives, in the same directory or in different directories. As **comp** compares the files, it displays their locations and filenames.

Syntax

comp [*data1*] [*data2*] [**/d**] [**/a**] [**/l**] [**/n**=*number*] [**/c**]

Parameters

data1

Specifies the location and name of the first file or set of files you want to compare. You can use wildcards (* and ?) to specify multiple files.

data2

Specifies the location and name of the second file or set of files you want to compare. You can use wildcards (* and ?) to specify multiple files.

Switches

/d Displays differences in decimal format. (The default format is hexadecimal.)

/a Displays differences as characters.

/l Displays the number of the line on which a difference occurs, instead of displaying the byte offset.

/n=*number*
Compares the first *number* of lines of both files, even if the files are different sizes.

/c Performs a comparison that is not case-sensitive.

Notes

Comparing files with the same names

The files you want to compare can have the same filename, provided they are in different directories or on different drives. If you do not specify a filename for *data2*, the default filename for *data2* is the same as the filename in *data1*. You can use wildcards (* and ?) to specify filenames.

Special cases for *data1* and *data2*

If you omit necessary components of either *data1* or *data2* or if you omit *data2*, **comp** prompts you for the missing information. If *data1* contains only a drive letter or a directory name with no filename, the default filename for *data1* is *.*. Therefore, **comp** compares all the files in the specified directory to the file specified in *data2*. If *data2* contains only a drive letter or a directory name, the default filename for *data2* is the same as that in *data1*.

How the comp command identifies mismatching information

During the comparison, **comp** displays messages to identify the locations of unequal information in the two files. Each message indicates the offset memory address of the unequal bytes and the contents of the bytes themselves (in hexadecimal notation unless you specify the **/a** or **/d** switch). The message has the following format:

```
Compare error at OFFSET xxxxxxxx
file1 = xx
file2 = xx
```

After 10 unequal comparisons, **comp** stops comparing the files and displays the following message:

```
10 Mismatches - ending compare
```

Comparing files of different sizes

You cannot compare files of different sizes unless you specify the **/n** switch. If the file sizes are different, **comp** displays the following message:

```
Files are different sizes
Compare more files (Y/N)?
```

Press Y to compare another pair of files. Press N to stop the **comp** command.

If you press Y in response to the prompt, **comp** includes any switches you specified on the command line in every comparison it makes, until you press N or retype the command.

When comparing files of different sizes, use the **/n** switch to compare only the first portion of each file.

Comparing files sequentially

If you use wildcards to specify multiple files, **comp** finds the first file matching *data1* and compares it with the corresponding file in *data2*, if it exists. **Comp** reports the results of the comparison, then does the same for each file matching *data1*. When finished, **comp** displays the following message:

```
Compare more files (Y/N)?
```

To compare more files, press Y. **Comp** prompts you for the locations and names of the new files. To stop the comparisons, press N. When you press Y, **comp** prompts you for switches to use. If you don't specify any switches, **comp** uses the ones you specified before.

If comp cannot find the files

If **comp** cannot find the file(s) you specify, it prompts you with a message to determine whether you want to compare more files.

Related Commands

For information about comparing the contents of two floppy disks, see the **diskcomp** command.

For information about doing a complete comparison of two files of different sizes, see the **fc** command.

- ■ MS-DOS
- ☐ Batch
- ☐ CONFIG.SYS
- ■ Internal
- ☐ External
- ■ Network

Copy

Copies one or more files to another location.

This command can also be used to combine files. When more than one file is copied, MS-DOS displays each filename as the file is copied. For an introduction to the **copy** command, see Chapter 4, "Working with Files."

Syntax

copy [/a|/b] *source* [/a|/b] [+ *source* [/a|/b] [+ ...]] [*destination* [/a|/b]] [/v]

Parameters

source

Specifies the location and name of a file or set of files from which you want to copy. *Source* can consist of a drive letter and colon, a directory name, a filename, or a combination.

destination

Specifies the location and name of a file or set of files to which you want to copy. *Destination* can consist of a drive letter and colon, a directory name, a filename, or a combination.

Switches

/a Indicates an ASCII text file. When the **/a** switch precedes the list of filenames on the command line, it applies to all files whose names follow the **/a** switch, until **copy** encounters a **/b** switch, in which case the **/b** switch applies to the file whose name precedes the **/b** switch.

When the **/a** switch follows a filename, it applies to the file whose name precedes the **/a** switch and to all files whose names follow the **/a** switch, until **copy** encounters a **/b** switch, in which case the **/b** switch applies to the file whose name precedes the **/b** switch.

An ASCII text file can use an end-of-file character (CTRL+Z) to indicate the end of the file. When combining files, **copy** treats files as ASCII text files by default.

/b Indicates a binary file. When the **/b** switch precedes the list of filenames on the command line, it applies to all files whose names follow the **/b** switch, until **copy** encounters an **/a** switch, in which case the **/a** switch applies to the file whose name precedes the **/a** switch.

When the **/b** switch follows a filename, it applies to the file whose name precedes the **/b** switch and to all files whose names follow the **/b** switch, until **copy** encounters an **/a** switch, in which case the **/a** switch applies to the file whose name precedes the **/a** switch.

The **/b** switch specifies that the command interpreter is to read the number of bytes specified by the file size in the directory. The **/b** switch is the default value for **copy** unless **copy** is combining files.

/v Verifies that new files are written correctly.

Notes

Copying to and from devices

You can substitute a device name for one or more occurrences of *source* or for *destination*.

Using or omitting the /b switch when copying to a device

When *destination* is a device (for example, COM1 or LPT1), the **/b** switch causes MS-DOS to copy data to the device in binary mode. In binary mode, all characters (including such special characters as CTRL+C, CTRL+S, CTRL+Z, and carriage return) are copied to the device as data. Whereas, omission of the **/b** switch causes MS-DOS to copy data to the device in ASCII mode. In ASCII mode, such special characters as those previously listed may cause MS-DOS to take special action during the copying process.

Using the default destination file

If you do not specify a destination file, MS-DOS creates a copy with the same name, creation date, and creation time as the original file, placing the new copy in the current directory on the current drive. If the source file is on the current drive and in the current directory and you do not specify a different drive or directory for the destination file, the **copy** command stops and MS-DOS displays the following error message:

```
File cannot be copied onto itself
0 File(s) copied
```

Using the /v switch

If MS-DOS cannot verify a write operation, it displays an error message. Although recording errors rarely occur with the **copy** command, the **/v** switch lets you verify that critical data has been correctly recorded. The **/v** switch also slows down the **copy** command, because MS-DOS must check each sector recorded on the disk.

Using the /a and /b switches

The effect of an **/a** or **/b** switch depends upon its position on the command line. When the **/a** or **/b** switch follows the source filename, **copy** performs as shown in the following list:

/a Treats the file as an ASCII (text) file and copies data that precedes the first end-of-file character. **Copy** does not copy the first end-of-file character or the remainder of the file.

/b Copies the entire file, including any end-of-file character.

When the **/a** or **/b** switch follows the destination filename, **copy** performs as shown in the following list:

/a Adds an end-of-file character as the last character of the file.

/b Does not add an end-of-file character.

Combining files with the copy command

If you specify more than one *source*, separating entries with a plus sign (+), **copy** combines the files, creating a single file. If you use wildcards in *source* but specify a single filename in *destination*, **copy** combines all files matching the filename in *source* and creates a single file with the filename specified in *destination*.

In either case, **copy** assumes the combined files are ASCII files unless you specify the **/b** switch.

If the name of the destination file is the same as the name of one of the files being copied (except the first file), the original contents of the destination file are lost. When this happens, **copy** displays the following message:

```
Content of destination lost before copy
```

Copying files in subdirectories

To copy all of a directory's files and subdirectories, you should use the **xcopy** command.

Copying zero-length files

Copy does not copy files that are 0 bytes long. Use **xcopy** to copy these files.

Changing the time and date of a file

If you want to assign the current time and date to a file without modifying the file, use a command in the following format. The commas indicate the omission of the *destination* parameter.

```
copy /b source+,,
```

Examples

The following command copies a file and ensures that an end-of-file character is at the end of the copied file:

```
copy memo.doc letter.doc /a
```

To copy a file named ROBIN.TYP from the current drive and directory to an existing directory named BIRDS that is located on drive C, type the following command:

```
copy robin.typ c:\birds
```

If the BIRDS directory doesn't exist, MS-DOS copies the file ROBIN.TYP into a file named BIRDS that is located in the root directory on the disk in drive C.

To copy several files into one file, list any number of files as *source* parameters on the **copy** command line. Separate filenames with a plus sign (+) and specify a filename for the resulting combined file, as the following example shows:

```
copy mar89.rpt + apr89.rpt + may89.rpt report
```

This command combines the files named MAR89.RPT, APR89.RPT, and MAY89.RPT from the current drive and directory and places them in a file named REPORT in the current directory on the current drive. When files are combined, the destination file is created with the current date and time. If you omit *destination*, MS-DOS combines the files and stores them under the name of the first specified file. For example, if a file named REPORT already exists, you can use the following command to combine all four files in REPORT:

```
copy report + mar89.rpt + apr89.rpt + may89.rpt
```

You can also combine several files into one by using wildcards, as the following example shows:

```
copy *.txt combin.doc
```

This command combines all files in the current directory on the current drive that have the extension .TXT into one file named COMBIN.DOC, also in the current directory on the current drive.

If you want to combine several binary files into one by using wildcards, include the **/b** switch, as the following example shows:

```
copy /b *.exe combin.exe
```

This prevents MS-DOS from treating CTRL+Z as an end-of-file character.

CAUTION If you combine binary files, the resulting file might not be usable due to internal formatting.

In the following example, **copy** combines each file that has a .TXT extension with its corresponding .REF file. The result is a file with the same filename but with a .DOC extension. Thus, **copy** combines FILE1.TXT with FILE1.REF to form FILE1.DOC. Then **copy** combines FILE2.TXT with FILE2.REF to form FILE2.DOC, and so on.

```
copy *.txt + *.ref *.doc
```

The following **copy** command combines first all files with the .TXT extension, then all files with the .REF extension into one file named COMBIN.DOC:

```
copy *.txt + *.ref combin.doc
```

Related Command

For information about copying directories and subdirectories, see the **xcopy** command.

☐ MS-DOS
☐ Batch
■ CONFIG.SYS
☐ Internal
☐ External
☐ Network

Country

Enables MS-DOS to use international time, dates, currency, case conversions, and decimal separators.

The **country** command configures MS-DOS to recognize the character set and punctuation conventions observed when using one of the supported languages. For an introduction to using the **country** command and code pages, see Chapter 13, "Customizing for International Use."

Syntax

country=*xxx*[,[*yyy*][,[*drive*:][*path*]*filename*]]

Parameters

xxx Specifies the country code.

yyy Specifies the code page for the country.

[*drive*:][*path*]*filename*
Specifies the location and name of the file containing country information.

Notes

Changing default settings

MS-DOS uses the United States as the default setting. You can use the **country** command in your CONFIG.SYS file to change the setting.

If you do not specify the location and name of the file containing country-specific information, MS-DOS tries to find the COUNTRY.SYS file in the root directory of your startup drive.

Specifying supported languages

The following table lists each country or language supported by MS-DOS version 5.0. The table also lists the code pages you can use with each country code. For example, if you use country code 003, you can use only code page 437 or 850 for the *yyy* parameter. The first of the two code pages listed for each country or language is its default code page.

Country or language	Country code	Code pages
United States	001	437, 850
Canadian-French	002	863, 850
Latin America	003	850, 437
Netherlands	031	850, 437
Belgium	032	850, 437
France	033	850, 437
Spain	034	850, 437
Hungary	036	852, 850
Yugoslavia	038	852, 850
Italy	039	850, 437
Switzerland	041	850, 437
Czechoslovakia	042	852, 850
United Kingdom	044	437, 850
Denmark	045	850, 865
Sweden	046	850, 437
Norway	047	850, 865
Poland	048	852, 850
Germany	049	850, 437
Brazil	055	850, 437
International English	061	437, 850
Portugal	351	850, 860
Finland	358	850, 437

Code pages for the following countries or languages are also available with special versions of MS-DOS: Arabic, Israel, Japan, Korea, People's Republic of China, and Taiwan.

Specifying international time and date formats

The country code specifies the time and date formats used by the following MS-DOS commands: **backup**, **date**, **restore**, and **time**.

The following table lists the date and time formats related to each country code. For each country code, the "Date format" column shows how MS-DOS displays January 3, 1991, and the "Time format" column shows how MS-DOS displays 5:35 P.M. (with 0 seconds and 0 hundredths of a second).

Country or language	Country code	Date format	Time format
United States	001	01/03/1991	5:35:00.00p
Canadian-French	002	1991-01-03	17:35:00,00
Latin America	003	03/01/1991	5:35:00.00p
Netherlands	031	03-01-1991	17:35:00,00
Belgium	032	03/01/1991	17:35:00,00
France	033	03.01.1991	17:35:00,00
Spain	034	03/01/1991	17:35:00,00
Hungary	036	1991-01-03	17:35:00,00
Yugoslavia	038	1991-01-03	17:35:00,00
Italy	039	03/01/1991	17.35.00,00
Switzerland	041	03.01.1991	17,35,00.00
Czechoslovakia	042	1991-01-03	17:35:00,00
United Kingdom	044	03/01/1991	17:35:00.00
Denmark	045	03-01-1991	17.35.00,00
Sweden	046	1991-01-03	17.35.00,00
Norway	047	03.01.1991	17:35:00,00
Poland	048	1991-01-03	17:35:00,00
Germany	049	03.01.1991	17:35:00,00
Brazil	055	03/01/1991	17:35:00,00
International English	061	03/01/1991	17:35:00.00
Portugal	351	03-01-1991	17:35:00,00
Finland	358	03.01.1991	17.35.00,00

Code pages for the following countries or languages are also available with special versions of MS-DOS: Arabic, Israel, Japan, Korea, People's Republic of China, and Taiwan.

Examples

To convert international currency, time, date, and case to French conventions, add the following command to your CONFIG.SYS file:

```
country=033
```

For this example, assume that the COUNTRY.SYS file is in the root directory of the startup drive. If COUNTRY.SYS is in a different location, you specify the location in [*drive*:]*path* on the command line.

To specify a code page with the country code for France, type the following:

```
country=033,850
```

If you omit the code page but include the [*drive*:][*path*]*filename* parameter, you must still type the comma that would have preceded the code page, as the following example shows:

```
country=033,,c:\dos\country.sys
```

Related Commands

For information about changing characters and their arrangement on your keyboard, see the **keyb** command.

For information about preparing and selecting code pages, see the **mode** (set device code pages) command.

For information about loading country-specific information, see the **nlsfunc** command.

■ MS-DOS
☐ Batch
☐ CONFIG.SYS

■ Internal
☐ External

■ Network

Ctty

Changes the terminal device used to control your system.

Use the **ctty** command if you want to use another device to enter commands.

Syntax

ctty *device*

Parameter

device

Specifies the alternative device you want to use to type MS-DOS commands.

Notes

Using valid values for *device*

The valid values for the *device* parameter are: **prn**, **lpt1**, **lpt2**, **lpt3**, **con**, **aux**, **com1**, **com2**, **com3**, **com4**.

Setting up the serial port for ctty

Use the **mode** command to set up your serial port for baud rate, parity, bits, and stop bit before using the **ctty** command.

Using ctty with programs that do not use MS-DOS

Many programs do not use MS-DOS for input or output. These programs send input directly to the hardware on your computer. The **ctty** command has no effect on these programs; it affects only programs that use MS-DOS for reading keyboard input and displaying output.

Setting the terminal device with command

In addition to the **ctty** command, you can use the *device* parameter of the **command** command to specify the input device.

Examples

The following command changes control of all input and output from the current device (your computer screen and keyboard) to the AUX port:

```
ctty aux
```

In this example, a remote terminal device connected to the AUX port controls input and output for your system.

To transfer input and output back to the screen and keyboard, type the following command at the remote terminal:

```
ctty con
```

Related Commands

For more information about changing the input device when specifying a command interpreter, see the **command** command.

For more information about setting up the serial port, see the **mode** (configure serial port) command.

- [x] MS-DOS
- [ ] Batch
- [ ] CONFIG.SYS
- [x] Internal
- [ ] External
- [x] Network

Date

Displays the date or allows you to change the date from your terminal or from a batch program.

MS-DOS records the current date for each file you create or change; this date is listed next to the filename in the directory. For an introduction to the **date** command, see Chapter 2, "Command-Line Basics," and Chapter 11, "Customizing Your System."

Syntax

date [*mm-dd-yy*]

Parameter

mm-dd-yy

Sets the date you specify. Values for day, month, and year must be separated by periods (.), hyphens (-), or slash marks (/). The date format depends on the **country** setting you are using in your CONFIG.SYS file. The following list shows the valid values for the month, day, and year portions of the *mm-dd-yy* parameter.

mm	1 through 12		
dd	1 through 31		
yy	80 through 99	or	1980 through 2099

Notes

Adjusting for days in a month

MS-DOS is programmed to change months and years correctly, whether the month has 28, 29, 30, or 31 days.

Using the date command in your AUTOEXEC.BAT file

When you use an AUTOEXEC.BAT file, MS-DOS does not automatically display a prompt for a date when you start your system. To prompt users for the date every time the system is restarted, include the **date** command in AUTOEXEC.BAT. For more information about using the AUTOEXEC.BAT file, see Chapter 11, "Customizing Your System."

Changing the date format

It is possible to change the *mm-dd-yy* format to display the date in other formats. You can add the **country** command to your CONFIG.SYS file to change the date format to the European standard (*dd-mm-yy*) or to the Scientific International (Metric) format (*yy-mm-dd*). For information about changing the date format, see Chapter 13, "Customizing for International Use."

Related Command

For information about changing the current time, see the **time** command.

■ MS-DOS
☐ Batch
☐ CONFIG.SYS

☐ Internal
■ External

☐ Network

Debug

Starts Debug, a program that allows you to test and debug executable files.

Syntax

debug [[*drive*:][*path*]*filename* [*testfile-parameters*]]

Parameters

[*drive*:][*path*]*filename*

Specifies the location and name of the executable file you want to test.

testfile-parameters

Specifies any command-line information required by the executable file you want to test.

Notes

Using the debug command without specifying a file to be tested

If you use the **debug** command without a location and filename, you then type all Debug commands in response to the Debug prompt, a hyphen (-).

Debug commands

The following is a list of Debug commands:

? Displays a list of the **debug** commands.

a Assembles 8086/8087/8088 mnemonics.

c Compares two portions of memory.

d Displays the contents of a portion of memory.

e Enters data into memory starting at a specified address.

f Fills a range of memory with specified values.

g Runs the executable file that is in memory.

h Performs hexadecimal arithmetic.

i Displays one byte value from a specified port.

l Loads the contents of a file or disk sectors into memory.

m Copies the contents of a block of memory.

n Specifies a file for an **l** or **w** command, or specifies the parameters for the file you are testing.

o Sends one byte value to an output port.

p Executes a loop, a repeated string instruction, a software interrupt, or a subroutine.

q Stops the Debug session.

r Displays or alters the contents of one or more registers.

s Searches a portion of memory for a specified pattern of one or more byte values.

t Executes one instruction and then displays the contents of all registers, the status of all flags, and the decoded form of the instruction that Debug will execute next.

u Disassembles bytes and displays the corresponding source statements.

w Writes the file being tested to a disk.

xa Allocates expanded memory.

xd Deallocates expanded memory.

xm Maps expanded memory pages.

xs Displays the status of expanded memory.

For descriptions of these commands, see the following pages.

Separating command parameters

All Debug commands accept parameters, except the **q** command. You can separate parameters with commas or spaces, but these separators are required only between two hexadecimal values. Therefore, the following commands are equivalent:

```
dcs:100 110

d cs:100 110

d,cs:100,110
```

Specifying valid address entries

An *address* parameter in a Debug command specifies a location in memory. *Address* is a two-part designation containing either an alphabetic segment register or a 4-digit segment address, plus an offset value. You can omit the segment register or segment address. The default segment for the **a**, **g**, **l**, **t**, **u**, and **w** commands is CS. The default segment for all other commands is DS. All numeric values are in hexadecimal format.

The following are valid addresses:

```
CS:0100

04BA:0100
```

The colon between the segment name and the offset value is required.

Specifying valid range entries

A *range* parameter in a Debug command specifies a range of memory. You can choose from two formats for *range*: a starting address and an ending address, or a starting address and the length (denoted by **l**) of the range.

For example, both of the following syntaxes specify a 16-byte range beginning at CS:100:

```
cs:100 10f

cs:100 l 10
```

- ■ MS-DOS
- ☐ Batch
- ☐ CONFIG.SYS
- ☐ Internal
- ■ External
- ☐ Network

Debug: A (Assemble)

Assembles 8086/8087/8088 mnemonics directly into memory.

This command creates executable machine code from assembly-language statements. All numeric values are in hexadecimal format, and you must type them as 1 to 4 characters. You specify a prefix mnemonic in front of the operation code (opcode) to which it refers.

Syntax

a [*address*]

Parameter

address

Specifies the location where you type assembly-language mnemonics. You use hexadecimal values for *address* and type each value without the trailing "h" character. If you do not specify an address, **a** starts assembling where it last stopped.

Notes

Using mnemonics

The segment-override mnemonics are **cs:**, **ds:**, **es:**, and **ss:**. The mnemonic for the far return is **retf**. String-manipulation mnemonics must explicitly state the string size. For example, use **movsw** to move word strings (16 bits), and use **movsb** to move byte strings (8 bits).

Assembling jumps and calls

The assembler automatically assembles a short, near, or far jump or call, depending on byte displacement, to the destination address. You can override such a jump or call by using a **near** or **far** prefix, as the following example shows:

```
-a0100:0500
0100:0500 jmp  502        ; a 2-byte short jump
0100:0502 jmp  near 505   ; a 3-byte near jump
0100:0505 jmp  far  50a   ; a 5-byte far jump
```

You can abbreviate the **near** prefix to **ne**.

Distinguishing word and byte memory locations

When an operand can refer to either a word memory location or a byte memory location, you must specify the data type with the prefix **word ptr** or the prefix **byte ptr**. Acceptable abbreviations are **wo** and **by**, respectively. The following example shows the two formats:

```
dec     wo [si]

neg     byte ptr [128]
```

Specifying operands

Debug uses the common convention that an operand enclosed in brackets ([ ]) refers to a memory location. This is because Debug cannot otherwise differentiate between an immediate operand and an operand that is a memory location. The following example shows the two formats:

```
mov     ax,21     ; load AX with 21h
mov     ax,[21]   ; load AX with the
                  ; contents of
                  ; memory location 21h
```

Using pseudoinstructions

Two popular pseudoinstructions are available with the **a** command: the **db** opcode, which assembles byte values directly into memory, and the **dw** opcode, which assembles word values directly into memory. Following are examples of both pseudoinstructions:

```
db      1,2,3,4,"THIS IS AN EXAMPLE"
db      'THIS IS A QUOTATION MARK: "'
db      "THIS IS A QUOTATION MARK: '"

dw      1000,2000,3000,"BACH"
```

Examples

The **a** command supports all forms of register-indirect commands, as the following example shows:

```
add     bx,34[bp+2].[si-1]
pop     [bp+di]
push    [si]
```

All opcode synonyms are also supported, as the following example shows:

```
loopz   100
loope   100

ja      200
jnbe    200
```

For 8087 opcodes, you must specify the **wait** or **fwait** prefix, as the following example shows:

```
fwait fadd st,st(3)        ; this line assembles
                           ; an fwait prefix
```

Related Commands

For information about entering data into specific bytes, see the Debug **e** (enter) command.

For information about disassembling bytes, see the Debug **u** (unassemble) command.

■ MS-DOS
☐ Batch
☐ CONFIG.SYS

☐ Internal
■ External

☐ Network

Debug: C (Compare)

Compares two portions of memory.

Syntax

c *range address*

Parameters

range

Specifies the starting and ending addresses, or the starting address and length, of the first area of memory you want to compare. For information about valid *range* values, see the **debug** command.

address

Specifies the starting address of the second area of memory you want to compare. For information about valid *address* values, see the **debug** command.

Note

If the *range* and *address* memory areas are identical, Debug displays nothing and returns directly to the Debug prompt. If there are differences, Debug displays them in the following format:

address1 byte1 byte2 address2

See the following example for an illustration of this format.

Example

The following commands have the same effect:

```
c100,10f 300

c100l10 300
```

Each command compares the block of memory from 100h through 10Fh with the block of memory from 300h through 30Fh.

Debug responds to either of the previous commands with a display similar to the following (assuming DS = 197F):

```
197F:0100 4D E4 197F:0300
197F:0101 67 99 197F:0301
197F:0102 A3 27 197F:0302
197F:0103 35 F3 197F:0303
197F:0104 97 BD 197F:0304
197F:0105 04 35 197F:0305
197F:0107 76 71 197F:0307
197F:0108 E6 11 197F:0308
197F:0109 19 2C 197F:0309
197F:010A 80 0A 197F:030A
197F:010B 36 7F 197F:030B
197F:010C BE 22 197F:030C
197F:010D 83 93 197F:030D
197F:010E 49 77 197F:030E
197F:010F 4F 8A 197F:030F
```

Notice that the addresses 197F:0106 and 197F:0306 are missing from the list. This means that the values in those addresses are identical.

■ MS-DOS
☐ Batch
☐ CONFIG.SYS
☐ Internal
■ External
☐ Network

Debug: D (Dump)

Displays the contents of a range of memory addresses.

Syntax

d [*range*]

Parameter

range

Specifies the starting and ending addresses, or the starting address and length, of the memory area whose contents you want to display. For information about valid *range* values, see the **debug** command. If you do not specify *range*, Debug displays the contents of 128 bytes, starting at the end of the address range specified in the previous **d** command.

Note

When you use the **d** command, Debug displays memory contents in two portions: a hexadecimal portion (each byte value is shown in hexadecimal format) and an ASCII portion (each byte value is shown as an ASCII character). Each nonprinting character is denoted by a period (.) in the ASCII portion of the display. Each display line shows the contents of 16 bytes, with a hyphen between the eighth and ninth bytes. Each display line begins on a 16-byte boundary.

Examples

Suppose you type the following command:

```
dcs:100 10f
```

Debug displays the the contents of the range in the following format:

```
04BA:0100 54 4F 4D 00 53 41 57 59-45 52 00 00 00 00 00 00 TOM.SAWYER......
```

If you type the **d** command without parameters, Debug formats the display as described in the previous example. Each line of the display begins with an address that is 16 bytes greater than the address on the previous line (or 8 bytes if you have a 40-column screen).

For each subsequent **d** command you type without parameters, Debug displays the bytes immediately following those last displayed.

If you type the following command, Debug displays the contents of 20h bytes, starting at CS:100:

```
dcs:100 l 20
```

If you type the following command, Debug displays the contents of all bytes in the range of lines from 100h through 115h in the CS segment:

```
dcs:100 115
```

Related Commands

For information about displaying the contents of registers, see the Debug **r** (register) command.

For information about disassembling bytes, see the Debug **u** (unassemble) command.

- ■ MS-DOS
- ☐ Batch
- ☐ CONFIG.SYS
- ☐ Internal
- ■ External
- ☐ Network

Debug: E (Enter)

Enters data into memory at the address you specify.

You can type data in either hexadecimal or ASCII format. Any data previously stored at the specified address is lost.

Syntax

e *address* [*list*]

Parameters

address

Specifies the first memory location where you want to enter data.

list

Specifies the data you want to enter into successive bytes of memory.

Notes

Using the *address* parameter

If you specify a value for *address* without specifying a value for the optional *list* parameter, Debug displays the address and its contents, repeats the address on the next line, and waits for your input. At this point, you can perform one of the following actions:

- Replace the byte value. To do this, you type a new value after the current value. If the value you type is not a valid hexadecimal value or if it contains more than two digits, Debug does not echo the invalid or extra character.
- Advance to the next byte. To do this, you press the SPACEBAR. To change the value in that byte, type a new value after the current value. If you move beyond an 8-byte boundary when you press the SPACEBAR, Debug starts a new display line and displays the new address at the beginning of the line.
- Return to the preceding byte. To do this, you press the HYPHEN key. You can press the HYPHEN key repeatedly to move back more than 1 byte. When you press HYPHEN, Debug starts a new line and displays the current address and byte value.
- Stop the **e** command. To do this, you press the ENTER key. You can press ENTER at any byte position.

Using the *list* parameter

If you specify values for the *list* parameter, the **e** command sequentially replaces the existing byte values with the values from the list. If an error occurs, no byte values are changed.

List values can be either hexadecimal byte values or strings. You separate values by using a space, a comma, or a tab character. You must enclose strings within single or double quotation marks.

Examples

Suppose you type the following command:

```
ecs:100
```

Debug displays the contents of the first byte in the following format:

```
04BA:0100 EB._
```

To change this value to 41, type **41** at the cursor, as follows:

```
04BA:0100 EB.41_
```

You can type consecutive byte values with one **e** command. Instead of pressing ENTER after typing the new value, press the SPACEBAR. Debug displays the next value. In this example, if you press the SPACEBAR three times, Debug displays the following values:

```
04BA:0100 EB.41  10. 00. BC._
```

To change the hexadecimal value BC to 42, type **42** at the cursor, as follows:

```
04BA:0100 EB.41  10. 00. BC.42_
```

Now suppose that you decide the value 10 should be 6F. To correct this value, press the HYPHEN key twice to return to address 0101 (value 10). Debug displays the following:

```
04BA:0100 EB.41  10. 00. BC.42-
04BA:0102  00.-
04BA:0101  10._
```

Type **6f** at the cursor to change the value, as follows:

```
04BA:0101  10.6f_
```

Press ENTER to stop the **e** command and return to the Debug prompt.

The following is an example of a string entry:

```
eds:100 "This is the text example"
```

This string will fill 24 bytes, starting at DS:100.

Related Commands

For information about assembling mnemonics, see the Debug **a** (assemble) command.

For information about displaying the contents of a portion of memory, see the Debug **d** (dump) command.

■ MS-DOS
☐ Batch
☐ CONFIG.SYS

☐ Internal
■ External

☐ Network

Debug: F (Fill)

Fills addresses in the specified memory area with values you specify.

You can specify data in either hexadecimal or ASCII format. Any data you previously stored at the specified address is lost.

Syntax

f *range list*

Parameters

range

Specifies the starting and ending addresses, or the starting address and length, of the memory area you want to fill. For information about valid *range* values, see the **debug** command.

list

Specifies the data you want to enter. *List* can consist of hexadecimal numbers or a string enclosed in quotation marks.

Notes

Using the *range* parameter

If *range* contains more bytes than the number of values in *list*, Debug assigns the values in *list* repeatedly until all bytes in *range* are filled.

If any of the memory in *range* is bad or doesn't exist, Debug displays an error message and stops the **f** command.

Using the *list* parameter

If *list* contains more values than the number of bytes in *range*, Debug ignores the extra values in *list*.

Example

Suppose you type the following command:

```
f04ba:100l100 42 45 52 54 41
```

In response, Debug fills memory locations 04BA:100 through 04BA:1FF with the values specified. Debug repeats the five values until all the 100h bytes are filled.

■ MS-DOS
☐ Batch
☐ CONFIG.SYS

☐ Internal
■ External

☐ Network

Debug: G (Go)

Runs the program currently in memory.

Syntax

g [=*address*] [*breakpoints*]

Parameters

=*address*

Specifies the address in the program currently in memory at which you want execution to begin. If you do not specify *address*, MS-DOS begins program execution at the current address in the CS:IP registers.

breakpoints

Specifies 1 to 10 temporary breakpoints that you can set as part of the **g** command.

Notes

Using the *address* parameter

You must precede the *address* parameter with an equal sign (=) to distinguish the starting address (*address*) from the breakpoint addresses (*breakpoints*).

Specifying breakpoints

The program stops at the first breakpoint it encounters, regardless of where you typed that breakpoint in the *breakpoints* list. Debug replaces the original instruction at each breakpoint with an interrupt code.

When the program reaches a breakpoint, Debug restores all breakpoint addresses to their original instructions and displays the contents of all registers, the status of all flags, and the decoded form of the last instruction executed. Debug displays the same information as it would display if you used the Debug **r** (register) command and specified the breakpoint address.

If you do not stop the program at one of the breakpoints, Debug does not replace the interrupt codes with the original instructions.

Limitations on setting breakpoints

You can set breakpoints only at addresses containing the first byte of an 8086 operation code (opcode). If you set more than 10 breakpoints, Debug displays the following message:

```
bp Error
```

Requirements for the user stack pointer

The user stack pointer must be valid and must have 6 bytes available for the **g** command. This command uses an **iret** instruction to jump to the program being tested. Debug sets the user stack pointer and pushes the user flags, the code segment register, and the instruction pointer onto the user stack. (If the user stack is not valid or is too small, the operating system might fail.) Debug places an interrupt code (0CCh) at the specified breakpoint address(es).

Restarting a program

Do not attempt to restart a program after MS-DOS displays the following message:

```
Program terminated normally
```

To run the program properly, you must reload it by using the Debug **n** (name) and **l** (load) commands.

Examples

Suppose you type the following command:

```
gcs:7550
```

MS-DOS runs the program currently in memory up to the breakpoint address 7550 in the CS segment. Debug then displays the contents of the registers and the status of the flags and stops the **g** command.

The following command sets two breakpoints:

```
gcs:7550, cs:8000
```

If you type the **g** command again after Debug encounters a breakpoint, execution begins at the instruction after the breakpoint, rather than at the usual starting address.

Related Commands

For information about executing a loop, a repeated string instruction, a software interrupt, or a subroutine, see the Debug **p** (proceed) command.

For information about executing one instruction, see the Debug **t** (trace) command.

■ MS-DOS
☐ Batch
☐ CONFIG.SYS
☐ Internal
■ External
☐ Network

Debug: H (Hex)

Performs hexadecimal arithmetic on two parameters you specify.

Syntax

h *value1 value2*

Parameters

value1

Represents any hexadecimal number in the range 0 through FFFFh.

value2

Represents a second hexadecimal number in the range 0 through FFFFh.

Note

Debug first adds the two parameters you specify and then subtracts the second parameter from the first. The results of these calculations are displayed on one line — first the sum, then the difference.

Example

Suppose you type the following command:

```
h19f 10a
```

Debug performs the calculations and displays the following result:

```
02A9 0095
```

- ■ MS-DOS
- ☐ Batch
- ☐ CONFIG.SYS
- ☐ Internal
- ■ External
- ☐ Network

Debug: I (Input)

Reads and displays one byte value from the port you specify.

Syntax

i *port*

Parameter

port

Specifies the input port by address. The address can be a 16-bit value.

Example

Suppose you type the following command:

```
i2f8
```

Suppose also that the byte value at the port is 42h. Debug reads the byte and then displays the value, as follows:

```
42
```

Related Command

For information about sending the value of a byte to an output port, see the Debug **o** (output) command.

- ■ MS-DOS
- ☐ Batch
- ☐ CONFIG.SYS
- ☐ Internal
- ■ External
- ☐ Network

Debug: L (Load)

Loads a file or contents of specific disk sectors into memory.

Syntax

To load the contents of the number of bytes specified in the BX:CX registers from a disk file, use the following syntax:

l [*address*]

To bypass the MS-DOS file system and directly load specific sectors, use the following syntax:

l *address drive start number*

Parameters

address

Specifies the memory location where you want to load the file or the sector contents. If you do not specify *address*, Debug uses the current address in the CS register.

drive

Specifies the drive that contains the disk from which specific sectors are to be read. This value is numeric: 0 = A, 1 = B, 2 = C, and so on. You use the *drive*, *start*, and *number* parameters only if you want to load the contents of specific sectors rather than load the file specified on the **debug** command line or in the most recent Debug **n** (name) command.

start

Specifies the hexadecimal number of the first sector whose contents you want to load.

number

Specifies the hexadecimal number of consecutive sectors whose contents you want to load.

Notes

Using the l command without parameters

When you use the **l** command without parameters, the file you specified on the **debug** command line is loaded into memory, beginning at address CS:100. Debug also sets the BX and CX registers to the number of bytes loaded. If you did not specify a file on the **debug** command line, the file loaded is the one you most recently specified by using the **n** command.

Using the l command with the *address* parameter

If you use the **l** command with the *address* parameter, Debug begins loading the file or the contents of the specified sectors at the memory location *address*.

Using the l command with all parameters

If you use the **l** command with all parameters, Debug loads the contents of specific disk sectors instead of loading a file.

Loading the contents of specific sectors

Each sector in the range you specify is read from *drive*. Debug begins loading with *start* and continues until the contents of the number of sectors specified in *number* have been loaded.

Loading an .EXE file

Debug ignores the *address* parameter for .EXE files. If you specify an .EXE file, Debug relocates the file to the loading address specified in the header of the .EXE file. The header itself is stripped off the .EXE file before the file is loaded into memory, so the size of an .EXE file on disk differs from its size in memory. If you want to examine a complete .EXE file, rename the file with a different extension.

Opening a hex file

A *hex file* is a file that uses the Intel hexadecimal format, as described in *The MS-DOS Encyclopedia*. Debug assumes that files with the .HEX extension are hexadecimal-format files. You can type the **l** command with no parameters to load a hex file beginning at the address specified in the hex file. If the **l** command you type includes the *address* parameter, Debug adds the specified address to the address found in the hex file to determine the starting address.

Examples

Suppose you start Debug and type the following command:

```
nfile.com
```

You can now type the **l** command to load FILE.COM. Debug loads the file and displays the Debug prompt.

Suppose that you want to load the contents of 109 (6Dh) sectors from drive C, beginning with logical sector 15 (0Fh), into memory beginning at address 04BA:0100. To do this, type the following command:

```
l04ba:100 2 0f 6d
```

Related Commands

For information about specifying a file for the **l** command, see the Debug **n** (name) command.

For information about writing the file being debugged to a disk, see the Debug **w** (write) command.

- ■ MS-DOS
- ☐ Batch
- ☐ CONFIG.SYS
- ☐ Internal
- ■ External
- ☐ Network

Debug: M (Move)

Copies the contents of a block of memory to another block of memory.

Syntax

m *range address*

Parameters

range

Specifies the starting and ending addresses, or the starting address and the length, of the memory area whose contents you want to copy.

address

Specifies the starting address of the location to which you want to copy the contents of *range*.

Notes

Effects of the copy operation on existing data

If the addresses in the block being copied do not have new data written to them, the original data remains intact. However, if the destination block already contains data (as it might in an *overlapping copy operation*), that data is overwritten. (Overlapping copy operations are those in which part of the destination block overlaps part of the source block.)

Performing overlapping copy operations

The **m** command performs overlapping copy operations without losing data at the destination addresses. The contents of addresses that will be overwritten are copied first. Thus, if data is to be copied from higher addresses to lower addresses, the copy operation begins at the source block's lowest address and progresses toward the highest address. Conversely, if data is to be copied from lower addresses to higher addresses, the copy operation begins at the source block's highest address and progresses toward the lowest address.

Example

Suppose you type the following command:

```
mcs:100 110 cs:500
```

Debug first copies the contents of address CS:110 to CS:510, then copies the contents of CS:10F to CS:50F, and so on until it has copied the contents of CS:100 to CS:500. To view the results, you can use the Debug **d** (dump) command, specifying the destination address you used with the **m** command.

- ■ MS-DOS
- ☐ Batch
- ☐ CONFIG.SYS
- ☐ Internal
- ■ External
- ☐ Network

Debug: N (Name)

Specifies the name of an executable file for a Debug **l** (load) or **w** (write) command, or specifies parameters for the executable file being debugged.

Syntax

n [*drive*:][*path*]*filename*

To specify parameters for the executable file you are testing, use the following syntax:

n *file-parameters*

To clear the current specifications, use the following syntax:

n

Parameters

[*drive*:][*path*]*filename*

Specifies the location and name of the executable file you want to test.

file-parameters

Specifies parameters and switches for the executable file you are testing.

Notes

The two uses of the n command

You can use the **n** command in two ways. First, you can use it to specify a file to be used by a later **l** or **w** command. If you start Debug without naming a file to be debugged, you must use the command **n***filename* before you can use the **l** command to load the file. The filename is correctly formatted for a file control block at CS:5C. Second, you can use the **n** command to specify command-line parameters and switches for the file being debugged.

Memory areas

The following four areas of memory can be affected by the **n** command:

Memory location	Contents
CS:5C	File control block (FCB) for file 1
CS:6C	File control block (FCB) for file 2
CS:80	Length of **n** command line (in characters)
CS:81	Beginning of **n** command-line characters

The first filename you specify for the **n** command is placed in a file control block (FCB) at CS:5C. If you specify a second filename, this name is placed in an FCB at CS:6C. The number of characters typed on the **n** command line (exclusive of the first character, **n**) is stored at location CS:80. The actual characters on the **n** command line (again, exclusive of the letter **n**) are stored beginning at CS:81. Note that these characters can be any switches and delimiters that would be legal in a command typed at the MS-DOS prompt.

Examples

Suppose you've started Debug and loaded the program PROG.COM for debugging. You subsequently decide to specify two parameters for PROG.COM and run the program. Following is the sequence of commands for this example:

```
debug prog.com
nparam1 param2
g
```

In this case, the Debug **g** (go) command runs the program as if you had typed the following command at the MS-DOS prompt:

```
prog param1 param2
```

Testing and debugging therefore reflect a typical run-time environment for PROG.COM.

In the following sequence of commands, the first **n** command specifies FILE1.EXE as the file for the subsequent **l** command, which loads FILE1.EXE into memory. The second **n** command specifies the parameters to be used by FILE1.EXE. Finally, the **g** command runs FILE1.EXE as if you had typed **file1 file2.dat file3.dat** at the MS-DOS prompt.

```
nfile1.exe
l
nfile2.dat file3.dat
g
```

Note that you do not use the **l** command after the second form of the **n** command. Also note that if you now use the **w** command, MS-DOS saves FILE1.EXE, the file being debugged, with the name FILE2.DAT. To avoid this result, you should always use the first form of the **n** command immediately before either an **l** or a **w** command.

Related Commands

For information about loading the contents of a file or of specific disk sectors into memory, see the Debug **l** command.

For information about writing the file being debugged to a disk, see the Debug **w** command.

- ■ MS-DOS
- ☐ Batch
- ☐ CONFIG.SYS
- ☐ Internal
- ■ External
- ☐ Network

Debug: O (Output)

Sends the value of a byte to an output port.

Syntax

o *port byte-value*

Parameters

port

Specifies the output port by address. The port address can be a 16-bit value.

byte-value

Specifies the byte value you want to direct to *port*.

Example

To send the byte value 4Fh to the output port at address 2F8h, type the following command:

```
o2f8 4f
```

Related Command

For information about reading the value of a byte from an input port, see the Debug **i** (input) command.

■ MS-DOS
☐ Batch
☐ CONFIG.SYS
☐ Internal
■ External
☐ Network

Debug: P (Proceed)

Executes a loop, a repeated string instruction, a software interrupt, or a subroutine; or traces through any other instruction.

Syntax

p [=*address*] [*number*]

Parameters

=*address*

Specifies the location of the first instruction to execute. If you do not specify an address, the default address is the current address specified in the CS:IP registers.

number

Specifies the number of instructions to execute before returning control to Debug. The default value is 1.

Notes

Transferring control to the program being tested

When the **p** command transfers control from Debug to the program being tested, that program runs without interruption until the loop, repeated string instruction, software interrupt, or subroutine at the specified address is completed, or until the specified number of machine instructions have been executed. Control then returns to Debug.

Limitations on the *address* parameter

If the *address* parameter does not specify a segment, Debug uses the CS register of the program being tested. If you omit *address*, the program is executed beginning at the address specified by its CS:IP registers. You must precede the *address* parameter with an equal sign (=) to distinguish it from the *number* parameter. If the instruction at the specified address is not a loop, a repeated string instruction, a software interrupt, or a subroutine, the **p** command works the same way as the Debug **t** (trace) command.

Messages displayed with the p command

After **p** executes an instruction, Debug displays the contents of the program's registers, the status of its flags, and the decoded form of the next instruction to be executed.

CAUTION You cannot use the **p** command to trace through read-only memory (ROM).

Example

Suppose that the program you're testing contains a **call** instruction at address CS:143F. To run the subroutine that is the destination of **call** and then return control to Debug, type the following command:

```
p=143f
```

Debug displays the results in the following format:

```
AX=0000  BX=0000  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=2246  ES=2246  SS=2246  CS=2246  IP=1443   NV UP EI PL NZ AC PO NC
2246:1442 7505          JNZ     144A
```

Related Commands

For information about running the program currently in memory, see the Debug **g** (go) command.

For information about executing one instruction, see the Debug **t** command.

- [■] MS-DOS
- [ ] Batch
- [ ] CONFIG.SYS
- [ ] Internal
- [■] External
- [ ] Network

Debug: Q (Quit)

Stops the Debug session, without saving the file currently being tested.

After you type **q**, control returns to MS-DOS.

Syntax

q

Parameter

This command takes no parameters.

Example

To stop the debugging session, type the following command:

```
q
```

MS-DOS displays the MS-DOS prompt.

Related Command

For information about saving a file, see the Debug **w** (write) command.

■ MS-DOS
☐ Batch
☐ CONFIG.SYS
☐ Internal
■ External
☐ Network

Debug: R (Register)

Displays or alters the contents of one or more central-processing-unit (CPU) registers.

Syntax

r [*register-name*]

To display the contents of all registers and flags in the register storage area, use the following syntax:

r

Parameter

register-name

Specifies the name of the register whose contents you want to display.

Notes

Using the r command

If you specify a register name, MS-DOS displays the 16-bit value of that register in hexadecimal notation and displays a colon as the prompt. If you want to change the value contained in the register, type a new value and press ENTER; otherwise, just press ENTER to return to the Debug prompt.

Valid register names

The following are valid values for *register-name*: **ax**, **bx**, **cx**, **dx**, **sp**, **bp**, **si**, **di**, **ds**, **es**, **ss**, **cs**, **ip**, **pc**, and **f**. Both **ip** and **pc** refer to the instruction pointer.

If you specify a register name other than one from the preceding list, MS-DOS displays the following message:

```
br error
```

Using the f character instead of a register name

If you type the **f** character instead of a register name, Debug displays the current setting of each flag as a two-letter code and then displays the Debug prompt. To change the setting of a flag, type the appropriate two-letter code from the following table:

Flag name	Set	Clear
Overflow	**ov**	**nv**
Direction	**dn** (decrement)	**up** (increment)
Interrupt	**ei** (enabled)	**di** (disabled)
Sign	**ng** (negative)	**pl** (positive)
Zero	**zr**	**nz**
Auxiliary Carry	**ac**	**na**
Parity	**pe** (even)	**po** (odd)
Carry	**cy**	**nc**

You can type new flag values in any order. You need not leave spaces between these values. To stop the **r** command, press ENTER. Any flags for which you did not specify new values remain unchanged.

Messages displayed with the r command

If you specify more than one value for a flag, Debug displays the following message:

```
df error
```

If you specify a flag code not listed in the preceding table, Debug displays the following message:

```
bf error
```

In both cases, Debug ignores all settings specified after the invalid entry.

Default settings for Debug

When you start Debug, the segment registers are set to the bottom of free memory, the instruction pointer is set to 0100h, all flags are cleared, and the remaining registers are set to zero, except for **sp**, which is set to FFEEh.

Examples

To view the contents of all registers, the status of all flags, and the decoded form of the instruction at the current location, type the following command:

```
r
```

If the current location is CS:11A, the display will look similar to the following:

```
AX=0E00 BX=00FF CX=0007 DX=01FF SP=039D BP=0000 SI=005C DI=0000
DS=04BA ES=04BA SS=04BA CS=04BA IP=011A  NV UP DI NG NZ AC PE NC
04BA:011A  CD21          INT     21
```

To view only the status of the flags, type the following command:

```
rf
```

Debug displays the information in the following format:

```
NV UP DI NG NZ AC PE NC - _
```

Now you can type one or more valid flag values, in any order, with or without spaces, as in the following command:

```
nv up di ng nz ac pe nc - pleicy
```

Debug stops the **r** command and displays the Debug prompt. To see the changes, type either the **r** or **rf** command. Debug then displays the following:

```
NV UP EI PL NZ AC PE CY - _
```

Press ENTER to return to the Debug prompt.

Related Commands

For information about displaying the contents of a portion of memory, see the Debug **d** (dump) command.

For information about disassembling bytes, see the Debug **u** (unassemble) command.

- ■ MS-DOS
- ☐ Batch
- ☐ CONFIG.SYS
- ☐ Internal
- ■ External
- ☐ Network

Debug: S (Search)

Searches a range of addresses for a pattern of one or more byte values.

Syntax

s *range list*

Parameters

range

Specifies the beginning and ending addresses of the range you want to search. For information about valid values for the *range* parameter, see the **debug** command.

list

Specifies the pattern of one or more byte values or a string you want to search for. Separate each byte value from the next with a space or a comma. Enclose string values in quotation marks.

Note

If the *list* parameter contains more than one byte value, Debug displays only the first address where the byte value occurs. If *list* contains only one byte value, Debug displays all addresses where the value occurs in the specified range.

Examples

Suppose you want to find all addresses in the range CS:100 through CS:110 that contain the value 41. To do this, type the following command:

```
scs:100 110 41
```

Debug displays the results in the following format:

```
04BA:0104
04BA:010D
-
```

The following command searches for the string "Ph" in the range CS:100 through CS:1A0:

```
scs:100 1a0 "Ph"
```

- [■] MS-DOS
- [☐] Batch
- [☐] CONFIG.SYS
- [☐] Internal
- [■] External
- [☐] Network

Debug: T (Trace)

Executes one instruction and displays the contents of all registers, the status of all flags, and the decoded form of the instruction executed.

Syntax

t [=*address*] [*number*]

Parameters

=*address*

Specifies the address at which Debug is to start tracing instructions. If you omit the *address* parameter, tracing begins at the address specified by your program's CS:IP registers. For information about valid values for the *address* parameter, see the **debug** command.

number

Specifies the number of instructions to be traced. This value must be a hexadecimal number. The default value is 1.

Notes

Tracing instructions in read-only memory

The **t** command uses the hardware trace mode of the 8086 or 8088 microprocessor. Therefore, you can also trace instructions stored in read-only memory (ROM).

Using the *address* parameter

You must precede the *address* parameter with an equal sign (=) to distinguish it from the *number* parameter.

Example

To execute one instruction (pointed to by CS:IP), and then display the contents of the registers, the status of the flags, and the decoded form of the instruction, type the following command:

```
t
```

If the position of the instruction in the program were 04BA:011A, Debug might display the following information:

```
AX=0E00 BX=00FF CX=0007 DX=01FF SP=039D BP=0000 SI=005C DI=0000
DS=04BA ES=04BA SS=04BA CS=04BA IP=011A  NV UP DI NG NZ AC PE NC
04BA:011A  CD21          INT     21
```

Related Commands

For information about executing a loop, a repeated string instruction, a software interrupt, or a subroutine, see the Debug **p** (proceed) command.

For information about executing the program currently in memory, see the Debug **g** (go) command.

- ■ MS-DOS
- ☐ Batch
- ☐ CONFIG.SYS
- ☐ Internal
- ■ External
- ☐ Network

Debug: U (Unassemble)

Disassembles bytes and displays their corresponding source statements, including addresses and byte values. The disassembled code looks like a listing for an assembled file.

Syntax

u [*range*]

To disassemble 20h bytes (the default number), beginning at the first address after the address displayed by the previous **u** command, use the following syntax:

u

Parameter

range

Specifies the starting and ending addresses, or the starting address and length, of the code you want to disassemble. For information about valid values for the *range* parameter, see the **debug** command.

Examples

To disassemble 16 (10h) bytes, beginning at address 04BA:0100, type the following command:

```
u04ba:100l10
```

Debug displays the results in the following format:

```
04BA:0100  206472    AND  [SI+72],AH
04BA:0103  69        DB   69
04BA:0104  7665      JBE  016B
04BA:0106  207370    AND  [BP+DI+70],DH
04BA:0109  65        DB   65
04BA:010A  63        DB   63
04BA:010B  69        DB   69
04BA:010C  66        DB   66
04BA:010D  69        DB   69
04BA:010E  63        DB   63
04BA:010F  61        DB   61
```

To display only the information for the specific addresses 04BA:0100 through 04BA:0108, type the following command:

```
u04ba:0100 0108
```

Debug displays the following:

```
04BA:0100  206472    AND  [SI+72],AH
04BA:0103  69        DB   69
04BA:0104  7665      JBE  016B
04BA:0106  207370    AND  [BP+DI+70],DH
```

Related Commands

For information about assembling mnemonics, see the Debug **a** (assemble) command.

For information about displaying the contents of a portion of memory, see the Debug **d** (dump) command.

■ MS-DOS
☐ Batch
☐ CONFIG.SYS
☐ Internal
■ External
☐ Network

Debug: W (Write)

Writes a file or specific sectors to disk.

You must have specified the name of the disk file when you started Debug or in the most recent Debug **n** (name) command. Both of these methods properly format a filename for a file control block at address CS:5C.

Syntax

To write the contents of the number of bytes specified in the BX:CX registers to a disk file, use the following syntax:

w [*address*]

To bypass the MS-DOS file system and directly write specific sectors, use the following syntax:

w *address drive start number*

CAUTION Writing specific sectors is extremely risky because it bypasses the MS-DOS file handler. The disk's file structure can easily be damaged if the wrong values are typed.

Parameters

address

Specifies the beginning memory address of the file, or portion of the file, you want to write to a disk file. If you do not specify *address*, Debug starts from CS:100. For information about valid values for the *address* parameter, see the **debug** command.

drive

Specifies the drive that contains the destination disk. This value is numeric: 0 = A, 1 = B, 2 = C, and so on.

start

Specifies the hexadecimal number of the first sector to which you want to write.

number

Specifies the number of sectors to which you want to write.

Notes

Resetting BX:CX before using the w command without parameters
If you have used a Debug **g** (go), **t** (trace), **p** (proceed), or **r** (register) command, you must reset the BX:CX registers before using the **w** command without parameters.

Writing a modified file to a disk
If you modify the file but do not change the name, length, or starting address, Debug can still correctly write the file to the original disk location.

Limitation on the w command
You cannot write an .EXE or .HEX file with this command.

Example

Suppose you want to write the contents of memory, beginning at the address CS:100, to the disk in drive B. You want the data to begin in the disk's logical sector number 37h and continue for 2Bh sectors. To do this, type the following command:

```
wcs:100 1 37 2b
```

When the write operation is complete, Debug displays the Debug prompt again.

Related Commands

For information about specifying a file for the **w** command, see the Debug **n** (name) command.

For information about loading the contents of a file or file sectors into memory, see the Debug **l** (load) command.

■ MS-DOS
☐ Batch
☐ CONFIG.SYS
☐ Internal
■ External
☐ Network

Debug: XA (Allocate Expanded Memory)

Allocates a specified number of pages of expanded memory.

To use expanded memory, you must have installed an expanded-memory device driver that conforms to version 4.0 of the Lotus/Intel/Microsoft Expanded Memory Specification (LIM EMS). For an introduction to device drivers, see Chapter 15, "Device Drivers."

Syntax

xa [*count*]

Parameter

count

Specifies the number of 16-kilobyte pages of expanded memory to allocate.

Note

If the specified number of pages is available, Debug displays a message indicating the hexadecimal number of the handle created; otherwise, Debug displays an error message.

Example

To allocate 8 pages of expanded memory, type the following command:

```
xa8
```

If the command is successful, Debug displays a message similar to the following:

```
Handle created=0003
```

Related Commands

For information about other Debug commands that work with expanded memory, see the Debug commands **xd** (deallocate expanded memory), **xm** (map expanded-memory pages), and **xs** (display expanded-memory status).

■ MS-DOS
☐ Batch
☐ CONFIG.SYS
☐ Internal
■ External
☐ Network

Debug: XD (Deallocate Expanded Memory)

Deallocates a handle to expanded memory.

To use expanded memory, you must have installed an expanded-memory device driver that conforms to version 4.0 of the Lotus/Intel/Microsoft Expanded Memory Specification (LIM EMS). For an introduction to device drivers, see Chapter 15, "Device Drivers."

Syntax

xd [*handle*]

Parameter

handle

Specifies the handle you want to deallocate.

Example

To deallocate handle 0003, type the following command:

```
xd 0003
```

If the command is successful, Debug displays the following message:

```
Handle 0003 deallocated
```

Related Commands

For information about other Debug commands that work with expanded memory, see the Debug commands **xa** (allocate expanded memory), **xm** (map expanded-memory pages), and **xs** (display expanded-memory status).

■ MS-DOS
☐ Batch
☐ CONFIG.SYS
☐ Internal
■ External
☐ Network

Debug: XM (Map Expanded Memory Pages)

Maps a logical page of expanded memory, belonging to the specified handle, to a physical page of expanded memory.

To use expanded memory, you must have installed an expanded-memory device driver that conforms to version 4.0 of the Lotus/Intel/Microsoft Expanded Memory Specification (LIM EMS). For an introduction to device drivers, see Chapter 15, "Device Drivers."

Syntax

xm [*lpage*] [*ppage*] [*handle*]

Parameters

lpage

Specifies the number of the logical page of expanded memory that you want to map to physical page *ppage*.

ppage

Specifies the number of the physical page to which *lpage* is to be mapped.

handle

Specifies the handle.

Example

To map logical page 5 of handle 0003 to physical page 2, type the following command:

```
xm 5 2 0003
```

If the command is successful, Debug displays the following message:

```
Logical page 05 mapped to physical page 02
```

Related Commands

For information about other Debug commands that work with expanded memory, see the Debug commands **xa** (allocate expanded memory), **xd** (deallocate expanded memory), and **xs** (display expanded-memory status).

- ■ MS-DOS
- ☐ Batch
- ☐ CONFIG.SYS
- ☐ Internal
- ■ External
- ☐ Network

Debug: XS (Display Expanded-Memory Status)

Displays information about the status of expanded memory.

To use expanded memory, you must have installed an expanded-memory device driver that conforms to version 4.0 of the Lotus/Intel/Microsoft Expanded Memory Specification (LIM EMS). For an introduction to device drivers, see Chapter 15, "Device Drivers."

Syntax

xs

Parameter

This command takes no parameters.

Note

The information that Debug displays has the following format:

```
Handle xx has xx pages allocated
Physical page xx = Frame segment xx
    xx of a total xx EMS pages have been allocated
    xx of a total xx EMS handles have been allocated
```

Example

To display expanded-memory information, type the following command:

```
xs
```

Debug displays information similar to the following:

```
Handle 0000 has 0000 pages allocated
Handle 0001 has 0002 pages allocated

Physical page 00 = Frame segment C000
Physical page 01 = Frame segment C400
Physical page 02 = Frame segment C800
Physical page 03 = Frame segment CC00
   2 of a total 80 EMS pages have been allocated
   2 of a total FF EMS handles have been allocated
```

Related Commands

For information about other Debug commands that work with expanded memory, see the Debug commands **xa** (allocate expanded memory), **xd** (deallocate expanded memory), and **xm** (map expanded-memory pages).

- ■ MS-DOS
- ☐ Batch
- ☐ CONFIG.SYS
- ■ Internal
- ☐ External
- ■ Network

Del (Erase)

Deletes specified files.

For an introduction to the **del** command, see Chapter 4, "Working with Files."

Syntax

del [*drive*:][*path*]*filename* [**/p**]

erase [*drive*:][*path*]*filename* [**/p**]

Parameter

[*drive*:][*path*]*filename*

Specifies the location and name of the file or set of files you want to delete.

Switch

/p Prompts you for confirmation before deleting the specified file.

Notes

Using the /p switch

If you use the **/p** switch, **del** displays the name of a file and prompts you with a message in the following format:

```
filename, Delete (Y/N)?
```

Press Y to confirm the deletion, N to cancel the deletion and display the next filename (if you specified a group of files), or CRTL+C to stop the **del** command.

Deleting more than one file at a time

You can delete all the files in a directory by typing the **del** command followed by [*drive*:]*path*. You can also use wildcards (* and ?) to delete more than one file at a time. However, you should use wildcards cautiously with the **del** command to avoid deleting files unintentionally. Suppose you type the following command:

```
del *.*
```

Del displays the following prompt:

```
All files in directory will be deleted!
Are you sure (Y/N)?
```

Press Y and then ENTER to delete all files in the current directory, or press N and then ENTER to cancel the deletion.

Before you use wildcards with the **del** command to delete a group of files, you can use the same wildcards with the **dir** command to see a list of the names of all the files included in the group.

CAUTION Once you delete a file from your disk, you may not be able to retrieve it. Although the **undelete** command can retrieve deleted files, it can do so with certainty only if no other files have been created or changed on the disk. If you accidentally delete a file that you want to keep, stop what you are doing and immediately use the **undelete** command to retrieve the file.

Examples

To delete all the files in a directory named TEST on drive C, you can use either of the following commands:

```
del c:\test

del c:\test\*.*
```

Related Commands

For information about retrieving a deleted file, see the **mirror** and **undelete** commands.

For information about removing a directory, see the **rmdir** command.

☐ MS-DOS
☐ Batch
■ CONFIG.SYS
☐ Internal
☐ External
☐ Network

Device

Loads into memory the device driver you specify.

For an introduction to using the CONFIG.SYS file, see Chapter 11, "Customizing Your System."

Syntax

device=[*drive*:][*path*]*filename* [*dd-parameters*]

Parameters

[*drive*:][*path*]*filename*

Specifies the location and name of the device driver you want to load.

[*dd-parameters*]

Specifies any command-line information required by the device driver.

Notes

Using standard device drivers

The standard installable device drivers provided with MS-DOS are ANSI.SYS, DISPLAY.SYS, DRIVER.SYS, EGA.SYS, PRINTER.SYS, RAMDRIVE.SYS, EMM386.EXE, HIMEM.SYS, and SMARTDRV.SYS. For more information about these installable device drivers, see Chapter 11, "Customizing Your System," and Chapter 15, "Device Drivers."

The files COUNTRY.SYS and KEYBOARD.SYS are not device drivers. MS-DOS loads these files whenever necessary. Do not try to load either of these files with the **device** command. If you do, your system locks up and you cannot restart MS-DOS. For information about loading COUNTRY.SYS, see the **country** command. For information about loading KEYBOARD.SYS, see the **keyb** command.

Installing device drivers for other products

When you purchase a mouse, a scanner, or a similar product, the manufacturer usually includes device-driver software. To install a device driver, specify its location and name on a **device** command line.

Installing a third-party console driver

If you install both DISPLAY.SYS and a third-party console driver, such as VT52.SYS, the third-party device driver must be installed first. Otherwise, the third-party device driver may disable DISPLAY.SYS.

Example

If you plan to use an ANSI escape sequence to control the screen and keyboard, you should add the following command to your CONFIG.SYS file (assuming DOS files are in the DOS directory on drive C):

```
device=c:\dos\ansi.sys
```

Related Command

For information about loading device drivers into the upper memory area, see the **devicehigh** command.

☐ MS-DOS
☐ Batch
■ CONFIG.SYS
☐ Internal
☐ External
☐ Network

Devicehigh

Loads device drivers into the upper memory area.

Loading a device driver into the upper memory area frees more bytes of conventional memory for other programs. For an introduction to the **devicehigh** command and the upper memory area, see Chapter 12, "Optimizing Your System."

Syntax

devicehigh=[*drive*:][*path*]*filename* [*dd-parameters*]

To specify the minimum amount of memory that must be available before **devicehigh** attempts to load a device driver into the upper memory area, use the following syntax:

devicehigh size=*hexsize* [*drive*:][*path*]*filename* [*dd-parameters*]

Parameters

[*drive*:][*path*]*filename*

Specifies the location and name of the device driver you want to load into the upper memory area.

dd-parameters

Specifies any command-line information required by the device driver.

hexsize

Specifies the minimum amount of memory (the number of bytes, in hexadecimal format) that must be available before **devicehigh** attempts to load a device driver into the upper memory area. You must use both **size** and *hexsize*, as shown in the second syntax line.

Notes

Using the dos=umb command

To use the **devicehigh** command, you must also include the **dos=umb** command in your CONFIG.SYS file. If you do not specify this command, all device drivers are loaded into conventional memory, as if you had used the **device** command. For more information about the **umb** switch, see the **dos** command.

Installing HIMEM.SYS and a UMB provider

Before you can load a device driver into the upper memory area, you must use the **device** command once to install the HIMEM.SYS device driver and then again to install an upper-memory-block (UMB) provider. These commands must appear before the **devicehigh** command in your CONFIG.SYS file. If your computer has an 80386 or 80486 processor, you can use EMM386.EXE as the UMB provider. If your computer has a different processor, you must supply a different UMB provider. On some computers, you might even be able to use HIMEM.SYS itself as the UMB provider. For more information about memory management and the **device** command, see Chapter 12, "Optimizing Your System."

Specifying a size limit

If the device driver you specify on the **devicehigh** command line attempts to allocate more buffer space than is available in a block of the upper memory area, your system may lock up. You can try using the *hexsize* parameter to avoid this problem. In *hexsize*, indicate, in hexadecimal format, the amount of memory the device driver needs. To find this value for a particular device driver, load the driver into conventional memory and use the **mem /debug** command. This method is usually, but not always, effective.

If no upper memory area is available

If there is not enough upper memory area available to load the device driver you specified with the **devicehigh** command, MS-DOS will load it into conventional memory (as if you had used the **device** command).

Example

If you include the following commands in your CONFIG.SYS file, MS-DOS attempts to load a device driver named MYDRIV.SYS into the upper memory area of an 80386 computer:

```
device=c:\dos\himem.sys
dos=umb
device=c:\dos\emm386.exe
devicehigh=mydriv.sys
```

Related Commands

For information about loading programs into the upper memory area, see the **loadhigh** command.

For information about loading device drivers into conventional memory, see the **device** command.

■ MS-DOS
☐ Batch
☐ CONFIG.SYS
■ Internal
☐ External
■ Network

Dir

Displays a list of a directory's files and subdirectories.

When you use **dir** without parameters or switches, it displays the disk's volume label and serial number; one directory or filename per line, including the filename extension, the file size in bytes, and the date and time the file was last modified; and the total number of files listed, their cumulative size, and the free space (in bytes) remaining on the disk. For an introduction to the **dir** command, see Chapter 4, "Working with Files."

Syntax

dir [*drive*:][*path*][*filename*] [**/p**] [**/w**] [**/a**[[:]*attributes*]] [**/o**[[:]*sortorder*]] [**/s**] [**/b**] [**/l**]

Parameters

[*drive*:][*path*]

Specifies the drive and directory for which you want to see a listing.

[*filename*]

Specifies a particular file or group of files for which you want to see a listing.

Switches

/p Displays one screen of the listing at a time. To see the next screen, press any key.

/w Displays the listing in wide format, with as many as five filenames or directory names on each line.

/a[[:] *attributes*]

Displays only the names of those directories and files with the attributes you specify. If you omit this switch, **dir** displays the names of all files except hidden and system files. If you use this switch without specifying *attributes*, **dir** displays the names of all files, including hidden and system files. The following list describes each of the values you can use for *attributes*. The colon (**:**) is optional. Use any combination of these values, and do not separate the values with spaces.

h	Hidden files
–h	Files that are not hidden
s	System files
–s	Files other than system files
d	Directories
–d	Files only (not directories)
a	Files ready for archiving (backup)
–a	Files that have not changed since the last backup
r	Read-only files
–r	Files that are not read-only

/o[[**:**] *sortorder*]

Controls the order in which **dir** sorts and displays directory names and filenames. If you omit this switch, **dir** displays the names in the order in which they occur in the directory. If you use this switch without specifying *sortorder*, **dir** displays the names of the directories, sorted in alphabetic order, and then displays the names of files, sorted in alphabetic order. The colon (**:**) is optional. The following list describes each of the values you can use for *sortorder*. Use any combination of the values, and do not separate these values with spaces.

n	In alphabetic order by name
–n	In reverse alphabetic order by name (Z through A)
e	In alphabetic order by extension
–e	In reverse alphabetic order by extension (Z through A)
d	By date and time, earliest first
–d	By date and time, latest first
s	By size, smallest first
–s	By size, largest first
g	With directories grouped before files
–g	With directories grouped after files

/s Lists every occurrence, in the specified directory and all subdirectories, of the specified filename.

/b Lists each directory name or filename, one per line (including the filename extension). This switch displays no heading information and no summary. The **/b** switch overrides the **/w** switch.

/l Displays unsorted directory names and filenames in lowercase. This switch does not convert extended characters to lowercase.

Notes

Using wildcards with dir

You can use wildcards (**?** and ***) to display a listing of a subset of files and subdirectories. For an example illustrating the use of a wildcard, see the following "Examples" section.

Specifying file display attributes

If you specify the **/a** switch with more than one value in *attributes*, **dir** displays the names of only those files with all the specified attributes. For example, if you specify the **/a** switch with the **r** and **–h** values for *attributes* by using either **/a:r–h** or **/ar–h**, **dir** displays only the names of read-only files that are not hidden.

Specifying filename sorting

If you specify more than one *sortorder* value, **dir** sorts the filenames by the first criterion first, then by the second criterion, and so on. For example, if you specify the **/o** switch with the **e** and **–s** values for *sortorder* by using either **/o:e–s** or **/oe–s**, **dir** sorts the names of directories and files by extension, with the largest first, and displays the final result. The alphabetic sorting by extension causes filenames with no extensions to appear first, then directory names, then filenames with extensions.

Setting date and time formats

The date and time formats used by **dir** depend on the **country** setting you use in your CONFIG.SYS file. If you don't use the **country** command, the formats are those for the United States. For information about the **country** command, see Chapter 13, "Customizing for International Use."

Using redirection symbols and pipes

When you use a redirection symbol (>) to send **dir** output to a file or a pipe (¦) to send **dir** output to another command, use the **/a:-d** and **/b** switches to list only the filenames. You can use the *filename* parameter with the **/b** and **/s** switches to specify that **dir** is to search the current directory and its subdirectories for all filenames that match *filename*. **Dir** lists only the drive letter, directory name, filename, and filename extension, one path per line, for each filename it finds.

Before using a pipe for redirection, you should set the TEMP environment variable in your AUTOEXEC.BAT file. For information about using the redirection and pipe symbols, see Chapter 7, "Advanced Command Techniques."

Presetting dir parameters and switches

You can preset **dir** parameters and switches by including the **set** command with the DIRCMD environment variable in your AUTOEXEC.BAT file. You can use any valid combination of **dir** parameters and switches with the **set DIRCMD** command, including the location and name of a file.

For example, to use the DIRCMD environment variable to set the wide display format (**/w**) as the default format, include the following command in your AUTOEXEC.BAT file:

```
set dircmd=/w
```

For a single use of the **dir** command, you can override a switch set by using the DIRCMD environment variable. To do so, you use the same switch on the **dir** command line, but you must also precede the switch letter with a minus sign, as the following example shows:

```
dir /-w
```

You can change the DIRCMD default settings by typing the **set** command at the command prompt with a new parameter or switch after the equal sign (=). The new default settings are effective for all subsequent **dir** commands until you use **set DIRCMD** again on the command line or until you restart MS-DOS.

To clear all default settings, type the following command:

```
set dircmd=
```

You can view the current settings of the DIRCMD environment variable by typing the following command:

```
set
```

MS-DOS displays a list of environment variables and their settings. For more information about setting environment variables, see the **set** command.

Examples

Suppose you want **dir** to display one directory listing after another, until it has displayed the listing for every directory on the disk in the current drive. Suppose also that you want **dir** to alphabetize each directory listing, display it in wide format, and pause after each screen. To specify such a display, be sure the root directory is the current directory and then type the following command:

```
dir /s/w/o/p
```

Dir lists the name of the root directory, the names of the subdirectories of the root directory, and the names of the files in the root directory (including extensions). Then **dir** lists the subdirectory names and filenames in each subdirectory in the directory tree.

To alter the preceding example so that **dir** displays the filenames and extensions but omits the directory names, type the following command:

```
dir /s/w/o/p/a:-d
```

To print a directory listing, type the redirection symbol and **prn** after any form of the **dir** command, as the following example shows:

```
dir > prn
```

When you specify **prn** on the **dir** command line, the directory listing is sent to the printer attached to the LPT1 port. If your printer is attached to a different port, you must replace **prn** with the name of the correct port.

You can also redirect output of the **dir** command to a file by replacing **prn** with a filename. A path is also accepted on the command line. For example, to direct **dir** output to the file DIR.DOC in the RECORDS directory, type the following command:

```
dir > \records\dir.doc
```

If DIR.DOC does not exist, MS-DOS creates it, unless the directory RECORDS also does not exist. In that case, MS-DOS displays the following message:

```
File creation error
```

To display a list of all the filenames with the .TXT extension in all directories on drive C, type the following command:

```
dir c:\*.txt /w/o/s/p
```

Dir displays, in wide format, an alphabetized list of the matching filenames in each directory and pauses each time the screen fills, until you press a key to continue.

Related Command

For information about displaying the directory structure of a path or disk, see the **tree** command.

- ■ MS-DOS
- ☐ Batch
- ☐ CONFIG.SYS
- ☐ Internal
- ■ External
- ☐ Network

Diskcomp

Compares the contents of two floppy disks.

This command performs a track-by-track comparison. **Diskcomp** determines the number of sides and sectors per track to compare based on the format of the first disk you specify.

Syntax

diskcomp [*drive1*: [*drive2*:]] [**/1**] [**/8**]

Parameters

drive1:

Specifies the drive containing one of the floppy disks.

drive2:

Specifies the drive containing the other floppy disk.

Switches

/1 Compares only the first sides of the disks, even if the disks are double-sided and the drives can read double-sided disks.

/8 Compares only the first 8 sectors per track, even if the disks contain 9 or 15 sectors per track.

Notes

Invalid drive for diskcomp

The **diskcomp** command works only with floppy disks. You cannot use **diskcomp** with a hard disk. If you specify a hard disk drive for *drive1* or *drive2*, **diskcomp** displays the following error message:

```
Invalid drive specification
Specified drive does not exist
or is non-removable
```

Diskcomp messages

If all tracks on the two disks being compared are the same, **diskcomp** displays the following message:

```
Compare OK
```

If the tracks are not the same, **diskcomp** displays a message similar to the following:

```
Compare error on
side 1, track 2
```

When **diskcomp** completes the comparison, it displays the following message:

```
Compare another diskette (Y/N)?
```

If you press Y, **diskcomp** prompts you to insert disks for the next comparison. If you press N, **diskcomp** stops the comparison.

Diskcomp ignores a disk's volume number when it makes the comparison.

Omitting drive parameters

If you omit the *drive2* parameter, **diskcomp** uses the current drive for *drive2*.If you omit both drive parameters, **diskcomp** uses the current drive for both. If the current drive is the same as *drive1*, **diskcomp** prompts you to swap disks as necessary.

Using one drive for the comparison

If you specify the same floppy disk drive for *drive1* and *drive2*, **diskcomp** does a comparison by using one drive and prompts you to insert the disks as necessary. You might have to swap the disks more than once, depending on the capacity of the disks and the amount of available memory.

Comparing different types of disks

Diskcomp cannot compare a single-sided disk with a double-sided disk, nor a high-density disk with a double-density disk. If the disk in *drive1* is not of the same type as the disk in *drive2*, **diskcomp** displays the following message:

```
Drive types or diskette types not compatible
```

Using diskcomp with networks and redirected drives

Diskcomp does not work on a network drive or on a drive created or affected by an **assign**, **join**, or **subst** command. If you attempt to use **diskcomp** with a drive of any of these types, **diskcomp** displays an error message.

Comparing an original disk with a copy

When you use **diskcomp** with a disk that you made with the **copy** command, **diskcomp** may display a message similar to the following:

```
Compare error on
side 0, track 0
```

This type of error can occur even if the files on the disks are identical. Although the **copy** command duplicates information, it doesn't necessarily place it in the same location on the destination disk. For more information about comparing individual files on two disks, see the **fc** command.

Diskcomp exit codes

The following list shows each exit code and a brief description of its meaning:

0 The disks are the same.

1 Differences were found.

2 The user pressed CTRL+C to stop the process.

3 A hard error occurred.

4 An initialization error occurred.

You can use the **errorlevel** parameter on the **if** command line in a batch program to process exit codes returned by **diskcomp**. For an example of a batch program that processes exit codes, see the following "Examples" section. For more information about batch programs, see Chapter 10, "Working with Batch Programs."

Examples

If your system has only one floppy disk drive, drive A, and you want to compare two disks, type the following command:

```
diskcomp a: a:
```

Diskcomp prompts you to insert each disk, as required.

Following is an example of a batch program that uses the **errorlevel** parameter on the **if** command line to process a **diskcomp** exit code:

```
rem CHECKOUT.BAT compares the disks in drive A and B
echo off
diskcomp a: b:
if errorlevel 4 goto ini_error
if errorlevel 3 goto hard_error
if errorlevel 2 goto break
if errorlevel 1 goto no_compare
if errorlevel 0 goto compare_ok
:ini_error
echo ERROR: Insufficient memory or command invalid
goto exit
:hard_error
echo ERROR: An irrecoverable error occurred
goto exit
:break
echo You just pressed CTRL+C to stop the comparison
goto exit
:no_compare
echo Disks are not the same
goto exit
:compare_ok
echo The comparison was successful; the disks are the same
goto exit
:exit
```

Related Commands

For information about comparing two files, see the **comp** and **fc** commands.

- ■ MS-DOS
- ☐ Batch
- ☐ CONFIG.SYS
- ☐ Internal
- ■ External
- ☐ Network

Diskcopy

Copies the contents of the floppy disk in the source drive to a formatted or unformatted floppy disk in the destination drive. **Diskcopy** destroys the existing contents of the destination disk as it copies the new information to it.

This command determines the number of sides to copy based on the source drive and disk.

Syntax

diskcopy [*drive1*: [*drive2*:]] [**/1**] [**/v**]

Parameters

drive1:

Specifies the drive containing the source disk.

drive2:

Specifies the drive containing the destination disk.

Switches

/1 Copies only the first side of a disk.

/v Verifies that the information is copied correctly. Use of this switch slows the copying process.

Notes

Invalid drive for diskcopy

The **diskcopy** command works only with removable disks, such as floppy disks. You cannot use **diskcopy** with a hard disk. If you specify a hard disk drive for *drive1* or *drive2*, **diskcopy** displays the following error message:

```
Invalid drive specification
Specified drive does not exist
or is non-removable
```

Diskcopy messages

The **diskcopy** command prompts you to insert the source and destination disks and waits for you to press any key before continuing.

After copying, **diskcopy** displays the following message:

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

If you press Y, **diskcopy** prompts you to insert source and destination disks for the next copy operation. To stop the **diskcopy** process, press N.

If you are copying to an unformatted floppy disk in *drive2*, **diskcopy** formats the disk with the same number of sides and sectors per track as are on the disk in *drive1*. **Diskcopy** displays the following message while it formats the disk and copies the files:

```
Formatting while copying
```

If the capacity of the source disk is greater than that of the destination disk and your computer can detect this difference, **diskcopy** displays the following message:

```
TARGET media has lower capacity than SOURCE
Continue anyway (Y/N)?
```

If you press Y, **diskcopy** attempts to format the destination disk and copy the files.

Disk serial numbers

If the source disk has a volume serial number, **diskcopy** creates a new volume serial number for the destination disk and displays the number when the copy operation is complete.

Omitting drive parameters

If you omit the *drive2* parameter, **diskcopy** uses the current drive as the destination drive. If you omit both drive parameters, **diskcopy** uses the current drive for both. If the current drive is the same as *drive1*, **diskcopy** prompts you to swap disks as necessary.

Using one drive for copying

If *drive1* and *drive2* are the same, **diskcopy** prompts you whenever you should switch disks. If you omit both drive parameters and the current disk drive is a floppy disk drive, **diskcopy** prompts you each time you should insert a disk in the drive. If the disks contain more information than available memory can hold, **diskcopy** cannot read all of the information at once. **Diskcopy** reads from the source disk, writes to the destination disk, and prompts you to insert the source disk again. This process continues until the entire disk has been copied.

Avoiding disk fragmentation

Because **diskcopy** makes an exact copy of the source disk on the destination disk, any *fragmentation* on the source disk is transferred to the destination disk. Fragmentation is the presence of small areas of unused disk space between existing files on a disk.

A fragmented source disk can slow down the finding, reading, or writing of files. To avoid transferring fragmentation from one disk to another, use either the **copy** command or the **xcopy** command to copy your disk. Because **copy** and **xcopy** copy files sequentially, the new disk is not fragmented.

CAUTION You cannot use **xcopy** to copy a startup disk.

Diskcopy exit codes

The following list briefly describes the meaning of each **diskcopy** exit code (**errorlevel**):

0 The copy operation was successful.

1 A nonfatal read/write error occurred.

2 The user pressed CTRL+C to stop the process.

3 A fatal hard error occurred.

4 An initialization error occurred.

You can use the **errorlevel** parameter on the **if** command line in a batch program to process exit codes returned by **diskcopy**. For an example of a batch program that processes exit codes, see the **diskcomp** command. For more information about batch programs, see Chapter 10, "Working with Batch Programs."

Related Commands

For information about copying one or more files, see the **copy** command.

For information about copying directories and subdirectories, see the **xcopy** command.

☐ MS-DOS
☐ Batch
■ CONFIG.SYS
☐ Internal
☐ External
☐ Network

Dos

Specifies that MS-DOS is to maintain a link to the upper memory area or is to load part of itself into the high memory area (HMA).

For an introduction to using the **dos** command and the upper memory area, see Chapter 12, "Optimizing Your System."

Syntax

dos=high|low[,umb|,noumb]

dos=[high,|low,]umb|noumb

Parameters

umb|noumb

Specifies whether MS-DOS should maintain a link between conventional memory and the upper memory area. The **umb** parameter provides this link. The **noumb** parameter disconnects this link. The default setting is **noumb**.

high|low

Specifies whether MS-DOS should attempt to load a part of itself into the HMA. Use the **high** parameter to enable MS-DOS to load itself into the HMA. Use the **low** parameter to keep all of MS-DOS in conventional memory. The default setting is **low**.

Notes

Must install HIMEM.SYS for dos=umb or dos=high

You must install the HIMEM.SYS device driver before you specify either **dos=umb** or **dos=high**. For more information, see Chapter 12, "Optimizing Your System."

Using the umb parameter

You must specify the **dos=umb** command in order to load programs and device drivers into the upper memory area. Using the upper memory area frees more space in conventional memory for programs. In addition to using this command, you must install an upper-memory-block (UMB) provider. For more information about UMB providers, see Chapter 12, "Optimizing Your System."

Using the high parameter

If you specify the **high** parameter, MS-DOS attempts to load part of itself into the HMA. Loading part of MS-DOS into the HMA frees conventional memory for programs.

Combining parameters

You can include more than one parameter on a single **dos** command line, using commas to separate them. For example, the following command lines are valid:

```
dos=umb,low

dos=high,umb
```

You can place the **dos** command anywhere in your CONFIG.SYS file.

Related Commands

For information about loading a device driver into the upper memory area, see the **devicehigh** command.

For information about loading a program into the upper memory area, see the **loadhigh** command.

■ MS-DOS
☐ Batch
☐ CONFIG.SYS

☐ Internal
■ External

■ Network

Doskey

Starts the Doskey program, which recalls MS-DOS commands, edits command lines, and creates macros.

The Doskey program is a terminate-and-stay-resident program. You can use Doskey to customize and automate MS-DOS command lines. When installed, Doskey occupies about 3 kilobytes of resident memory. For an introduction to the Doskey program, see Chapter 7, "Advanced Command Techniques."

Syntax

doskey [**/reinstall**] [**/bufsize**=*size*] [**/macros**] [**/history**] [**/insert**|**/overstrike**] [*macroname*=[text]]

To start the Doskey program and use the default settings, use the following syntax:

doskey

Parameter

macroname=[*text*]

Creates a macro that carries out one or more MS-DOS commands (a Doskey macro). *Macroname* specifies the name you want to assign to the macro. *Text* specifies the commands you want to record.

Switches

/reinstall

Installs a new copy of the Doskey program, even if one is already installed. In the latter case, the **/reinstall** switch also clears the buffer.

/bufsize=*size*

Specifies the size of the buffer in which Doskey stores commands and Doskey macros. The default size is 512 bytes. The minimum buffer size is 256 bytes.

/macros

Displays a list of all Doskey macros. You can use a redirection symbol (>) with the **/macros** switch to redirect the list to a file. You can abbreviate the **/macros** switch as **/m**.

/history

Displays a list of all commands stored in memory. You can use a redirection symbol (>) with the **/history** switch to redirect the list to a file. You can abbreviate the **/history** switch as **/h**.

/insert|/overstrike

Specifies whether new text you type is to replace old text. If you use the **/insert** switch, new text that you type on a line is inserted into old text (as if you had pressed the INSERT key). If you use the **/overstrike** switch, new text replaces old text. The default setting is **/overstrike**.

Notes

Recalling a command

To recall a command, you can use any of the following keys after loading Doskey into memory:

UP ARROW

Recalls the MS-DOS command you used before the one displayed.

DOWN ARROW

Recalls the MS-DOS command you used after the one displayed.

PAGE UP

Recalls the oldest MS-DOS command you used in the current session.

PAGE DOWN

Recalls the most recent MS-DOS command you used.

Editing the command line

With the Doskey program, you can edit the current command line. The following list describes the Doskey editing keys and their functions:

LEFT ARROW
: Moves the cursor back one character.

RIGHT ARROW
: Moves the cursor forward one character.

CTRL+LEFT ARROW
: Moves the cursor back one word.

CTRL+RIGHT ARROW
: Moves the cursor forward one word.

HOME
: Moves the cursor to the beginning of the line.

END
: Moves the cursor to the end of the line.

ESC
: Clears the command from the display.

F1
: Copies one character from the *template* to the MS-DOS command line. (The template is a memory buffer that holds the last command you typed.)

F2
: Searches forward in the template for the next key you type after pressing F2. Doskey inserts the text from the template up to but not including the character you specify.

F3
: Copies the remainder of the template to the command line. Doskey begins copying characters from the position in the template that corresponds to the position indicated by the cursor on the command line.

F4 Deletes characters, beginning with the first character in the template, up to a character you specify. To use this editing key, you press F4 and type a character. Doskey deletes up to, but not including, that character.

F5 Copies the current command into the template and clears the command line.

F6 Places an end-of-file character (CTRL+Z) at the end of the current command line.

F7 Displays all commands stored in memory, with their associated numbers. Doskey assigns these numbers sequentially, beginning with 1 for the first (oldest) command stored in memory.

ALT+F7

Deletes all commands stored in memory.

F8 Searches memory for a command that you want Doskey to display. To use this editing key, type the first character, or the first few characters, of the command you want Doskey to search for and then press F8. Doskey displays the most recent command that begins with the text you typed. Press F8 repeatedly to cycle through all the commands that start with the characters you specified.

F9 Prompts you for a command number and displays the command associated with the number you specify. To display all the numbers and their associated commands, press F7.

ALT+F10

Deletes all macro definitions.

Specifying a default insert mode

If you press the INSERT key, you can type text on the Doskey command line in the middle of old text without replacing the old text. However, once you press ENTER, Doskey returns your keyboard to replace mode. You must press INSERT again to return to insert mode.

The **/insert** switch puts your keyboard in insert mode each time you press ENTER. Your keyboard effectively remains in insert mode until you use the **/overstrike** switch. You can temporarily return to replace mode by pressing the INSERT key; but once you press ENTER, Doskey returns your keyboard to insert mode.

The cursor changes shape when you use the INSERT key to change from one mode to the other.

Creating a macro

You can use the Doskey program to create macros that carry out one or more MS-DOS commands.

You can use the following special characters to control command operations when defining a macro:

$G or **$g**

Redirects output. Use either of these special characters to send output to a device or a file instead of to the screen. This character is equivalent to the redirection symbol for output (>).

GG or **gg**

Appends output to the end of a file. Use either of these special double characters to append output to an existing file rather than replace the data in the file. These double characters are equivalent to the "append" redirection symbol for output (>>).

$L or **$l**

Redirects input. Use either of these special characters to read input from a device or a file instead of from the keyboard. This character is equivalent to the redirection symbol for input (<).

$B or **$b**

Sends macro output to a command. Using one of these special characters is equivalent to using the pipe (|) on a command line.

$T or **$t**

Separates commands. Use either of these special characters to separate commands when you are creating macros or typing commands on the Doskey command line.

**$$** Specifies the dollar-sign character ($).

$1 through **$9**

Represent any command-line information you want to specify when you run the macro. The special characters **$1** through **$9** are batch parameters, which make it possible for you to use different data on the command line each time you run the macro. The **$1** character in a **doskey** command is similar to the **%1** character in a batch program.

$* Represents *all* the command-line information you want to specify when you type the macro name. The special character **$*** is a replaceable parameter that is similar to the batch parameters **$1** through **$9**, with one important difference. Here, *everything* you type on the command line after the macro name is substituted for the **$*** in the macro.

For example, to create a macro that performs a quick and unconditional format of a disk, type the following command:

```
doskey qf=format $1 /q /u
```

For information about quick and unconditional formatting, see the **format** command.

You can use the **doskey** command in a batch program to create a macro.

Running a macro

To run a macro, type the macro name starting at the first position on the command line. If the macro was defined with **$*** or any of the batch parameters **$1** through **$9**, use a space to separate parameters.

You could run the **qf** macro created in the previous example to format a disk in drive A quickly and unconditionally. To do so, you would type the following command:

```
qf a:
```

You cannot run a macro from a batch program.

Creating a macro with the same name as an MS-DOS command

You might want to create a macro that has the same name as an MS-DOS command. This can be useful, for example, if you always use a certain command with specific switches. To specify whether you want to run the macro or the MS-DOS command, follow these guidelines:

- To run the macro, begin typing the macro name immediately after the command prompt, with no space between the prompt and the command name.
- To carry out the command, insert one or more spaces between the command prompt and the command name.

Deleting a macro

To delete a macro type the following command:

```
doskey macroname=
```

Examples

The **/macros** and **/history** switches are useful for creating batch programs to save macros and commands. For example, to create a batch program named MACINIT.BAT that includes all Doskey macros, type the following command:

```
doskey /macros > macinit.bat
```

To use the MACINIT.BAT file, edit it to include the **doskey** command at the beginning of each macro line.

To create a batch program named TMP.BAT that contains recently used commands, type the following command:

```
doskey /history > tmp.bat
```

To define a macro with multiple commands, use **$t** to separate commands, as follows:

```
doskey tx=cd\temp$tdir/w $*
```

In the preceding example, the **tx** macro changes the current directory to TEMP and then displays a directory listing, using the wide display format. You can use **$*** at the end of the macro to append other switches to the **dir** command when you run **tx**.

The following macro uses a batch parameter for a new directory name. The macro first creates a new directory and then changes to it from the current directory.

```
doskey mc=md $1$tcd $1
```

To use the preceding macro to create and change to a directory named BOOKS, you type the following:

```
mc books
```

To create a macro that uses batch parameters for moving a file or group of files, type the following command:

```
doskey mv=copy $1 $2 $t del $1
```

- ■ MS-DOS
- □ Batch
- □ CONFIG.SYS
- □ Internal
- ■ External
- ■ Network

Dosshell

Starts MS-DOS Shell, a graphical interface to MS-DOS.

For an introduction to using MS-DOS Shell, see Chapter 3, "MS-DOS Shell Basics."

Syntax

To start MS-DOS Shell in text mode, use the following syntax:

dosshell [**/t**[:*res*[*n*]]] [**/b**]

To start MS-DOS Shell in graphics mode, use the following syntax:

dosshell [**/g**[:*res*[*n*]]] [**/b**]

Parameters

res Specifies a screen-resolution category. Valid values are **l**, **m**, and **h** to specify low, medium, and high resolution, respectively. The default value of *res* depends on your hardware.

n Specifies a screen resolution when there is more than one choice within a category. For information about the valid values for this parameter, see the following "Note" section. The default value of *n* depends on your hardware.

Switches

/t Starts MS-DOS Shell in text mode.

/b Starts MS-DOS Shell using a black-and-white color scheme.

/g Starts MS-DOS Shell in graphics mode.

Note

Once you have started MS-DOS Shell, you can adjust the screen resolution by using the Display command on the Options menu. A dialog box displays the mode (text or graphics), the number of lines, the resolution category, and the specific number within each category for all possible screen-resolution modes available for your hardware.

Example

To start MS-DOS Shell in graphics mode, type the following command:

```
dosshell /g
```

☐ MS-DOS
☐ Batch
■ CONFIG.SYS
☐ Internal
☐ External
☐ Network

Drivparm

Defines parameters for block devices when you start MS-DOS.

The **drivparm** command modifies the parameters of an existing physical drive. It does not create a new logical drive. The settings specified in the **drivparm** command override the driver definitions for any previous block device.

For an introduction to using CONFIG.SYS, see Chapter 11, "Customizing Your System."

Syntax

drivparm=/d:*number* [**/c**] [**/f:***factor*] [**/h:***heads*] [**/i**] [**/n**] [**/s:***sectors*] [**/t:***tracks*]

Switches

/d:*number*

Specifies the physical drive number. Values for *number* must be in the range 0 through 255 (for example, drive number 0 = drive A, 1 = drive B, 2 = drive C, and so on).

/c Specifies that the drive can detect whether the drive door is closed.

/f:*factor*

Specifies the drive type. The following list shows the valid values for *factor* and a brief description of each. The default value is 2.

0	160K/180K or 320K/360K
1	1.2 megabyte (MB)
2	720K (3.5-inch disk)
5	Hard disk
6	Tape
7	1.44 MB (3.5-inch disk)
8	Read/write optical disk
9	2.88 MB (3.5-inch disk)

/h:*heads*

Specifies the maximum number of heads. Values for *heads* must be in the range 1 through 99. The default value depends upon the value you specify for **/f:***factor*.

/i Specifies an electronically-compatible 3.5-inch floppy disk drive. (Electronically-compatible drives are installed on your computer and use your existing floppy-disk-drive controller.) Use the **/i** switch if your computer's ROM BIOS does not support 3.5-inch floppy disk drives.

/n Specifies a non-removable block device.

/s:*sectors*

Specifies the number of sectors per track that the block device supports. Values for *sectors* must be in the range 1 through 99. The default value depends upon the value you specify for **/f:***factor*.

/t:*tracks*

Specifies the number of tracks per side that the block device supports. The default value depends upon the value you specify for **/f:***factor*.

Notes

Using the /i switch

Use the **/i** switch if your system does not support 3.5-inch floppy disk drives. (Some IBM PC/AT-compatible systems do not have a ROM BIOS that supports 3.5-inch floppy disk drives.)

Disk drive change-line support

Change-line support means that a physical disk drive can detect whether the drive door is open. Change-line support improves performance by letting MS-DOS know when one floppy disk has been replaced by another. The **/c** switch allows MS-DOS to make use of change-line support. To find out whether your disk drive has change-line support, see your disk-drive documentation.

Creating a logical drive

Drivparm modifies the parameters of an existing physical drive and does not create a new logical drive. For information about creating a new logical drive and associating it with a physical drive, see Chapter 11, "Customizing Your System."

Example

Suppose your system has an internal tape drive with one head on drive D that is configured at startup to write 20 tracks of 40 sectors per track. To reconfigure this tape drive to write 10 tracks of 99 sectors each, add the following command to your CONFIG.SYS file:

```
drivparm=/d:3 /f:6 /h:1 /s:99 /t:10
```

☐ MS-DOS
■ Batch
☐ CONFIG.SYS

■ Internal
☐ External

☐ Network

Echo

Turns the command-echoing feature on or off, or displays a message.

When you run a batch program, MS-DOS typically displays (echoes) the batch program's commands on the screen. You can turn this feature on or off by using the **echo** command. For an introduction to using batch programs, see Chapter 10, "Working with Batch Programs."

Syntax

echo [**on**|**off**]

To use the **echo** command to display a message, use the following syntax:

echo [*message*]

Parameters

on|**off**

Specifies whether to turn the command-echoing feature on or off. To display the current **echo** setting, use the **echo** command without a parameter.

message

Specifies text you want MS-DOS to display on the screen.

Notes

Using a message with the echo command

The **echo** *message* command is useful when **echo** is off. To display a message that is several lines long without displaying other commands, you can include several **echo** *message* commands after the **echo off** command in your batch program.

Hiding the command prompt

If you use the **echo off** command on the command line, the command prompt does not appear on your screen. To redisplay the command prompt, type **echo on**.

Preventing MS-DOS from echoing a line

You can insert an at sign (@) in front of a command in a batch program to prevent MS-DOS from echoing that line.

Echoing a blank line

To echo a blank line on the screen, you can type **echo** and then a period (**echo.**). There must be no intervening space.

Displaying pipes and redirection characters

You cannot display a pipe (|) or redirection character (> or <) by using the **echo** command.

Examples

The following example shows a batch program that includes a three-line message preceded and followed by a blank line:

```
echo off
echo.
echo This batch program
echo formats and checks
echo new disks
echo.
```

If you want to turn **echo** off and you do not want to echo the **echo** command itself, include an at sign (@) before the command, as follows:

```
@echo off
```

You can use the **if** and **echo** commands on the same command line, as follows:

```
if exist *.rpt echo The report has arrived.
```

Related Command

For information about suspending the execution of a batch program, see the **pause** command.

- ■ MS-DOS
- ☐ Batch
- ☐ CONFIG.SYS
- ☐ Internal
- ■ External
- ■ Network

Edit

Starts MS-DOS Editor, which creates and changes ASCII text files.

MS-DOS Editor is a full-screen editor that allows you to create, edit, save, and print ASCII text files. Using MS-DOS Editor, you can choose commands from menus and specify information and preferences in dialog boxes. MS-DOS Editor includes extensive online Help to give you information about MS-DOS Editor techniques and commands. For an introduction to using MS-DOS Editor, see Chapter 9, "Working with MS-DOS Editor."

Syntax

edit [[*drive*:][*path*]*filename*] [**/b**] [**/g**] [**/h**] [**/nohi**]

Parameter

[drive:][path]filename

Specifies the location and name of an ASCII text file. If the file does not exist, MS-DOS Editor creates it. If the file exists, MS-DOS Editor opens it and displays its contents on the screen.

Switches

/b Displays MS-DOS Editor in black and white. You use this option if MS-DOS Editor isn't displayed correctly on a monochrome monitor.

/g Uses the fastest screen updating for a CGA monitor.

/h Displays the maximum number of lines possible for the monitor you are using.

/nohi

Enables you to use 8-color monitors with MS-DOS Editor. Usually, MS-DOS uses 16 colors.

CAUTION MS-DOS Editor does not work if the file QBASIC.EXE is not in the current directory or in the search path or in the same directory as the file EDIT.COM. If you delete QBASIC.EXE to save space on your hard disk, you cannot use MS-DOS Editor.

Note

Some monitors may not support the display of shortcut keys by default. If your monitor does not display shortcut keys, use the **/b** switch (for CGA monitors) and the **/nohi** switch (for systems that do not support bold characters).

■ MS-DOS
□ Batch
□ CONFIG.SYS
□ Internal
■ External
■ Network

Edlin

Starts Edlin, a line-oriented text editor with which you can create and change ASCII files.

Edlin numbers each line of the text file that is located in memory. You can use Edlin to insert, modify, copy, move, and delete lines of the file. If you want to use a full-screen editor, use the **edit** command.

Syntax

edlin [*drive*:][*path*]*filename* [**/b**]

Parameter

[*drive*:][*path*]*filename*

Specifies the location and name of an ASCII file on a disk. If the file exists, Edlin opens it. If the file does not exist, Edlin creates a file in memory and uses the specified location and filename to create the file on a disk when you use the Edlin **e** command.

Switch

/b Specifies that Edlin is to ignore the end-of-file character (CTRL+Z).

Notes

Maximum line length

Edlin accepts a maximum of 253 characters per line.

Edlin commands

The following is a list of Edlin commands with a brief description of each command:

[*line*]

Displays the line you specify.

? Displays a list of Edlin commands.

a Loads a portion of a file into memory when insufficient memory prohibits loading the entire file.

c Copies a block of consecutive lines to the line number you specify.

d Deletes a block of consecutive lines.

e Writes the edited file from memory to a disk (saves the file), and stops the Edlin session.

i Inserts one or more lines.

l Displays a block of consecutive lines.

m Moves a block of consecutive lines.

p Displays a file one page at a time.

q Stops the Edlin session without writing the edited file from memory to a disk.

r Searches for a string of one or more characters, and replaces it.

s Searches for a string of one or more characters.

t Merges the contents of another file on a disk with the contents of the file that is in memory.

w Writes the first portion of the file in memory to a disk.

For descriptions of these commands, see the following pages.

Meaning of the asterisk character in Edlin

The asterisk (*) is used for two purposes in Edlin. When an asterisk appears as the only character on the display line, it is the Edlin prompt after which you type Edlin commands. When an asterisk appears after a line number on the display line, it indicates that the line is the current line (where the cursor is located).

Meaning of a page of text

A *page* of text is one full screen of information. With a 25-line screen mode, Edlin displays 24 lines of text per page. The number of lines per page depends on the screen mode you are using.

Starting and stopping insert mode

To insert lines into the file in memory, use the Edlin **i** (insert) command. Once you have finished inserting lines, press ENTER and then CTRL+C to stop insert mode. For more information about inserting lines, see the Edlin **i** command.

Editing keys

MS-DOS provides several editing keys that you can use to edit the file in memory. For information about these keys, see the "Using Editing Keys" section of Chapter 7, "Advanced Command Techniques."

- ■ MS-DOS
- ☐ Batch
- ☐ CONFIG.SYS
- ☐ Internal
- ■ External
- ☐ Network

Edlin: [*line*]

Displays the line of text you specify.

When you type a line number as a command, Edlin displays two lines. The first line contains the number you specified and its associated text. The second line contains the number again, followed by the cursor. The text on the first line of the display serves as a template for the second line. On the second line of the display, you can press ENTER to cancel the command without changing the text, type replacement text, or edit the line of text.

Syntax

[*line*]

Parameter

line

Specifies the number of the line you want Edlin to display. To see the number and text of the current line, press ENTER.

Note

Entering changes into memory

After you edit a line, press ENTER to enter the changes into memory.

CAUTION If you press ENTER while the cursor is in the middle of a line, Edlin deletes the portion of the line that is between the cursor and the end of the line.

For information about saving the edited file from memory to a disk, see the Edlin **e** and **w** commands.

Example

Suppose that the following file is in memory and ready to edit. When you use the Edlin **l** (list) command at the Edlin prompt, Edlin displays the contents of the file.

```
1: Dear Mr. Muster:
2:
3: Congratulations on your promotion
4: to the position of Senior Chemical
5: Engineer. I continue to be most
6: impressed with your work.
```

To edit line 6, type **6**. Edlin displays the following two lines:

```
6:*impressed with your work.
6:*_
```

The first line contains the specified line number and its associated text. The second line contains the same line number and the cursor.

Now suppose you want to insert the word "fine" before the word "work" in the previous example. You can specify that Edlin is to redisplay a portion of the first line, beginning at the cursor position on the second line. First, press F2 and type **w**. Edlin displays up to, but not including, the first "w" in line 6, as follows:

```
6:*impressed _
```

Then, press F2 and type **w** again. Edlin displays up to, but not including the next "w" in line 6, as follows:

```
6:*impressed with your _
```

Now press the INSERT key and type **fine** and then a space. Then press the F3 key. Edlin displays the edited line, as follows:

```
6:*impressed with your fine work._
```

Press ENTER to accept the change.

At the Edlin prompt, use the Edlin **l** (list) command to see a display of the edited file now in memory. Edlin displays the following:

```
1: Dear Mr. Muster:
2:
3: Congratulations on your promotion
4: to the position of Senior Chemical
5: Engineer. I continue to be most
6:*impressed with your fine work.
```

Edlin: A (Append)

■ MS-DOS	
☐ Batch	
☐ CONFIG.SYS	
☐ Internal	
■ External	
☐ Network	

Loads a portion of a file into memory when insufficient memory prevents Edlin from loading the entire file.

When you start Edlin, it reads as many lines as possible from your disk file into memory. If the size of your file exceeds available memory, you must edit your file in stages. That is, you edit the first part of the file, write that part of the file to your disk by using the **w** (write) command, and then load more unedited lines from your disk into memory.

Syntax

[*n*]**a**

Parameter

n Specifies the number of lines you want Edlin to read into memory from the disk.

Notes

Default setting

If you do not specify a value for *n*, Edlin loads lines from the disk file until available memory is 75-percent full. If available memory is already 75-percent full, Edlin loads no lines.

Freeing extra memory

If available memory is already full, you may be able to free memory by writing a portion of the file to a disk, by stopping other programs, or by restarting MS-DOS after quitting MS-DOS Editor. Restarting MS-DOS clears memory being used by memory-resident programs.

End-of-file message

After the **a** command reads the last line of the file into memory, Edlin displays the following message:

```
End of input file
```

Example

Suppose the last 100 lines of your disk file do not fit into memory. After you edit the first part of the file and write a portion of it back to a disk, you can type the following command to load the remaining 100 lines into memory:

```
100a
```

- ■ MS-DOS
- ☐ Batch
- ☐ CONFIG.SYS
- ☐ Internal
- ■ External
- ☐ Network

Edlin: C (Copy)

Copies a block of consecutive lines to one or more locations within the file in memory.

The **c** command copies the block of consecutive lines you specify to a line number you specify. This block can be copied as many times as necessary.

Syntax

[*line1*],[*line2*],*line3*[,*count*]**c**

Parameters

line1

Specifies the first line you want Edlin to copy.

line2

Specifies the last line you want Edlin to copy.

line3

Specifies the line before which Edlin is to insert the specified block of lines.

count

Specifies the number of times you want Edlin to copy the block of lines.

Notes

Default settings

If you omit *line1* or *line2*, Edlin copies only the current line. You must include the commas on the command line even if you omit one or both of these parameters.

If you omit the *count* parameter, Edlin copies the lines one time.

Line renumbering

After Edlin copies lines, you can use the Edlin **l** (list) command at the Edlin prompt to see the correctly renumbered lines.

Overlapping line numbers

The line you specify for the *line3* parameter cannot be part of the block of lines to be copied. If you overlap line numbers in this way, Edlin cannot complete the copy operation and displays the following message:

```
Entry error
```

For example, the following command results in an error message:

```
3,20,15c
```

Examples

If you type the following command, Edlin copies lines 1 through 5 one time, beginning on line 6:

```
1,5,6c
```

Lines 6 through 10 become identical to lines 1 through 5.

To copy the current line to line 5, use the following command:

```
,,5c
```

■ MS-DOS
☐ Batch
☐ CONFIG.SYS
☐ Internal
■ External
☐ Network

Edlin: D (Delete)

Deletes the block of consecutive lines you specify.

Syntax

[*line1*][,*line2*]**d**

Parameters

line1

Specifies the first line you want Edlin to delete.

line2

Specifies the last line you want Edlin to delete.

Notes

Default parameter values

If you omit both parameters or only the *line2* parameter, Edlin deletes the current line. However, if you omit only the *line1* parameter, Edlin deletes the block of text that includes the current line through the line whose number is specified for *line2*. In the latter case, you cannot specify a line number for *line2* that precedes the current line number. In general, the number you specify for *line2* cannot be smaller than the number you specify for *line1*. If you omit only the *line1* parameter, you need to insert a comma as a placeholder preceding *line2*, as shown in the syntax line.

Line renumbering

After Edlin deletes lines, you can use the Edlin **l** (list) command at the Edlin prompt to see the correctly renumbered lines that remain.

Examples

If you want Edlin to delete line 7, type:

```
7d
```

If you want Edlin to delete the block of text on lines 22 through 32, type the following command:

```
22,32d
```

Finally, suppose that the number of the current line is 7. To specify that Edlin is to delete the block of text that includes the current line through line 11, type the following command:

```
,11d
```

■ MS-DOS
☐ Batch
☐ CONFIG.SYS
☐ Internal
■ External
☐ Network

Edlin: E (End)

Writes the current file from memory to a disk and stops the Edlin session.

The **e** command renames the original input file on the disk with the .BAK extension, writes the edited file from memory to the original input file on the disk, and then stops the Edlin session. However, if the file in memory is one that you created during this session rather than one that Edlin loaded from a disk, Edlin does not create a backup (.BAK) file on the disk.

Syntax

e

Notes

Default drive and directory

Edlin writes the edited file from memory to the drive, directory, and filename on a disk that you specified when you started the current Edlin session. If you omitted a drive name at that time, Edlin writes to the current drive. If you omitted a directory name at that time, Edlin writes to the current directory.

Checking for disk space

Before using the **e** command, you should be sure your disk contains enough free space for the entire edited file that is in memory. If it does not, Edlin loses part or all of the file.

Read-only .BAK file

Suppose you want Edlin to save an edited file from memory to a disk, but the .BAK version of the file is a read-only file. In this case, Edlin displays a message in the following format to inform you that Edlin cannot replace the .BAK file:

```
Access denied  - [drive:][path]filename.BAK
```

Both the original and backup versions of your file on the disk remain unchanged.

■ MS-DOS
☐ Batch
☐ CONFIG.SYS
☐ Internal
■ External
☐ Network

Edlin: I (Insert)

Inserts lines before the line number you specify in the edited file in memory.

If you are creating a new file, you must type the **i** command before you can insert a new line. Edlin displays the next line number each time you press ENTER. Edlin remains in insert mode until you press CTRL+C.

Syntax

[*line*]**i**

Parameter

line

Specifies the line number before which you want Edlin to insert lines. The default value of *line* is the number of the current line.

Notes

Line renumbering

When you quit insert mode, the line immediately following the inserted lines becomes the current line. You can use the Edlin **l** (list) command at the Edlin prompt to see the correctly renumbered lines.

Inserting control characters

To insert a control character in text, type **^V** followed by the ASCII symbol that represents the control character. For example, to insert an escape character (CTRL+[), type the following:

```
^V[
```

To insert a character that produces a tone (CTRL+G), type the following:

```
^VG
```

Appending text

If the value for *line* exceeds the number of lines in the file you are editing or if you specify a number sign (#) for *line*, Edlin appends the inserted line(s) to the end of the file. In either case, the last line you insert becomes the current line. If only a portion of the file is in memory, the line is appended at the end of the portion in memory.

Example

Suppose you have used the Edlin **l** (list) command at the Edlin prompt to display the following text on your screen:

```
 1: Dear Mr. Muster:
 2:
 3: Congratulations on your promotion
 4: to the position of Senior Chemical
 5: Engineer. I continue to be most
 6: impressed with your work.
 7:
 8: Sincerely,
 9:
10: S.L. Martin, President
```

Suppose you want to add another paragraph to the letter. To insert text before line 8, type **8i**. Edlin displays the following:

```
8:*_
```

Now type the following line at the cursor on line 8:

```
8:*I think you will enjoy working with
```

Press ENTER at the complete of each new line and continue by typing the following lines:

```
 9:*Mr. Lang on the new project. Please
10:*let me know if there is anything I
11:*can do to assist you.
```

Edlin displays the following:

```
12:*_
```

Insert a blank line by pressing ENTER and complete the insertion by pressing CTRL+C on the next line. You can type **1l** to see the following correctly renumbered lines:

```
 1: Dear Mr. Muster:
 2:
 3: Congratulations on your promotion
 4: to the position of Senior Chemical
 5: Engineer. I continue to be most
 6: impressed with your work.
 7:
 8: I think you will enjoy working with
 9: Mr. Lang on the new project. Please
10: let me know if there is anything I
11: can do to assist you.
12:
13:*Sincerely,
14:
15: S.L. Martin, President
```

- ■ MS-DOS
- ☐ Batch
- ☐ CONFIG.SYS
- ☐ Internal
- ■ External
- ☐ Network

Edlin: L (List)

Displays the block of consecutive lines you specify.

Syntax

[*line1*][,*line2*]**l**

Parameters

line1

Specifies the first line you want Edlin to display.

line2

Specifies the last line you want Edlin to display.

Notes

Default values

You can omit the *line1* parameter, the *line2* parameter, or both. The following list describes the default value(s) for each of these cases:

- If you omit only the *line1* parameter, Edlin displays up to one page (full screen of text) at a time, beginning 11 lines before the current line and ending with the line whose number is specified in *line2*. When you omit only *line1*, you must insert a comma as a placeholder.
- If you omit only the *line2* parameter, Edlin displays up to one page, beginning with the line whose number is specified in *line1*.
- If you use the Edlin **l** (list) command with no parameters, Edlin displays up to one page, beginning 11 lines before the current line. If you install the ANSI.SYS device driver, the number of lines Edlin displays per page depends on the type of monitor you have. This number might be greater than 24.

Blocks of more than one page

When the block of lines you specify contains more than one page, Edlin displays the first page and then prompts you with the following message:

```
Continue (Y/N)?
```

Example

To see lines 5 through 10, type the following:

```
5,10l
```

■ MS-DOS
☐ Batch
☐ CONFIG.SYS

☐ Internal
■ External

☐ Network

Edlin: M (Move)

Moves the block of consecutive lines you specify to another location in the file in memory.

Syntax

[*line1*],[*line2*],*line3***m**

[*line1*],+*n*,*line3***m**

Parameters

line1

Specifies the first line you want Edlin to move.

line2

Specifies the last line you want Edlin to move.

line3

Specifies the line before which you want Edlin to move the block of lines.

+n

Specifies that you want Edlin to move the block of lines that begins with the line whose number is specified in *line1* and includes the next *n* lines. If you omit the *line1* parameter, the block of lines to be moved begins with the current line.

Notes

Line renumbering

After Edlin moves lines, you can use the Edlin **l** (list) command at the Edlin prompt to see the correctly renumbered lines.

Overlapping line numbers

The line you specify for the *line3* parameter cannot be part of the block of lines to be moved. If you overlap line numbers in this way, Edlin cannot complete the move operation and displays the following message:

```
Entry error
```

For example, the following command results in an error message:

```
5,10,8m
```

Examples

Suppose that the following file is in memory and ready to edit. You can type **1l** at the Edlin prompt to see the contents of the file.

```
 1: Dear Mr. Muster:
 2:
 3: Congratulations on your promotion
 4: to the position of Senior Chemical
 5: Engineer. I continue to be most
 6: impressed with your hard work.
 7:
 8: I think you will enjoy working with
 9: Mr. Lang on the new project. Please
10: let me know if there is anything I
11: can do to assist you.
12:
13: Sincerely,
14:
15: S.L. Martin, President
16: Rockdale Corporation
17: "A World Leader in Technology"
```

What if you prefer to have the motto at the beginning of the memo? You can move lines 16 and 17 before the existing line 1 by typing the following command:

```
16,17,1m
```

Type the Edlin **l** (list) command at the Edlin prompt to see the following correctly renumbered lines:

```
 1: Rockdale Corporation
 2: "A World Leader in Technology"
 3: Dear Mr. Muster:
 4:
 5: Congratulations on your promotion
 6: to the position of Senior Chemical
 7: Engineer. I continue to be most
 8: impressed with your hard work.
 9:
10: I think you will enjoy working with
11: Mr. Lang on the new project. Please
12: let me know if there is anything I
13: can do to assist you.
14:
15: Sincerely,
16:
17: S.L. Martin, President
```

The following command specifies that Edlin is to move the block of lines including the current line through the next 25 lines to immediately before line 100:

```
,+25,100m
```

- ■ MS-DOS
- ☐ Batch
- ☐ CONFIG.SYS
- ☐ Internal
- ■ External
- ☐ Network

Edlin: P (Page)

Displays all or part of a file, one page (full screen of text) at a time.

The last line displayed per screen becomes the current line.

Syntax

[*line1*][, *line2*]**p**

Parameters

line1

Specifies the first line you want Edlin to display.

line2

Specifies the last line you want Edlin to display.

Notes

Omitting only the *line1* parameter
When you omit the *line1* parameter, Edlin displays a page of text that begins with the current line through *line2*.

Omitting only the *line2* parameter
When you omit the *line2* parameter, Edlin displays a page of text that begins with the line whose number you specify for *line1*.

Omitting both parameters
When you omit both parameters, Edlin displays a page of text that begins with the line after the current line.

Example

To see lines 100 through 200, one page at a time, type the following command:

```
100,200p
```

■ MS-DOS
☐ Batch
☐ CONFIG.SYS

☐ Internal
■ External

☐ Network

Edlin: Q (Quit)

Stops the current Edlin session without writing the edited file from memory to a disk.

When you use the **q** command, the Edlin session stops and the MS-DOS prompt appears.

To specify that Edlin is to write the edited file from memory to a disk before ending the current session, you must use the **e** (end) command.

Syntax

q

Notes

A difference between the q and e commands
Suppose that the file you are editing is one that Edlin loaded into memory from a disk at the beginning of this session rather than one that you created in memory during the session. If you use the **q** command to quit the session, the contents of both the original input disk file and the .BAK version of the disk file (if one exists) remain unchanged. However, if you use the **e** command to quit the session and the file you are editing has changed during the session, the edits are saved and the original input disk file becomes the .BAK version.

Quitting Edlin without writing the edited file from memory to a disk

Use the following procedure to quit the Edlin session without writing the edited file from memory to a disk:

1. At the Edlin prompt, type **q**. Edlin displays the following message:

   ```
   Abort edit (Y/N)? _
   ```

2. Press Y (for yes).

■ MS-DOS
☐ Batch
☐ CONFIG.SYS
☐ Internal
■ External
☐ Network

Edlin: R (Replace)

Searches a block of consecutive lines for a string of one or more characters you specify, and replaces each occurrence of that string with another string you specify.

The last line in which the replacement occurs becomes the new current line.

Syntax

[*line1*][,*line2*][**?**]**r**[*string1*][*separator string2*]

Parameters

line1

Specifies the first line in which you want Edlin to replace the string specified in *string1*.

line2

Specifies the last line in which you want Edlin to replace the string specified in *string1*.

? (question mark)

Specifies that Edlin is to prompt you by displaying a confirmation message before replacing an occurrence of the string specified in *string1*.

string1

Specifies the string that you want Edlin to replace.

separator

Separates the *string1* and *string2* values. The only valid value for this parameter is the end-of-file character (CTRL+Z).

string2

Specifies the new string that is to replace each occurrence of the string specified for *string1*.

Notes

Command-line spacing

You must not insert a space between the **r** and any subsequent parameter on the command line.

Default settings

If you omit the *line1* parameter, Edlin begins the search on the line after the current line. If you omit the *line2* parameter, Edlin stops the search at the end of the file or at the end of the portion of text in memory.

If you omit the *string1* parameter, Edlin uses the more recently used of the following two values: the value that you specified for *string1* the last time you used the **r** command or the value that you specified for *string* the last time you used the **s** command during this session. If you omit *string1* and you have not used the **r** or **s** command yet during the editing session, the command stops.

If you omit the *string2* parameter, Edlin uses the value you specified the last time you used the **r** command during this session. If you omit the *string2* parameter and you have not used the **r** command yet during this session, Edlin deletes all occurrences of the string that is specified for *string1*.

Using the *separator* parameter

You must separate the *string1* and *string2* values by using the CTRL+Z key combination. Even if you omit *string1*, you need to press CTRL+Z to mark the beginning of *string2*. When you press the CTRL+Z key combination, the characters displayed are not "CTRL+Z". Instead, you see the following:

```
^Z
```

Using the question mark (?)

If you include the **?** parameter in your command, Edlin displays the line containing the first occurrence of the string specified for *string1* and prompts you by displaying the following confirmation message:

```
O.K.? _
```

If you press Y (for yes) or press ENTER, Edlin replaces this occurrence of the value for *string1* with the value for *string2* and searches for the next occurrence. If you press N (for no), Edlin does not replace this occurrence of the value for *string1* and searches for the next occurrence.

If you do not use the question mark (?)

If you do not use the **?** parameter to confirm replacements as they are made, Edlin makes all the replacements at once and then displays each line that contains a replacement. If a line contains two or more replacements, Edlin displays the line once for each replacement.

Examples

Suppose you want Edlin to carry out only each confirmed replacement of the word "mine" with the word "ours" within the first 20 lines of the edited file in memory. Type the first part of the command as follows, but do not press ENTER:

```
1,20?rmine
```

To complete the command, press CTRL+Z (which appears on the screen as ^Z), type the word **ours**, and press ENTER. The complete command appears on the screen as follows:

```
1,20?rmine^Zours
```

Suppose that the following file is in memory and ready to edit. You can type **1l** at the Edlin prompt to see the contents of the file.

```
 1: Dear Mr. Muster:
 2:
 3: Congratulations on your promotion
 4: to the position of Senior Chemical
 5: Engineer. I continue to be most
 6: impressed with your hard work.
 7:
 8: I think you will enjoy working with
 9: Mr. Lang on the new project. Please
10: let me know if there is anything I
11: can do to assist you.
12:
13: Sincerely,
14:
15: S.L. Martin, President
16: Rockdale Corporation
17: "A World Leader in Technology"
```

Now suppose that in lines 5 through 10 you want Edlin to replace all occurrences of the word "I" with the words "yours truly". Type the first part of the command as follows, but do not press ENTER:

```
5,10rI
```

To complete the command, press CTRL+Z (which appears on the screen as ^Z), type the words **yours truly**, and press ENTER. The complete command appears on the screen as follows:

```
5,10rI^Zyours truly
```

Because the **?** parameter is omitted, Edlin replaces the three occurrences of "I" without prompting you by displaying the confirmation message. When Edlin finishes carrying out the command, it displays the following lines, which are changed as a result of the three replacements:

```
 5: Engineer. yours truly continue to be most
 8: yours truly think you will enjoy working with
10: let me know if there is anything yours truly
```

In the previous example, two unintended replacements occurred—in lines 5 and 8. You can avoid such changes by adding the **?** parameter to the command. The completed command should appear on screen as follows:

```
5,10?rI^Zyours truly
```

Now, Edlin prompts you by displaying the confirmation message for each occurrence of the string specified in *string1* and carries out only confirmed replacements, as the following example shows:

```
       5: Engineer. yours truly continue to be most
O.K.? n
       8: yours truly think you will enjoy working with
O.K.? n
      10: let me know if there is anything yours truly
O.K.? y
```

When the **?** parameter is used, Edlin does not automatically display the lines that are changed as a result of the confirmed replacements. If you type the Edlin **l** (list) command at the Edlin prompt, Edlin displays the edited file that is in memory, as follows:

```
 1: Dear Mr. Muster:
 2:
 3: Congratulations on your promotion
 4: to the position of Senior Chemical
 5: Engineer. I continue to be most
 6: impressed with your hard work.
 7:
 8: I think you will enjoy working with
 9: Mr. Lang on the new project. Please
10: let me know if there is anything yours truly
11: can do to assist you.
12:
13: Sincerely,
14:
15: S.L. Martin, President
16: Rockdale Corporation
17: "A World Leader in Technology"
```

- ■ MS-DOS
- ☐ Batch
- ☐ CONFIG.SYS
- ☐ Internal
- ■ External
- ☐ Network

Edlin: S (Search)

Searches for the string of one or more characters that you specify.

Edlin displays the first line that contains an occurrence of the string. The search then stops and that line becomes the current line.

Syntax

[*line1*][,*line2*][**?**]**s**[*string*]

Parameters

line1

Specifies the first line you want Edlin to search.

line2

Specifies the last line you want Edlin to search.

? (question mark)

Specifies that Edlin is to prompt you by displaying a confirmation message when it finds the first occurrence of the value you specify for *string*.

string

Specifies the string for which you want Edlin to search. You must not insert a space before this parameter on the command line, unless the space is part of the search text.

Notes

Default settings

If you omit the *line1* parameter, Edlin starts the search on the line after the current line. If you omit the *line2* parameter, Edlin stops the search at the end of the file.

If you omit the *string* parameter, Edlin uses the more recently used of the following two values: the value that you specified for *string* the last time you used the **s** command, or the value that you specified for *string1* the last time you used the **r** (replace) command during this session. If you omit the *string* parameter and this is your first use of an **s** or **r** command during this session, the **s** command stops immediately.

Using the ? (question mark)

If you include the **?** parameter in your command, Edlin displays the line containing the first occurrence of the characters specified for *string* and prompts you with the following confirmation message:

```
O.K.? _
```

If you press Y (for yes) or press ENTER, the line displayed before the message becomes the current line and the search stops. If you press N (for no), the search continues until another occurrence is found or until Edlin displays the following message indicating that all lines have been searched:

```
Not found
```

Examples

Suppose that the following file is in memory and ready to edit. You can type **1l** at the Edlin prompt to see the contents of the file.

```
 1: Dear Mr. Muster:
 2:
 3: Congratulations on your promotion
 4: to the position of Senior Chemical
 5: Engineer. I continue to be most
 6: impressed with your hard work.
 7:
 8: I think you will enjoy working with
 9: Mr. Lang on the new project. Please
10: let me know if there is anything I
11: can do to assist you.
12:
13: Sincerely,
14:
15: S.L. Martin, President
```

To specify that Edlin is to search lines 2 through 12 for the first occurrence of the word "to", type the following command:

```
2,12sto
```

Edlin displays the following line:

```
4: to the position of Senior Chemical
```

To specify that Edlin is to display the line containing the first occurrence of "to" and then prompt you with a confirmation message, type the following command:

```
1,?sto
```

Edlin displays the following lines:

```
       4: to the position of Senior Chemical
O.K.? _
```

If you press any key other than Y or ENTER, the search continues. For this example, press N (for no), as follows:

```
O.K.? n
```

Edlin continues the search and displays the following lines:

```
       5: Engineer. I continue to be most
O.K.? _
```

Press Y to stop the search.

- ■ MS-DOS
- ☐ Batch
- ☐ CONFIG.SYS
- ☐ Internal
- ■ External
- ☐ Network

Edlin: T (Transfer)

Merges the contents of another file from a disk with the contents of the file that is in memory.

Syntax

[*line*]**t**[*drive*:][*path*]*filename*

Parameters

line

Specifies the line number before which you want Edlin to insert the file it is transferring from a disk. The default value of this parameter is the number of the current line.

[*drive*:][*path*]*filename*

Specifies the location and name of the file you want Edlin to insert before the line whose number is specified in the *line* parameter. The default value for *drive* is the current drive; the default value for *path* is the current directory.

Note

After Edlin merges a file from a disk, you can use the Edlin **l** (list) command at the Edlin prompt to see the correctly renumbered lines.

Example

To merge a file named TAXES.MEM to line 12 of the file you are editing, type the following command:

```
12t taxes.mem
```

■ MS-DOS
☐ Batch
☐ CONFIG.SYS
☐ Internal
■ External
☐ Network

Edlin: W (Write)

Writes the first portion of the edited file from memory to a disk.

When you start Edlin, it reads as many lines as possible from your disk file into memory. If the size of your file exceeds available memory, you must edit your file in stages. That is, you edit part of the file, write that part to your disk by using the **w** command, and then load the next part from disk by using the **a** command.

Syntax

[*n*]**w**

Parameter

n Specifies the number of lines that you want Edlin to write to the disk, beginning with the first line of the edited file in memory.

Notes

How the w command works

When you open a file, Edlin reads lines from disk until memory is more than 75-percent full. It reserves the other 25 percent for changes you might make to the text. If your entire file fits in memory, Edlin displays the following message:

```
End of input file
```

If you see this message, you do not need to use the **w** and **a** commands.

If Edlin does not display this message when you open a file, the size of the file exceeds available memory. Therefore, you must edit your file in stages by using the **w** and **a** commands to write and read parts of the file, respectively.

The **w** command does not write to disk the changes you make unless it was actually necessary to use the **w** command. Therefore, if you use the **w** command even though the whole file fit into memory and then you use the **q** command to quit Edlin, none of the changes you made to the file are saved.

Line renumbering

After Edlin writes the first portion of the edited file from memory to a disk, you can use the Edlin **l** (list) command at the Edlin prompt to see the correctly renumbered lines that remain, beginning with line number 1.

Default setting

If you omit the *n* parameter, Edlin writes lines from the edited file in memory to a disk until memory is 25-percent full.

Example

Suppose the final 100 lines of your disk file do not fit into memory. After you edit the first part of the file, you can free enough space to load the remainder of your disk file into memory and continue editing by typing the following command:

```
125w
```

- ■ MS-DOS
- ☐ Batch
- ☐ CONFIG.SYS
- ☐ Internal
- ■ External
- ☐ Network

EMM386

Enables or disables EMM386 expanded-memory support on a computer with an 80386 or higher processor.

The **emm386** command also enables or disables Weitek coprocessor support. For an introduction to using the EMM386.EXE device driver, see Chapter 12, "Optimizing Your System" and Chapter 15, "Device Drivers."

Syntax

emm386 [on|off|auto] [w=on|w=off]

To display the current status of EMM386 expanded-memory support, use the following syntax:

emm386

Parameters

on|off|auto

Activates the EMM386.EXE device driver (if set to **on**), or suspends the EMM386.EXE device driver (if set to **off**), or places the EMM386.EXE device driver in auto mode (if set to **auto**). Auto mode enables expanded-memory support only when a program calls for it. The default value is **on**.

w=on|w=off

Enables or disables Weitek coprocessor support. The default value is **w=off**.

Notes

Installing the EMM386.EXE device driver

You must use the **device** command to install the EMM386.EXE device driver before you can use the **emm386** command. To use the EMM386.EXE device driver and the **emm386** command, your computer must have an 80386 or higher processor. If you try to use the **emm386** command on a computer that does not have an 80386 or higher processor, MS-DOS displays the following message:

```
EMM386 driver not installed
```

Reactivating EMM386 expanded-memory support

If EMM386.EXE was loaded when MS-DOS started but is not currently in use, the **on** parameter reactivates expanded-memory support.

Suspending EMM386 expanded-memory support

If EMM386 expanded-memory support is currently active, handle 0 is the only handle allocated, and EMM386.EXE is not providing access to the upper memory area, then the **off** parameter suspends EMM386 expanded-memory support. When EMM386 expanded-memory support is off, the Lotus/Intel/Microsoft Expanded Memory Specification (LIM EMS) header is changed so that programs cannot use expanded memory.

Enabling and disabling Weitek coprocessor support

If the **w=on** parameter is specified and the **off** parameter (different from the **w=off** parameter) is not, EMM386 enables Weitek coprocessor support. The high memory area (HMA) must be available to enable Weitek coprocessor support.

If you specify the **w=on** or **w=off** parameter and no Weitek coprocessor is installed in your computer system, MS-DOS displays the following error message:

```
Weitek Coprocessor not installed
```

- ■ MS-DOS
- □ Batch
- □ CONFIG.SYS
- □ Internal
- ■ External
- ■ Network

Exe2bin

Converts .EXE (executable) files to binary format.

Exe2bin is included with MS-DOS as a courtesy to software developers. It is not useful for general users.

Syntax

exe2bin [*drive1*:][*path1*]*input-file* [[*drive2*:][*path2*]*output-file*]

Parameters

[*drive1*:][*path1*]*input-file*

Specifies the location and name of the input file.

[*drive2*:][*path2*]*output-file*

Specifies the location and name of the output file.

Notes

Restrictions on using exe2bin

The following restrictions apply when you use the **exe2bin** command:

- The input file must be in valid .EXE format produced by the linker and must not be packed.
- The resident, or actual, code and data portions of the file combined must be less than 64K.
- There must be no STACK segment.

Default values for parameters

Exe2bin takes specific actions, depending upon the values you use for the *input-file* and *output-file* parameters.

- The default filename extension for the filename you specify for *input-file* is .EXE. **Exe2bin** converts the input .EXE file to an output file in .BIN format (a memory image of the program) and uses the location and filename you specify for [*drive2*:][*path2*]*output-file* to store that output file.
- If you do not specify *drive2* or *path2*, **exe2bin** writes the output file to the current drive and directory.
- If you do not specify an output filename, **exe2bin** uses the input filename.
- The default extension for the filename specified for the *output-file* parameter is .BIN.

Types of conversion available with exe2bin

Two types of conversion are possible, depending upon whether the initial CS:IP (Code Segment:Instruction Pointer) is specified in the .EXE file. The following list presents the two types:

- If the CS:IP is not specified in the .EXE file, **exe2bin** performs a pure binary conversion. If segment fixups are necessary (that is, if the program contains instructions requiring segment relocation), **exe2bin** prompts you for the fixup value. This value is the absolute segment at which the program is to be loaded. The resulting program is usable only when loaded at the absolute memory address specified by your program. The command interpreter cannot load the program.
- If the CS:IP is specified as 0000:100H, the file runs as a .COM file with the instruction pointer set at 100H by the assembler statement **org**. Include the .COM extension in the *output-file* parameter. No segment fixups are allowed, because .COM files must be segment-relocatable; that is, they must assume the entry conditions explained in the Microsoft Macro Assembler manuals. The command interpreter can then load and run the program in the same way as it loads and runs the .COM programs supplied on your MS-DOS disk.

■ MS-DOS
☐ Batch
☐ CONFIG.SYS
■ Internal
☐ External
■ Network

Exit

Quits the COMMAND.COM program (the command interpreter) and returns to the program that started COMMAND.COM, if one exists.

Syntax

exit

Notes

Using exit with the command interpreter

When you use the MS-DOS **command** command to start a new command interpreter, you can use the **exit** command to return to the old command interpreter. Also, while running some programs, you can run the MS-DOS command interpreter and then use the **exit** command to return to your program. For more information about command interpreters, see the **command** command.

Using exit when the command program is loaded as permanent

If you start the COMMAND.COM program with the **/p** (permanent) switch, the **exit** command has no effect.

■ MS-DOS
☐ Batch
☐ CONFIG.SYS
☐ Internal
■ External
■ Network

Expand

Expands a compressed MS-DOS version 5.0 file.

You can use this command to retrieve one or more files from the installation or update disks that accompany MS-DOS version 5.0. These files are not usable unless you expand them.

Syntax

expand [*drive*:][*path*]*filename* [[*drive*:][*path*]*filename*[...]] *destination*

Parameters

[*drive*:][*path*]*filename*

Specifies the location and name of a compressed file or files to be expanded. You cannot use wildcards (* and ?).

destination

Specifies the location and/or name of the expanded file or files. *Destination* can consist of a drive letter and colon, a directory name, a filename, or a combination. However, you cannot specify a filename for *destination* unless you also specify a single compressed file for filename.

Note

Most of the files on the installation or update disks provided with MS-DOS version 5.0 are compressed. Each of these compressed files has a file extension that ends with an underscore character (_). When you installed MS-DOS, you ran the Setup program, which expanded these files before copying them to your system. However, if you need to retrieve just one or a few files from the original disks, you can use the **expand** command.

Example

Suppose you accidentally delete the SORT.EXE file from your DOS directory on drive C.

You can copy the compressed file SORT.EX_ from the MS-DOS version 5.0 installation or upgrade disks to your hard disk. First, you need to find out which disk contains SORT.EX_. To do this, you first insert Disk 2 (5.25-inch disks) or Disk 1 (3.5-inch disks) in drive A and use a text editor to open the file PACKING.LST. PACKING.LST is not a compressed file. This file lists the contents of each disk. SORT.EX_ is located on Disk 5 (5.25-inch disks) or Disk 3 (3.5-inch disks). You then insert the appropriate disk into drive A and use the following command to copy and expand the file:

```
expand a:\sort.ex_ c:\dos\sort.exe
```

■ MS-DOS
☐ Batch
☐ CONFIG.SYS
☐ Internal
■ External
☐ Network

Fastopen

Starts the Fastopen program, which decreases the amount of time needed to open frequently used files.

Fastopen tracks the location of files on a hard disk and stores the information in memory for fast access. For an introduction to the Fastopen command, see Chapter 12, "Optimizing Your System."

Syntax

fastopen *drive*:[[=]*n*] [*drive*:[[=]*n*][...]] [**/x**]

In your CONFIG.SYS file, use the following syntax:

install=[[*dos-drive*:]*dos-path*]**fastopen.exe** *drive*:[[=]*n*] [*drive*:[[=]*n*][...]] [**/x**]

Parameters

drive:

Specifies a hard disk drive for which you want Fastopen to track the opening of files.

n Specifies the number of files Fastopen can work with at the same time. Valid values for *n* are in the range 10 through 999. The default value is 48.

[*dos-drive*:]*dos-path*

Specifies the location of FASTOPEN.EXE.

Switch

/x Creates the name cache in expanded memory instead of in conventional memory. The *name cache* is an area of memory in which MS-DOS stores (caches) the locations and names of the files that you open. This cache conforms to version 4.0 of the Lotus/Intel/Microsoft Expanded Memory Specification (LIM EMS).

Notes

How Fastopen tracks information

Every time you open a file, Fastopen records its name and location in the name cache. If you later reopen a file recorded by Fastopen, the access time is greatly reduced.

Limits on using Fastopen

Fastopen works only on hard disks and does not work over a network. You can use Fastopen with as many as 24 hard-disk partitions at one time. For each partition, Fastopen can track the number of files specified by the *n* parameter.

You cannot run more than one copy of Fastopen at the same time. If you want to change the Fastopen settings, you must restart MS-DOS.

You should not use the **fastopen** command from MS-DOS Shell, because doing so can lock up your machine.

Memory requirements for Fastopen

Fastopen requires approximately 48 bytes of memory for each file that it tracks.

Adding the fastopen command to the CONFIG.SYS file

You can add a Fastopen command to your CONFIG.SYS file by using the **install** command. Use this technique when you do not want to start Fastopen from the MS-DOS command line or from your AUTOEXEC.BAT file. For more information about starting Fastopen by using the CONFIG.SYS file, see Chapter 12, "Optimizing Your System."

Cannot run disk-compaction program

To avoid losing data, do not run a disk-compaction program while FASTOPEN.EXE is loaded.

Example

If you want MS-DOS to track the location of as many as 100 files on drive C, add the following line to your CONFIG.SYS file:

```
install=c:\dos\fastopen.exe c:=100
```

- ■ MS-DOS
- ☐ Batch
- ☐ CONFIG.SYS
- ☐ Internal
- ■ External
- ■ Network

Fc

Compares two files and displays the differences between them.

For an introduction to the **fc** command, see Chapter 4, "Working with Files."

Syntax

To make an ASCII comparison, use the following syntax:

fc [**/a**] [**/c**] [**/l**] [**/lb***n*] [**/n**] [**/t**] [**/w**] [*/nnnn*] [*drive1*:][*path1*]*filename1* [*drive2*:][*path2*] *filename2*

To make a binary comparison, use the following syntax:

fc /b [*drive1*:][*path1*]*filename1* [*drive2*:][*path2*]*filename2*

Parameters

[*drive1*:][*path1*]*filename1*

Specifies the location and name of the first file you want to compare.

[*drive2*:][*path2*]*filename2*

Specifies the location and name of the second file you want to compare.

Switches

/a Abbreviates the output of an ASCII comparison. Instead of displaying all the lines that are different, **fc** displays only the first and last line for each set of differences.

/c Ignores the case of letters.

/l Compares the files in ASCII mode. **Fc** compares the two files line by line and attempts to resynchronize the files after finding a mismatch. This is the default mode for comparing files that do not have extensions of .EXE, .COM, .SYS, .OBJ, .LIB, or .BIN.

/lb*n*

Sets the number of lines for the internal line buffer. The default length of the line buffer is 100 lines. If the files being compared have more than this number of consecutive differing lines, **fc** cancels the comparison.

/n Displays the line numbers during an ASCII comparison.

/t Does not expand tabs to spaces. The default behavior is to treat tabs as spaces, with stops at each eighth character position.

/w Compresses white space (tabs and spaces) during the comparison. If a line contains many consecutive spaces or tabs, the **/w** switch treats these characters as a single space. When used with the **/w** switch, **fc** ignores (and does not compare) white space at the beginning and end of a line.

/nnnn

Specifies the number of consecutive lines that must match before **fc** considers the files to be resynchronized. If the number of matching lines in the files is less than this number, **fc** displays the matching lines as differences. The default value is 2.

/b Compares the files in binary mode. **Fc** compares the two files byte by byte and does not attempt to resynchronize the files after finding a mismatch. This is the default mode for comparing files that have extensions of .EXE, .COM, .SYS, .OBJ, .LIB, or .BIN.

Notes

Reporting differences between files for an ASCII comparison

When you use **fc** for an ASCII comparison, MS-DOS reports differences between two files by displaying the name of the first file, followed by the lines from *filename1* that differ between the files, followed by the first line to match in both files. MS-DOS then displays the name of the second file, followed by the lines from *filename2* that differ, followed by the first line to match.

Using the /b switch for binary comparisons

MS-DOS uses the following format to report mismatches found during a binary comparison:

xxxxxxxx: *yy zz*

The value of *xxxxxxxx* specifies the relative hexadecimal address for the pair of bytes, measured from the beginning of the file. Addresses start at 00000000; the hexadecimal values for *yy* and *zz* represent the mismatched bytes from *filename1* and *filename2*, respectively.

Using wildcards

You can use wildcards (? and *) in either of the filenames you specify with the **fc** command. If you use a wildcard in *filename1*, **fc** compares all the specified files to the file specified by *filename2*. If you use a wildcard in *filename2*, **fc** uses the corresponding value from *filename1*.

How fc uses memory

When comparing ASCII files, **fc** uses an internal buffer (large enough to hold 100 lines) as storage. If the files are larger than the buffer, **fc** compares what it can load into the buffer. If **fc** does not find a match in the loaded portions of the files, it stops and displays the following message:

```
Resynch failed. Files are too different.
```

When comparing binary files that are larger than available memory, **fc** compares both files completely, overlaying the portions in memory with the next portions from the disk. The output is the same as that for files that fit completely in memory.

Examples

Suppose you want to make an ASCII comparison of two text files that are named MONTHLY.RPT and SALES.RPT, and you want to display the results in abbreviated format. To make this comparison, type the following command:

```
fc /a monthly.rpt sales.rpt
```

To make a binary comparison of two batch files named PROFITS.BAT and EARNINGS.BAT, type the following command:

```
fc /b profits.bat earnings.bat
```

The results of this command will be similar to the following:

```
00000002: 72 43
00000004: 65 3A
0000000E: 56 92
00000012: 6D 5C
00000013: 0D 7C
00000014: 0D 0A
00000015: 0A 0D
0000001E: 43 7A
0000001F: 09 0A
00000022: 72 44
     ...
     ...
     ...
000005E0: 00 61
000005E1: 00 73
000005E2: 00 73
000005E3: 00 69
000005E4: 00 67
000005E5: 00 6E
000005E6: 00 6D
000005E7: 00 65
000005E8: 00 6E
FC: EARNINGS.BAT longer than PROFITS.BAT
```

If the PROFITS.BAT and EARNINGS.BAT files were identical, **fc** would display the following message:

```
FC: no differences encountered
```

To compare every .BAT file in the current directory with the file NEW.BAT, type the following command:

```
fc *.bat new.bat
```

To compare the file NEW.BAT on drive C with the file NEW.BAT on drive D, type the following command.

```
fc c:new.bat d:*.bat
```

To compare each batch file in the root directory on drive C to the file with the same name in the root directory on drive D, type the following command:

```
fc c:*.bat d:*.bat
```

☐ MS-DOS
☐ Batch
■ CONFIG.SYS
☐ Internal
☐ External
☐ Network

Fcbs

Specifies the number of file control blocks (FCBs) that MS-DOS can have open at the same time.

A file control block is a data structure that stores information about a file.

For an introduction to using the CONFIG.SYS file, see Chapter 11, "Customizing Your System."

Syntax

fcbs=*x*

Parameter

x Specifies the number of file control blocks that MS-DOS can have open at one time. Valid values for *x* are in the range 1 through 255. The default value is 4.

Notes

Limitation on opening files
If a program tries to open more than *x* files by using file control blocks, MS-DOS might close the files that were opened earlier.

Recommended use of the fcbs command
You should use the **fcbs** command only if a program requires you to do so. Most newer programs do not require file control blocks. However, some older programs might require you to use the **fcbs** command in your CONFIG.SYS file.

Example

To specify that MS-DOS can have up to eight file control blocks open at the same time, add the following line to your CONFIG.SYS file:

```
fcbs=8
```

■ MS-DOS
☐ Batch
☐ CONFIG.SYS
☐ Internal
■ External
☐ Network

Fdisk

Starts the Fdisk program, which configures a hard disk for use with MS-DOS.

Fdisk displays a series of menus to help you partition your hard disk for MS-DOS. For an introduction to using the Fdisk program and configuring a hard disk, see Chapter 6, "Managing Disks."

Syntax

fdisk

Notes

Using Fdisk to partition a hard disk

You can use Fdisk for the following tasks:

- Creating a primary MS-DOS partition
- Creating an extended MS-DOS partition
- Changing the active partition
- Deleting a partition
- Displaying partition data
- Selecting the next hard disk for partitioning, if a system has multiple hard disks.

Changing the size of a partition

To change the size of a partition, you must actually delete the partition and create a new one with a different size.

Maximum partition size

The maximum partition size is 2 gigabytes.

CAUTION Deleting a partition deletes all the data stored on that partition.

Using Fdisk with an assigned drive

Fdisk does not work on a drive formed by using the **assign**, **subst**, or **join** command.

☐ MS-DOS
☐ Batch
■ CONFIG.SYS
☐ Internal
☐ External
☐ Network

Files

Sets the number of files that MS-DOS can access at one time.

For an introduction to the **files** command, see Chapter 11, "Customizing Your System."

Syntax

files=*x*

Parameter

x Specifies the number of files that MS-DOS can access at one time. Valid values for *x* are in the range 8 through 255. The default value is 8.

Note

Although the default setting for the *x* parameter is 8, some programs require a larger value. A typical setting is 20.

Example

To specify that MS-DOS can access up to 20 files at one time, add the following line to your CONFIG.SYS file:

```
files=20
```

- ■ MS-DOS
- □ Batch
- □ CONFIG.SYS
- □ Internal
- ■ External
- ■ Network

Find

Searches for a specific string of text in a file or files.

After searching the specified files, **find** displays any lines of text that contain the specified string. For an introduction to the **find** command, see Chapter 4, "Working with Files."

Syntax

find [/v] [/c] [/n] [/i] "*string*" [[*drive*:][*path*]*filename*[...]]

Parameters

"*string*"

Specifies the group of characters you want to search for. You must enclose the text for *string* in quotation marks.

[*drive*:][*path*]*filename*

Specifies the location and name of the file in which to search for the specified string.

Switches

/v Displays all lines not containing the specified string.

/c Displays only a count of the lines that contain the specified string.

/n Precedes each line with the file's line number.

/i Specifies that the search is not to be case-sensitive.

Notes

Specifying a string

Unless you specify the **/i** switch, **find** searches for exactly what you specify for *string*. For example, to the **find** command, the characters "a" and "A" are different. If you were to use the **/i** switch, however, **find** would ignore case and search for "a" and "A" as if they were the same character.

If the string you want to search for contains quotation marks, you must use two quotation marks for each quotation mark contained within the string.

Using find as a filter

If you omit a filename, **find** acts as a filter, taking input from the MS-DOS standard source (usually the keyboard, a pipe, or a redirected file) and displaying any lines that contain *string*. For more information about using filters, see Chapter 7, "Advanced Command Techniques."

Using wildcards with find

You cannot use wildcards (* and ?) in filenames or extensions that you specify with the **find** command. To search for a string in a set of files you specify with wildcards, you can use the **find** command in a **for** command.

Using the /v or /n switch with the /c switch

If you specify the **/c** and **/v** switches in the same command, **find** displays a count of the lines that do not contain the specified string. If you specify the **/c** and **/n** switches in the same command, **find** ignores the **/n** switch.

Using find in files with carriage returns

The **find** command does not recognize carriage returns. When you use **find** to search for text in a file that includes carriage returns, you must limit the search string to text that can be found between carriage returns—that is, a string that is not likely to be interrupted by a carriage return. For example, **find** does not report a match for the string "tax file" wherever a carriage return occurs between the word "tax" and the word "file".

Examples

To display all lines from the file PENCIL.AD that contain the string "Pencil Sharpener", type the following command:

```
find "Pencil Sharpener" pencil.ad
```

To find a string that contains text within quotation marks, you must enclose the entire string in quotation marks and, in addition, use two quotation marks for each quotation mark contained within the string, as shown in the following example:

```
find "The scientists labeled their paper ""for discussion only."" It
is not a final report." report.doc
```

If you want to search for a set of files, you can use the **find** command with the **for** command. The following command uses this method to search the current directory for files that have the extension .BAT; in each file found, the command searches for the string "PROMPT":

```
for %f in (*.bat) do find "PROMPT" %f
```

Suppose you want **find** to search your hard disk to find and display the filenames on drive C that contain the string "CPU". To do this you can use the pipe (¦) to direct the results of a **dir** command to **find**, as shown in the following example:

```
dir c:\ /s /b ¦ find "CPU"
```

Before using a pipe for redirection, you should set the TEMP environment variable in your AUTOEXEC.BAT file.

Since **find** searches are case-sensitive and since **dir** produces uppercase output, you must either type the string "CPU" in uppercase letters or use the **/i** switch with **find**.

- ■ MS-DOS
- ■ Batch
- □ CONFIG.SYS
- ■ Internal
- □ External
- □ Network

For

Runs a specified command for each file in a set of files.

You can use the **for** command within a batch program or directly from the command prompt. For an introduction to using batch programs, see Chapter 10, "Working with Batch Programs."

Syntax

To use **for** in a batch program, use the following syntax:

for *%%variable* **in** (*set*) **do** *command* [*command-parameters*]

To use **for** from the command prompt, use the following syntax:

for *%variable* **in** (*set*) **do** *command* [*command-parameters*]

Parameters

%%variable or *%variable*

Represents a replaceable variable. The **for** command replaces *%%variable* (or *%variable*) with each text string in the specified set until the command (specified in the *command* parameter) processes all the files. Use *%%variable* to carry out the **for** command within a batch program. Use *%variable* to carry out **for** from the command prompt.

(set)

Specifies one or more files or text strings that you want to process with the specified command. The parentheses are required.

command

Specifies the command that you want to carry out on each file included in the specified set.

command-parameters

Specifies any parameters or switches that you want to use with the specified command (if the specified command uses any parameters or switches).

Notes

Using the in and do keywords

In and **do** are not parameters, but they are required in the **for** command. If you omit either of these keywords, MS-DOS displays an error message.

Using the replaceable variable

To avoid confusion with the batch parameters **%0** through **%9**, you can use any character for *variable* except the numerals 0 through 9. For simple batch programs, a single character such as **%%f** may be all that is necessary.

You can use multiple values for *variable* in complex batch programs to distinguish different replaceable variables. However, you cannot nest (add) multiple **for** commands on the same command line.

Specifying a group of files

The *set* parameter can represent a single group of files or several groups of files. You can use wildcards (* and ?) to specify a file set. The following are valid file sets:

```
(*.doc)

(*.doc *.txt *.me)

(jan*.doc jan*.rpt feb*.doc feb*.rpt)

(ar??1991.* ap??1991.*)
```

When you use the **for** command, the first value in *set* replaces *%%variable* (or *%variable*) and MS-DOS carries out the specified command in order to process this value; this continues until MS-DOS has processed all the files (or groups of files) that correspond to the value (or values) in *set*.

Examples

Suppose you want to use the **type** command to display the contents of all the files in the current directory that have the extension .DOC or .TXT. To do this and to use the replaceable variable **%f**, type the following command at the command prompt:

```
for %f in (*.doc *.txt) do type %f
```

In this example, each file that has the .DOC or .TXT extension in the current directory is substituted for the **%f** variable until the contents of every file are displayed. To use this command in a batch file, you simply replace every occurrence of **%f** with **%%f**. Otherwise, MS-DOS ignores the variable and displays an error message.

MS-DOS supports command switches, pipes, and redirection that you may want to use with the specified command. For example, to redirect the output of the previous example to PRN (the default printer port), you would type the following command:

```
for %f in (*.doc *.txt) do type %f > prn:
```

Format

■ MS-DOS
☐ Batch
☐ CONFIG.SYS
☐ Internal
■ External
☐ Network

Formats the disk in the specified drive to accept MS-DOS files.

The **format** command creates a new root directory and file allocation table for the disk. It can also check for bad areas on the disk, and it can delete all data on the disk. In order for MS-DOS to be able to use a new disk, you must first use this command to format the disk. For an introduction to the **format** command, see Chapter 6, "Managing Disks."

Syntax

format *drive*: [**/v**[:*label*]] [**/q**] [**/u**] [**/f:***size*] [**/b**|**/s**]

format *drive*: [**/v**[:*label*]] [**/q**] [**/u**] [**/t:***tracks* **/n:***sectors*] [**/b**|**/s**]

format *drive*: [**/v**[:*label*]] [**/q**] [**/u**] [**/1**] [**/4**] [**/b**|**/s**]

format *drive*: [**/q**] [**/u**] [**/1**] [**/4**] [**/8**] [**/b**|**/s**]

Parameter

drive:

Specifies the drive containing the disk you want to format. If you do not specify any of the following switches, **format** uses the drive type to determine the default format for the disk.

Switches

/v:*label*

Specifies the volume label. A volume label identifies the disk and can be a maximum of 11 characters. If you omit the **/v** switch, or use it without specifying a volume label, MS-DOS prompts you for the volume label after the formatting is completed. If you format more than one disk by using one **format** command, all of the disks will be given the same volume label. The **/v** switch is not compatible with the **/8** switch. For more information about disk volume labels, see the **dir**, **label**, and **vol** commands.

/q Deletes the file allocation table (FAT) and the root directory of a previously formatted disk, but does not scan the disk for bad areas. You should use the **/q** switch to format only previously formatted disks that you know are in good condition.

/u Specifies an unconditional format operation for a floppy disk or hard disk. Unconditional formatting destroys all existing data on a disk and prevents you from later "unformatting" the disk. You should use **/u** if you have received read and write errors during use of the disk. For information about unformatting a disk, see the **unformat** command.

/f:*size*

Specifies the size of the floppy disk to format. When possible, use this switch instead of the **/t** and **/n** switches. Use one of the following values for *size*:

160 or **160k** or **160kb**

160K, single-sided, double-density, 5.25-inch disk

180 or **180k** or **180kb**

180K, single-sided, double-density, 5.25-inch disk

320 or **320k** or **320kb**

320K, double-sided, double-density, 5.25-inch disk

360 or **360k** or **360kb**

360K, double-sided, double-density, 5.25-inch disk

720 or **720k** or **720kb**

720K, double-sided, double-density, 3.5-inch disk

1200 or **1200k** or **1200kb** or **1.2** or **1.2m** or **1.2mb**

1.2-MB, double-sided, quadruple-density, 5.25-inch disk

1440 or **1440k** or **1440kb** or **1.44** or **1.44m** or **1.44mb**

1.44-MB, double-sided, quadruple-density, 3.5-inch disk

2880 or **2880k** or **2880kb** or **2.88** or **2.88m** or **2.88mb**

2.88-MB, double-sided, 3.5-inch disk

/b Reserves space for the system files IO.SYS and MSDOS.SYS on a newly formatted disk (as hidden files). In previous versions of MS-DOS, it was necessary to reserve this space before using the **sys** command to copy the system files to the disk. This switch is maintained in MS-DOS version 5.0 for compatibility reasons only.

/s Copies the operating system files IO.SYS, MSDOS.SYS, and COMMAND.COM from your system's startup drive to the newly formatted disk. If **format** cannot find the operating system files, it prompts you to insert a system disk.

/t:*tracks*

Specifies the number of tracks on the disk. When possible, use the **/f** switch instead of this switch. If you use the **/t** switch, you must also use the **/n** switch. These two switches provide an alternative method of specifying the size of the disk being formatted. You cannot use the **/f** switch with the **/t** switch.

/n:*sectors*

Specifies the number of sectors per track. When possible, use the **/f** switch instead of this switch. If you use the **/n** switch, you must also use the **/t** switch. These two switches provide an alternative method of specifying the size of the disk being formatted. You cannot use the **/f** switch with the **/n** switch.

/1 Formats a single side of a floppy disk.

/4 Formats a 5.25-inch, 360K, double-sided, double-density floppy disk on a 1.2-MB disk drive. Some 360K drives cannot reliably read disks formatted with this switch. When used with the **/1** switch, this switch formats a 5.25-inch, 180K, single-sided floppy disk.

/8 Formats a 5.25-inch disk with 8 sectors per track. This switch formats a floppy disk to be compatible with MS-DOS versions prior to 2.0.

Notes

Typing a volume label

After formatting a floppy disk, **format** displays the following message:

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

The volume label can be a maximum of 11 characters (including spaces). If you do not want your disk to have a volume label, just press ENTER. For information about volume labels, see the **label** command.

Formatting a hard disk

When you use the **format** command to format a hard disk, MS-DOS displays a message of the following form *before* attempting to format the hard disk:

```
WARNING, ALL DATA ON NON-REMOVABLE DISK
DRIVE x: WILL BE LOST!
Proceed with Format (Y/N)?_
```

To format the hard disk, press Y; if you do not want to format the disk, press N.

Format messages

When formatting is complete, MS-DOS displays messages showing the total disk space, any space marked as defective, the total space used by the operating system (if you used the **/s** or **/b** switch), and the space available for your files.

Safe formatting

If you do not specify the **/u** switch or a switch that reformats the disk to a different size, **format** performs a "safe" format, meaning that it clears the file allocation table and root directory of the disk but does not delete any data. You can then use the **unformat** command to recover the disk if you did not originally intend to format the disk. **Format** also checks each sector on the disk to ensure that the sector can properly store data. If it locates a sector that cannot store data, **format** marks that sector to prevent MS-DOS from using it.

If you specify the **/u** switch or any switch that changes the size of the disk, **format** performs an unconditional format operation by deleting all data on the disk.

Quick formatting

You can speed up the formatting process by using the **/q** switch. Use this switch only if you have not received read or write errors on your hard disk. You can speed up the process even more by using both the **/q** and **/u** switches. If you use the **/u** switch, **format** does not save the information necessary to later unformat the disk.

Formatting a new disk

When you use **format** to format a disk that has never been formatted, specify the **/u** switch to minimize formatting time.

Using format with a reassigned drive or a network drive

You should not use the **format** command on a drive prepared by using the **assign**, **join**, or **subst** command. You cannot format disks over a network.

Format exit codes

The following list shows each exit code and a brief description of its meaning:

0 The format operation was successful.

3 The user pressed CTRL+C to stop the process.

4 A fatal error occurred (any error other than 0, 3, or 5).

5 The user pressed N in response to the prompt "Proceed with Format (Y/N)?" to stop the process.

You can check these exit codes by using the **errorlevel** condition with the **if** batch command. For an example of a batch program that supports **errorlevel** conditions, see the **backup** command.

Examples

To format a new floppy disk in drive A, using the default size, type the following command:

```
format a:
```

To perform a quick format operation on a previously formatted disk in drive A, type the following command:

```
format a: /q
```

To format a floppy disk in drive A, completely deleting all data on the disk, type the following command:

```
format a: /u
```

To format a 360K floppy disk in drive A and copy the operating system to it, type the following command:

```
format a: /f:360 /s
```

To format a floppy disk in drive A and assign it the volume label "DATA", type the following command:

```
format a: /v:DATA
```

Related Command

For information about restoring disks after using the **format** command, see the **unformat** command.

☐ MS-DOS
■ Batch
☐ CONFIG.SYS

■ Internal
☐ External

☐ Network

Goto

Directs MS-DOS to a line in a batch program marked by a label you specify.

The **goto** command directs MS-DOS within a batch program to a line identified by a label. When MS-DOS finds the label, it processes the commands beginning on the next line. For an introduction to using the **goto** command and batch programs, see Chapter 10, "Working with Batch Programs."

Syntax

goto *label*

Parameter

label

Specifies the line in a batch program to which MS-DOS should go.

Notes

Valid values for *label*

The *label* parameter can include spaces but cannot include other separators, such as semicolons or equal signs.

Goto uses the first eight characters of each label

The **goto** command uses only the first eight characters of a label. Therefore, the labels "hithere01" and "hithere02" are both equivalent to "hithere0".

Matching the *label* parameter with the label in the batch program

The *label* value you specify on the **goto** command line must match a label in the batch program. The label within the batch program must begin with a colon.

If your batch program does not contain the label that you specify, the batch program stops and MS-DOS displays the following message:

```
Label not found
```

MS-DOS recognizes a batch-program line beginning with a colon (:) as a label and does not process it as a command. If a line begins with a colon, MS-DOS ignores any commands on that line.

Using goto for conditional operations

Goto is often used on the same command line with other commands to perform conditional operations. For more information about using **goto** for conditional operations, see the **if** command.

Example

The following batch program formats a disk in drive A as a system disk. If the operation is successful, the **goto** command directs MS-DOS to a label named "end".

```
echo off
format a: /s
if not errorlevel 1 goto end
echo An error occurred during formatting.
:end
echo End of batch program.
```

- ■ MS-DOS
- □ Batch
- □ CONFIG.SYS
- □ Internal
- ■ External
- ■ Network

Graftabl

Enables MS-DOS to display the extended characters of a specified code page in graphics mode.

Most monitors can display extended characters (ASCII characters 128 through 255) without the **graftabl** command. Use this command only if your monitor does not properly display these characters in graphics mode.

Syntax

graftabl [*xxx*]

graftabl /status

Parameter

xxx Specifies the code page for which you want MS-DOS to define the appearance of extended characters in graphics mode. The following list shows each valid code-page identification number and its country or language:

437	United States
850	Multilingual (Latin I)
852	Slavic (Latin II)
860	Portuguese
863	Canadian-French
865	Nordic

Switch

/status

Identifies the code page selected for use by **graftabl**.

Notes

Graftabl does not change the active code page

Graftabl affects only the appearance of extended characters of the code page you specify. To change the code page you are using, use the **mode** or **chcp** command. For more information, see Chapter 13, "Customizing for International Use."

Graftabl exit codes

The following list shows each exit code and a brief description of its meaning:

0 Character set was loaded successfully; no previous code page was loaded.

1 Character set was already loaded and replaced by new table.

2 A file error occurred.

3 An incorrect parameter was specified; no action was taken.

4 An incorrect version of MS-DOS is in use; version 5.0 is required.

You can use the **errorlevel** parameter on the **if** command line in a batch program to process exit codes returned by **graftabl**. For an example of a batch program that processes exit codes, see the **backup** command. For more information about batch programs, see Chapter 10, "Working with Batch Programs."

Effect on memory

The **graftabl** command decreases the amount of available conventional memory by about 1K.

Example

To load the graphics character set for code page 437 into memory, type the following command:

```
graftabl
```

Related Commands

For information about using code pages, see the **chcp** and **mode** (set device code pages) commands.

■ MS-DOS
☐ Batch
☐ CONFIG.SYS

☐ Internal
■ External

■ Network

Graphics

Loads a program into memory that allows MS-DOS to print on a printer the displayed contents of the screen when you are using a color or graphics adapter.

The **graphics** command supports the CGA, EGA, and VGA graphics display modes.

Syntax

graphics [*type*] [[*drive*:][*path*]*filename*] [**/r**] [**/b**] [**/lcd**] [**/printbox:std**|**/printbox:lcd**]

Parameters

type Specifies the type of printer. The following list shows each valid value for this parameter and a brief description of its meaning:

color1

An IBM Personal Computer Color Printer with black ribbon

color4

An IBM Personal Computer Color Printer with RGB (red, green, blue, and black) ribbon

color8

An IBM Personal Computer Color Printer with CMY (cyan, magenta, yellow, and black) ribbon

hpdefault

Any Hewlett-Packard PCL printer

deskjet

A Hewlett-Packard DeskJet printer

graphics

An IBM Personal Graphics Printer, IBM Proprinter, or IBM Quietwriter printer

graphicswide

An IBM Personal Graphics Printer with an 11-inch-wide carriage

laserjet

A Hewlett-Packard LaserJet printer

laserjetii

A Hewlett-Packard LaserJet II printer

paintjet

A Hewlett-Packard PaintJet printer

quietjet

A Hewlett-Packard QuietJet printer

quietjetplus

A Hewlett-Packard QuietJet Plus printer

ruggedwriter

A Hewlett-Packard RuggedWriter printer

ruggedwriterwide

A Hewlett-Packard RuggedWriterwide printer

thermal

An IBM PC-convertible Thermal Printer

thinkjet

A Hewlett-Packard ThinkJet printer

[drive:][path]filename

Specifies the location and name of the printer profile that contains information about all supported printers. If this parameter is omitted, MS-DOS looks for a file called GRAPHICS.PRO in the current directory and in the directory that contains the GRAPHICS.COM file.

Switches

/r Prints the image as it appears on the screen (white characters on a black background) rather than reversed (black characters on a white background). The latter occurs by default.

/b Prints the background in color. This switch is valid for **color4** and **color8** printers.

/lcd

Prints an image by using the liquid crystal display (LCD) aspect ratio instead of the CGA aspect ratio. The effect of this switch is the same as that of **/printbox:lcd**.

/printbox:std|**/printbox:lcd**

Selects the print-box size. You can abbreviate **printbox** as **pb**. You should check the first operand of the **printbox** statement in your GRAPHICS.PRO file and specify the **/printbox:std** switch if that operand is **std** or the **/printbox:lcd** switch if that operand is **lcd**.

Notes

Printing the contents of the screen

To print the contents of the screen, press the SHIFT+PRINT SCREEN key combination. If the computer is in 320 × 200 color graphics mode and if the printer type is **color1** or **graphics**, the **graphics** command prints the screen contents with as many as four shades of gray. If the computer is in 640 × 200 color graphics mode, **graphics** prints the screen contents sideways on the paper (landscape orientation). You cannot use the SHIFT+PRINT SCREEN key combination to print the contents of a screen to a PostScript printer.

Effect on memory

The **graphics** command decreases the amount of available conventional memory.

Loading a new profile

If you have already loaded a printer profile and you want to load another one by using the **graphics** command, the new profile must be smaller than the one already loaded.

To load a new profile that is larger than the one currently loaded, you must restart your system and then use the **graphics** command to load the new profile.

If you try to use only the **graphics** command to load a new profile that is larger than the currently loaded profile, MS-DOS displays the following message:

```
Unable to reload with profile supplied
```

Example

To prepare to print a graphics screen on your printer, type the following command:

```
graphics
```

After you display the information you want to print, press SHIFT+PRINT SCREEN. MS-DOS scans the information displayed on the screen and sends it to the printer.

Related Command

For information about printing text files, see the **print** command.

- ■ MS-DOS
- □ Batch
- □ CONFIG.SYS
- □ Internal
- ■ External
- ■ Network

Help

Provides online information about the MS-DOS version 5.0 commands.

The information that this command displays is similar to, but less detailed than, that found in this chapter.

Syntax

help [*command*]

Parameter

command

Specifies the name of the command about which you want information. If you do not specify a command name, the **help** command lists and briefly describes every command provided with MS-DOS version 5.0.

Note

There are two ways to get online Help for a command. You can specify the name of the command on the **help** command line, or you can type the name of the command and the **/?** switch at the command prompt. For example, you can type either of the following commands to get information about the **xcopy** command:

```
help xcopy

xcopy /?
```

The second command is slightly faster.

☐ MS-DOS
■ Batch
☐ CONFIG.SYS
■ Internal
☐ External
☐ Network

If

Performs conditional processing in batch programs.

If the condition specified in an **if** command is true, MS-DOS carries out the command that follows the condition. If the condition is false, MS-DOS ignores the command. For an introduction to using the **if** command and batch programs see Chapter 10, "Working with Batch Programs."

Syntax

if [**not**] **errorlevel** *number command*

if [**not**] *string1*==*string2 command*

if [**not**] **exist** *filename command*

Parameters

not Specifies that MS-DOS should carry out the command only if the condition is false.

errorlevel *number*

Specifies a true condition only if the previous program run by COMMAND.COM returned an exit code equal to or greater than *number*.

command

Specifies the command that MS-DOS should carry out if the preceding condition is met.

string1==*string2*

Specifies a true condition only if *string1* and *string2* are the same. These values can be literal strings or batch variables (**%1**, for example). Literal strings do not need quotation marks.

exist *filename*

Specifies a true condition if *filename* exists.

Note

When a program stops, it returns an exit code to MS-DOS. The **errorlevel** parameter lets you use exit codes as conditions. For examples of using the **errorlevel** parameter to process exit codes, see the following "Examples" section and the **backup** command.

Examples

The following example displays the message "Can't find data file" if MS-DOS cannot find the file PRODUCT.DAT:

```
if not exist product.dat echo Can't find data file
```

The following example displays an error message if an error occurs during formatting of the disk in drive A. If no error occurs, the error message is skipped.

```
:begin
echo off
format a: /s
if not errorlevel 1 goto end
echo An error occurred during formatting.
:end
echo End of batch program.
```

The following example tests for the existence of a directory. The **if** command cannot be used to test directly for a directory, but the null (NUL) device does exist in every directory. Therefore, you can test for the null device to determine whether a directory exists.

```
if exist c:\mydir\nul goto process
```

- ☐ MS-DOS
- ☐ Batch
- ■ CONFIG.SYS
- ☐ Internal
- ☐ External
- ☐ Network

Install

Loads a memory-resident program into memory when you start MS-DOS.

Memory-resident programs stay in memory as long as your system is on. They can be used even when other programs are active. You can use the **install** command to load MS-DOS memory-resident programs such as Fastopen, Keyb, Nlsfunc, and Share. For an introduction to using the CONFIG.SYS file, see Chapter 11, "Customizing Your System."

Syntax

install=[*drive*:][*path*]*filename* [*command-parameters*]

Parameters

[*drive*:][*path*]*filename*

Specifies the location and name of the memory-resident program you want to run.

command-parameters

Specifies parameters for the program you specify for *filename*.

Note

Install does not create an environment for a program it loads. Therefore, slightly less memory is used if you load a program with **install** rather than from your AUTOEXEC.BAT file. Some programs might not run correctly if they are loaded with **install**. Do not use **install** to load programs that use environment variables or shortcut keys or that require COMMAND.COM to be present to handle critical errors.

Example

Suppose you want to install FASTOPEN.EXE, located in the DOS directory on drive C, from your CONFIG.SYS file instead of from your AUTOEXEC.BAT file or the command line. In addition, you want to specify that Fastopen is to track the opening of up to 100 files and directories on drive C. To do this, include the following command in your CONFIG.SYS file:

```
install=c:\dos\fastopen.exe c:=100
```

■ MS-DOS	
☐ Batch	
☐ CONFIG.SYS	
☐ Internal	
■ External	
☐ Network	

Join

Joins a disk drive to a directory on another disk drive.

When you use the **join** command, MS-DOS treats the directories and files on a disk drive as the contents of the other drive and path you specify.

Syntax

join [*drive1*: [*drive2*:]*path*]

join *drive*: **/d**

Parameters

drive1:

Specifies the floppy disk drive or logical drive that you want to join to a different drive and directory.

drive2:

Specifies the floppy disk drive or logical drive to which you want to join *drive1*.

path

Specifies the directory to which you want to join *drive1*. This directory must be empty before you join *drive1* to it. It must also be a directory other than the root directory.

drive:

Specifies a floppy disk drive or logical drive that was previously specified in a **join** command that you are now canceling.

Switch

/d Cancels any previous **join** commands for the drive you specify.

Notes

Drive1 becomes invalid

After you use the **join** command, the *drive1* you specify becomes invalid. If you then try to use it, MS-DOS displays the following message:

```
Invalid drive specification
```

Limitations on *path*

If the directory specified by *path* already exists before you use the **join** command, you cannot use that directory for any other purpose while **join** is in effect. If the directory is not empty, MS-DOS does not complete the join operation and displays the following message:

```
Directory not empty
```

If the directory does not exist, MS-DOS tries to create it.

Limitations on using join with other commands

The following commands do not work with drives formed by the **join** command:

assign	**diskcopy**	**mirror**
backup	**fdisk**	**recover**
chkdsk	**format**	**restore**
diskcomp	**label**	**sys**

Using join with no parameters

You can use the **join** command with no parameters to see a list of the currently joined drives.

Examples

You can join any directory or subdirectory in a tree structure. For example, the following commands are valid:

```
join d: c:\sales

join d: c:\sales\october
```

To reverse either of the previous **join** commands, type the *drive1* value followed by the **/d** switch, as follows:

```
join d: /d
```

Related Commands

For information about redirecting disk operations from one drive to another, see the **assign** command.

For information about substituting a drive letter for a directory name, see the **subst** command.

- ■ MS-DOS
- □ Batch
- □ CONFIG.SYS
- □ Internal
- ■ External
- ■ Network

Keyb

Starts the Keyb program, which configures a keyboard for a specific language.

Use Keyb to configure a keyboard for a language other than United States English. For an introduction to the Keyb program, see Chapter 13, "Customizing for International Use."

Syntax

keyb [*xx*[,[*yyy*][,[*drive*:][*path*]*filename*]]] [**/e**] [**/id**:*nnn*]

In your CONFIG.SYS file, use the following syntax:

install=[[*dos-drive*:]*dos-path*]**keyb.com** [*xx*[,[*yyy*][,[*drive*:][*path*]*filename*]]] [**/e**] [**/id**:*nnn*]

Parameters

xx Specifies the keyboard code.

yyy Specifies the code page.

[*drive*:][*path*]*filename*
Specifies the location and name of the keyboard definition file. The default filename is KEYBOARD.SYS.

[*dos-drive*:]*dos-path*
Specifies the location of the KEYB.COM file.

Switches

/e Specifies that an enhanced keyboard is installed. Use this switch if you are using an enhanced keyboard with an 8086 computer.

/id:*nnn*
Specifies the keyboard in use. This switch is necessary only for countries that have more than one keyboard layout for the same language (France, Italy, and the United Kingdom).

Notes

Values for *xx*, *yyy*, and *nnn*

The following table shows the valid values for *xx*, *yyy*, and *nnn* for each country or language:

Country or language	Keyboard code (*xx* value)	Code page (*yyy* value)	Keyboard identification (*nnn* value)
Belgium	be	850, 437	
Brazil	br	850, 437	
Canadian-French	cf	850, 863	
Czechoslovakia (Czech)	cz	852, 850	
Czechoslovakia (Slovak)	sl	852, 850	
Denmark	dk	850, 865	
Finland	su	850, 437	
France	fr	850, 437	120, 189
Germany	gr	850, 437	
Hungary	hu	852, 850	
Italy	it	850, 437	141, 142
Latin America	la	850, 437	
Netherlands	nl	850, 437	
Norway	no	850, 865	
Poland	pl	852, 850	
Portugal	po	850, 860	
Spain	sp	850, 437	
Sweden	sv	850, 437	
Switzerland (French)	sf	850, 437	
Switzerland (German)	sg	850, 437	
United Kingdom	uk	850, 437	166, 168
United States	us	850, 437	
Yugoslavia	yu	852, 850	

Installing code pages

The code page you specify for *yyy* must be installed on your system. For information about installing a code page, see Chapter 13, "Customizing for International Use."

Displaying the current keyboard code and code page

If you use the **keyb** command with no parameters or switches, MS-DOS lists the current keyboard code, the current keyboard's related code page, and the current code page used by your console (CON). The information is displayed in the following format:

```
Current keyboard code: FR code page: 437
Current CON code page: 437
```

Switching between Keyb settings

You can switch from the current Keyb keyboard configuration to the default keyboard configuration at any time by pressing CTRL+ALT+F1. To return to the memory-resident keyboard configuration, press CTRL+ALT+F2. You can switch to "typewriter mode," the standard for some countries, by pressing CTRL+ALT+F7.

Implementing Keyb

The following list shows the three different ways that you can start the Keyb program:

- Type **keyb** at the command prompt.
- Include an **install** command for KEYB.COM in your CONFIG.SYS file.
- Include the appropriate **keyb** command in your AUTOEXEC.BAT file.

Keyb exit codes

The following list shows each exit code and a brief description of its meaning:

0 Keyboard definition file was loaded successfully.

1 Invalid keyboard code, code page, or syntax was used.

2 Keyboard definition file is bad or missing.

4 An error occurred while communicating with the CON device.

5 The requested code page has not been prepared.

You can use the **errorlevel** parameter on the **if** command line in a batch program to process exit codes returned by Keyb. For an example of a batch program that processes exit codes, see the **diskcomp** command. For more information about batch programs, see Chapter 10, "Working with Batch Programs."

Example

If you want to use a German keyboard and your KEYBOARD.SYS file is in the DOS directory on drive C, type the following command:

```
keyb gr,,c:\dos\keyboard.sys
```

Related Command

For information about using active and prepared code pages, see the **chcp** command.

■ MS-DOS
☐ Batch
☐ CONFIG.SYS
☐ Internal
■ External
☐ Network

Label

Creates, changes, or deletes the volume label (name) of a disk.

MS-DOS displays the volume label as part of the directory listing. If a volume serial number exists, MS-DOS displays this number as well. For an introduction to the **label** command, see Chapter 6, "Managing Disks."

Syntax

label [*drive*:][*label*]

To specify that MS-DOS is to display the current volume label and serial number, if they exist, and that MS-DOS is to prompt you to enter a label or delete the existing one, use the following syntax:

label

Parameters

drive:

Specifies the location of the disk you want to name.

label

Specifies the new volume label. You must include a colon (:) between *drive* and *label*.

Notes

Label command messages

If you do not specify a label when you use the **label** command, MS-DOS displays a message in the following format:

```
Volume in drive A is xxxxxxxxxx
Volume Serial Number is xxxx-xxxx
Volume label (11 characters, ENTER for none)?
```

The "Volume Serial Number" part of the message is not displayed if the disk has no serial number.

You can type the volume label you want or press ENTER to delete the current label. If a disk has a label and you press ENTER for none, MS-DOS prompts you with the following message:

```
Delete current volume label (Y/N)?
```

Press Y to delete the label; press N to keep the label.

Limitations on volume label names

A volume label can contain as many as 11 characters and can include spaces but no tabs. Consecutive spaces may be interpreted as a single space.

Do not use any of the following characters in a volume label:

```
* ? / \ | . , ; : + =   [ ] ( ) & ^ < > "
```

MS-DOS displays volume labels in uppercase letters. If you enter a volume label in lower-case letters, the **label** command converts the letters to uppercase.

Using label with a redirected drive

Label does not work on a drive created with the **assign**, **join**, or **subst** command.

Example

To label a disk in drive A that contains sales information for 1991, you might type the following:

```
label a:sales1991
```

Related Commands

For information about displaying the current disk label, see the **dir** or **vol** command.

For information about the volume serial number of a disk, see the **vol** command.

☐ MS-DOS
☐ Batch
■ CONFIG.SYS
☐ Internal
☐ External
☐ Network

Lastdrive

Specifies the maximum number of drives you can access.

The value you specify represents the last valid drive MS-DOS is to recognize. For an introduction to using the **lastdrive** command and the CONFIG.SYS file, see Chapter 11, "Customizing Your System."

Syntax

lastdrive=*x*

Parameter

x Specifies a drive letter in the range A through Z. The minimum value for this parameter is the letter that corresponds to the number of drives installed on your system. For example, one drive = A, two drives = B, and so on.

Notes

Default setting
The last drive, by default, is the one after the last drive being used on your computer.

Effect on memory
MS-DOS allocates a data structure in memory for each drive specified by **lastdrive**, so you should not specify more drives than are necessary.

Example

The following command sets the last drive to M, giving your computer access to 13 logical drives:

```
lastdrive=m
```

■ MS-DOS
☐ Batch
☐ CONFIG.SYS

■ Internal
☐ External

■ Network

Loadhigh (lh)

Loads a program into the upper memory area.

Loading a program into the upper memory area leaves more room in conventional memory for other programs. For an introduction to the **loadhigh** command, see Chapter 12, "Optimizing Your System."

Syntax

loadhigh [*drive*:][*path*]*filename* [*parameters*]

lh [*drive*:][*path*]*filename* [*parameters*]

Parameters

[*drive*:][*path*]*filename*
Specifies the location and name of the program you want to load.

parameters
Specifies any command-line information required by the program.

Notes

Dos=umb command required

To use the **loadhigh** command, you must include the **dos=umb** command in your CONFIG.SYS file. For more information about the **dos=umb** command, see the **dos** command.

Upper-memory-area manager must be installed

Before you can load a program into the upper memory area, you must install an upper-memory-area manager. MS-DOS provides EMM386.EXE, which manages the upper memory area for computers with an 80386 or higher processor. Before you install EMM386.EXE, you must install the HIMEM.SYS extended-memory manager. Use the **device** command in your CONFIG.SYS file to install the HIMEM.SYS and EMM386.EXE. These commands must appear before the **loadhigh** command. For more information about using the upper memory area, see Chapter 12, "Optimizing Your System."

How loadhigh works

If you use the **loadhigh** command to load a program, MS-DOS attempts to load it into the upper memory area. If there is insufficient space in the upper memory area, MS-DOS loads the program into conventional memory. MS-DOS does not indicate which memory area is used.

Using loadhigh in your AUTOEXEC.BAT file

The most convenient way to use the **loadhigh** command is to include it in your AUTOEXEC.BAT file. For more information about using AUTOEXEC.BAT, see Chapter 11, "Customizing Your System."

- ■ MS-DOS
- ☐ Batch
- ☐ CONFIG.SYS
- ☐ Internal
- ■ External
- ■ Network

Mem

Displays the amount of used and free memory in your system.

You can use the **mem** command to display information about allocated memory areas, free memory areas, and programs that are currently loaded into memory.

Syntax

mem [**/program**|**/debug**|**/classify**]

To display the status of your system's used and free memory, use the following syntax:

mem

Switches

/program

Displays the status of programs that are currently loaded into memory. You cannot use the **/program** switch with the **/debug** switch or the **/classify** switch. You can abbreviate **/program** as **/p**.

/debug

Displays the status of currently loaded programs and of internal drivers, and displays other programming information. You cannot use the **/debug** switch with the **/program** switch or the **/classify** switch. You can abbreviate **/debug** as **/d**.

/classify

Displays the status of programs loaded into conventional memory and the upper memory area. This switch lists the size of each program in decimal and hexadecimal notation, provides a summary of memory use, and lists the largest memory blocks that are available. You cannot use the **/classify** switch with the **/program** switch or the **/debug** switch. You can abbreviate **/classify** as **/c**.

Notes

Displaying memory status

MS-DOS displays the status of extended memory only if you have installed memory above the 1-megabyte (MB) boundary in your system. MS-DOS displays the status of expanded memory only if you use expanded memory that conforms to version 4.0 of the Lotus/Intel/Microsoft Expanded Memory Specification (LIM EMS).

Allocating extended memory

To allocate Interrupt 15h memory and XMS memory at the same time, use the **/int15** switch when you load the HIMEM.SYS device driver. For more information, see Chapter 15, "Device Drivers."

Example

Suppose your system has both expanded memory and extended memory. To display the status of your system's total memory—conventional, expanded, extended—and to display a list of programs currently loaded into memory, type the following command:

```
mem /program
```

The results might look similar to the following:

```
Address      Name          Size       Type
-------      --------      ------     ------
000000                     000400     Interrupt Vector
000400                     000100     ROM Communication Area
000500                     000200     DOS Communication Area
```

```
000700      IO            000A80     System Data

001180      MSDOS         0014F0     System Data

002670      IO            006280     System Data
              HIMEM       0004A0      DEVICE=
              EMM386      002410      DEVICE=
              DISPLAY     002050      DEVICE=
                          0005D0      FILES=
                          000100      FCBS=
                          000200      BUFFERS=
                          0008F0      LASTDRIVE=
                          000740      STACKS=
008900      MSDOS         000040     System Program

008950      COMMAND       000940     Program
0092A0      MSDOS         000040     -- Free --
0092F0      COMMAND       000200     Environment
009500      MEM           000190     Environment
0096A0      NISAMR        001FB0     Program
00B660      MEM           0135A0     Program
01EC10      MSDOS         0813D0     -- Free --
09FFF0      MSDOS         028010     System Program

0C8010      MSDOS         000040     -- Free --
0C8060      MOUSE         003A80     Program
0CBAF0      MSDOS         000190     -- Free --
0CBC90      MSDOS         000AE0     -- Free --
0CC780      XNSBIOS       000180     Environment
0CC910      XNSBIOS       002610     Program
0CEF30      SESSION       000180     Environment
0CF0C0      PRTSC         000180     Environment
0CF250      PRTSC         000320     Program
0CF580      DOSKEY        000FE0     Program
0D0570      MSDOS         007A60     -- Free --
0D7FE0      MSDOS         008020     System Program

0E0010      IO            0069B0     System Data
              VT52        001060      DEVICE=
              SMARTDRV    005930      DEVICE=
0E69D0      SESSION       000410     Program
0E6DF0      REDIR         009200     Program

  656384 bytes total conventional memory
  655360 bytes available to MS-DOS
  608640 largest executable program size
 3145728 bytes total contiguous extended memory
       0 bytes available contiguous extended memory
 1027072 bytes available XMS memory
         MS-DOS resident in High Memory Area
```

Total conventional memory is the amount of memory on your computer up to the first 640K. *Available to MS-DOS* is the amount of conventional memory MS-DOS has for operating your computer, including the memory it needs for itself. *Largest executable program size* is the largest contiguous block of conventional memory available for a program.

Total EMS memory (not shown in the preceding example) is the amount of expanded memory on your computer. *Free EMS memory* (not shown in the preceding example) is the amount of expanded memory available for programs. If you use EMM386 to simulate expanded memory, that memory appears in these two values.

Total contiguous extended memory is the amount of memory beyond the 1-megabyte (MB) boundary on your computer. *Available contiguous extended memory* is the extended memory available for the Interrupt 15h interface. This memory is not being managed by an extended-memory manager, such as HIMEM.SYS. Some older programs use this different extended-memory scheme. *Available XMS memory* is memory that is being managed by an extended-memory manager, such as HIMEM.SYS, and that is available to programs that can use it.

Related Command

For information about checking the amount of space available on a disk, see the **chkdsk** command.

■ MS-DOS
□ Batch
□ CONFIG.SYS
□ Internal
■ External
■ Network

Mirror

Starts the Mirror program, which records information about one or more disks; the **unformat** and **undelete** commands can use this information to restore a reformatted disk or to recover deleted files.

For an introduction to the Mirror program, see Chapter 4, "Working with Files," and Chapter 6, "Managing Disks."

Syntax

mirror [*drive*:[...]] [**/1**] [**/t***drive*[-*entries*][...]]

mirror [**/u**]

mirror [**/partn**]

To save information about the disk in the current drive, use the following syntax:

mirror

Parameter *drive*:

Specifies the drive containing the disk for which you want Mirror to save information. This information is used by the **unformat** command to restore a disk.

Switches **/1** Retains only the latest information about the disk. If you do not specify this switch, Mirror makes a backup copy of the existing disk-information file before recording the current information.

/t*drive*[*-entries*]

Loads a terminate-and-stay-resident deletion-tracking program that records information used by the **undelete** command to recover deleted files. The required *drive* parameter specifies the drive containing the disk for which you want Mirror to save information about deleted files. The optional *entries* parameter, which must be a value in the range 1 through 999, specifies the maximum number of entries in the deletion-tracking file (PCTRACKR.DEL). The default value for *entries* is dependent upon the type of disk being tracked.

The following list shows each disk size, its default number of entries, and its corresponding file size:

Disk size	Entries	File size
360K	25	5K
720K	50	9K
1.2 megabyte (MB)	75	14K
1.44 MB	75	14K
20 MB	101	18K
32 MB	202	36K
>32 MB	303	55K

CAUTION Do not use deletion tracking for any drive that has been redirected by using the **join** or **subst** command. If you intend to use the **assign** command, you must do so before using Mirror to install deletion tracking.

/u Unloads the deletion-tracking program from memory, disabling deletion tracking. You cannot unload the tracking program if you loaded any other memory-resident programs after it.

/partn

Saves system information about how a hard disk is partitioned. The switch saves the information in a file on a floppy disk. The **unformat** command can use this file later to rebuild the partitions of a disk.

Notes

Saving information about a disk

The Mirror program saves a copy of the file allocation table and the root directory of the disk in the specified drive. The **unformat** command can use this information to rebuild a disk that has been unintentionally formatted, or it can use the information to recover files and subdirectories in the disk's root directory.

Because **unformat** restores the disk's system area to the condition it was in when you last used Mirror, you should save this information frequently for every hard disk drive in your system. To ensure that the information is saved each time you turn on your computer, you may want to add a **mirror** command to your AUTOEXEC.BAT file.

Removing the deletion-tracking program from memory

You may need to remove the deletion-tracking program from memory. To do so, remove all memory-resident programs that you loaded after the deletion-tracking program, and then use the **mirror** command with the **/u** switch. Since this turns off deletion tracking, any files deleted after you remove the tracking program can be recovered only by using information in the directory.

Saving information about hard-disk partitions

Every formatted hard disk drive has at least one partition. To identify a hard disk drive, MS-DOS uses information stored in a special disk partition table. If this table is corrupted, MS-DOS cannot locate the hard disk.

You can save partition-table information for a hard disk by using the **mirror** command with the **/partn** switch. This switch creates a file named PARTNSAV.FIL, which the **unformat** command can use to rebuild the partition table. Because MS-DOS cannot gain access to your hard disk if the partition table is damaged, you should *not* put this file on the hard disk itself. Instead, you should put the file on a floppy disk (which you should keep in a safe place) or on another hard disk drive, such as a network server.

Examples

To save a copy of the file allocation table and the root directory of drive C and to install deletion tracking for drives A and C, type the following command:

```
mirror c: /ta /tc
```

Suppose you want to save a copy of the file allocation table and the root directory of the disk in the current drive, and you want to install the deletion-tracking program for drive C. To do this and to set the maximum number of deletions to be tracked to 500, type the following command. (Note that since no *drive* parameter is specified, Mirror saves the information about the disk in the current drive.)

```
mirror /tc-500
```

To save a copy of the partition table for your hard disk drive, type the following command:

```
mirror /partn
```

The Mirror program displays the following information:

```
Disk Partition Table saver.

The partition information from your hard drive(s) has been read.

Next, the file PARTNSAV.FIL will be written to a floppy disk. Please
insert a formatted diskette and enter the name of the diskette drive.
What drive? A
```

The default disk drive is drive A. If you want to use a different drive, type the drive letter (making sure it does not identify a partition on the hard disk drive), insert a formatted floppy disk in the drive (if necessary), and press ENTER.

Related Commands

For information about retrieving deleted files, see the **undelete** command.

For information about restoring formatted disks, see the **unformat** command.

■ MS-DOS
☐ Batch
☐ CONFIG.SYS
■ Internal
☐ External
■ Network

Mkdir (md)

Creates a directory.

You can use the **mkdir** command to create a multilevel directory structure. For an introduction to the **mkdir** command, see Chapter 5, "Working with Directories."

Syntax

mkdir [*drive*:]*path*

md [*drive*:]*path*

Parameters

drive:

Specifies the drive on which you want to create the new directory.

path

Specifies the name and location of the new directory. The maximum length of any single path from the root directory to the final directory is 63 characters, including backslashes (\).

Examples

Suppose you want to create a directory on the disk in the current drive and use the directory to store all your tax information. To create a directory named TAXES, type the following command:

```
mkdir \taxes
```

Now suppose that the TAXES directory is the current directory and that you want to create a subdirectory of TAXES named PROPERTY. To create the PROPERTY directory, type the following command:

```
mkdir property
```

Related Commands

For information about deleting a directory, see the **rmdir** command.

For information about changing directories, see the **chdir** command.

- ■ MS-DOS
- ☐ Batch
- ☐ CONFIG.SYS
- ☐ Internal
- ■ External
- ■ Network

Mode

Configures system devices.

The **mode** command performs many different tasks, such as displaying system status, changing system settings, or reconfiguring ports or devices. For an introduction to the **mode** command, see Chapter 11, "Customizing Your System."

Syntax

Because the **mode** command can perform many different tasks, the syntax necessary to carry out each task is different. Therefore, this reference discusses the tasks separately. For information about the syntax that **mode** uses for a specific task, see the following pages.

Notes

Using the mode command

The following is a list of tasks for which you can use the **mode** command. Following each task is the name of the command description as it appears in this chapter.

- Reconfiguring a printer attached to a parallel port (PRN, LPT1, LPT2, or LPT3) for printing at 80 or 132 characters per line, 6 or 8 lines per inch, or both (if the printer supports these features). See **mode** (configure printer).
- Configuring the baud rate, parity, and number of data bits and stop bits of a serial communications port (COM1, COM2, COM3, and COM4) for use with a specific printer, modem, or other serial device. See **mode** (configure serial port).
- Displaying the status of all devices or of a single device. See **mode** (display device status).

- Redirecting printer output from a parallel port to a serial port so that the serial port becomes the system's default printer port. See **mode** (redirect printing).
- Preparing devices for code-page switching. See **mode** (set device code pages).
- Selecting another display adapter or changing the configuration of the current display adapter. See **mode** (set display mode).
- Setting the keyboard's typematic rate. See **mode** (set typematic rate).

Requirement for ANSI.SYS and DISPLAY.SYS

Mode can perform some tasks, such as setting the display mode, only if you have included a **device** command for the ANSI.SYS device driver in your CONFIG.SYS file. You must install DISPLAY.SYS to use **mode** (set device code pages) for code-page switching.

Adding mode commands to AUTOEXEC.BAT

Although you can type each form of the **mode** command at the command prompt, you can also use **mode** commands within your AUTOEXEC.BAT file to reconfigure your system automatically each time you turn on or restart your computer. For more information about using **mode** commands in the AUTOEXEC.BAT file, see Chapter 11, "Customizing Your System."

■ MS-DOS
☐ Batch
☐ CONFIG.SYS
☐ Internal
■ External
■ Network

Mode (Configure Printer)

Configures a printer connected to a parallel printer port.

This version of the **mode** command sets the characteristics for an IBM-compatible or Epson-compatible printer connected to a parallel printer port (PRN, LPT1, LPT2, or LPT3). For an introduction to configuring printers by using the **mode** command, see Chapter 11, "Customizing Your System."

Syntax

mode lpt*n*[**:**] [*c*][**,**[*l*][**,***r*]]

mode lpt*n*[**:**] [**cols=***c*] [**lines=***l*] [**retry=***r*]

Parameters

lpt*n*

Specifies the parallel port to which the device is attached. Valid values for *n* are in the range 1 through 3.

If you omit any of the following three parameters, **mode** uses the most recent setting for the omitted parameter. If you are using the shorter form of the syntax (without the words **cols**=, **lines**=, and **retry**=), the **mode** command "recognizes" the parameters by their positions. Thus, if you do not specify a value for a parameter, you must still type the comma that precedes the next parameter.

cols=*c*

Specifies the number of characters (columns) per line: 80 or 132. The default value is 80. You can abbreviate this parameter by simply omitting **cols**= and specifying a value for *c* .

lines=*l*

Specifies the vertical spacing and the number of lines per inch: 6 or 8. The default value is 6. You can abbreviate this parameter by simply omitting **lines**= and specifying a value for *l*.

retry=*r*

Specifies the retry action to take if a time-out error occurs when **mode** attempts to send output to a parallel printer. This parameter causes part of **mode** to remain resident in memory. The following list shows each valid value for *r* and a brief description of its meaning:

e	Return an error from a status check of a busy port.
b	Return "busy" from a status check of a busy port.
p	Continue retrying until printer accepts output.
r	Return "ready" from a status check of a busy port.
n	Take no retry action (default value). You can also specify **none** for this value.

If you are using the **mode** command over a network, do not use any of the *r* values.

You can abbreviate this parameter by simply omitting **retry**= and specifying a value for *r*.

Notes

Update to mode parameter

The **retry**=**b** setting provides the same support as the **p** parameter did in previous versions of MS-DOS.

Breaking out of a time-out loop

To break out of a time-out loop, press CTRL+C.

Setting parallel-printer modes

For parallel-printer modes, you can use PRN and LPT1 interchangeably.

Examples

Suppose you want to be able to print 80 characters per line and 8 lines per inch on a parallel printer that is connected to the second parallel printer port (LPT2). To do this, type the following command:

```
mode lpt2:80,8
```

Because 80 characters per line is the default setting, however, you could achieve the same result typing the following command:

```
mode lpt2:,8
```

Suppose that, when printing a file, you want your system to keep trying to print the file until it is successful. To do this, type the following command:

```
mode lpt2:,8,b
```

To stop your system from continually retrying to print, press CTRL+C or type the **mode** command without specifying a value for *r*.

- ■ MS-DOS
- ☐ Batch
- ☐ CONFIG.SYS
- ☐ Internal
- ■ External
- ■ Network

Mode (Configure Serial Port)

Configures a serial communications port.

This version of the **mode** command sets the parameters for a serial port (COM1, COM2, COM3, or COM4). For an introduction to configuring serial ports by using the **mode** command, see Chapter 11, "Customizing Your System."

Syntax

mode com*m*[**:**] [*b*[,*p*[,*d*[,*s*[,*r*]]]]]

mode com*m*[**:**] [**baud=***b*] [**parity=***p*] [**data=***d*] [**stop=***s*] [**retry=***r*]

Parameters

com*m*

Specifies the number of the asynchronous-communications (COM) port. Valid values for *m* are in the range 1 through 4.

If you omit any of the following five parameters, **mode** uses the most recent setting for the omitted parameter. If you are using the shorter form of the syntax (without the words **baud**=, **parity**=, **data**=, and so on), the **mode** command "recognizes" the parameters by their positions. Thus, if you do not specify a value for a parameter, you must still type the comma that precedes the next parameter.

baud=*b*

Specifies the first two digits of the transmission rate in bits per second. The following list shows each valid value for *b* and its related rate:

11	110 baud
15	150 baud
30	300 baud
60	600 baud
12	1200 baud
24	2400 baud
48	4800 baud
96	9600 baud
19	19,200 baud

The *b* value of 19 is not supported on all computers (check your hardware manual). You can abbreviate this parameter by simply omitting **baud**= and specifying a value for *b*.

parity=*p*

Specifies how the system uses the parity bit to check for transmission errors. The *p* value can be one of the following: **n** (none), **e** (even), **o** (odd), **m** (mark), or **s** (space). The default value is **e**. Not all computers support the values **m** and **s**. You can abbreviate this parameter by simply omitting **parity**= and specifying a value for *p*.

data=*d*

Specifies the number of data bits in a character. Valid values for *d* are in the range 5 through 8. The default value is 7. Not all computers support the values 5 and 6. You can abbreviate this parameter by simply omitting **data**= and specifying a value for *d*.

stop=*s*

Specifies the number of stop bits that define the end of a character: 1, 1.5, or 2. If the baud rate is 110, the default value is 2; otherwise, the default value is 1. Not all computers support the value 1.5. You can abbreviate this parameter by simply omitting **stop**= and specifying a value for *s*.

retry=*r*

Specifies the retry action to take if a time-out error occurs when **mode** attempts to send output to a parallel printer. This parameter causes part of **mode** to remain resident in memory. The following list shows each valid value for *r* and a brief description of its meaning:

e	Return an error from a status check of a busy port.
b	Return "busy" from a status check of a busy port.
p	Continue retrying until printer accepts output.
r	Return "ready" from a status check of a busy port.
n	Take no retry action (default value). You can also specify **none** for this value.

If you are using the **mode** command over a network, do not use any of the *r* values.

You can abbreviate this parameter by simply omitting **retry=** and specifying a value for *r*.

Note

The **retry=b** setting provides the same support as the **p** parameter did in previous versions of MS-DOS.

- ■ MS-DOS
- ☐ Batch
- ☐ CONFIG.SYS
- ☐ Internal
- ■ External
- ■ Network

Mode (Display Device Status)

Displays the status of one or all of the devices installed on your system.

Syntax

mode [*device*] [**/status**]

To display the status of all devices installed on your system, use the following syntax:

mode

Parameter

device

Specifies the name of the device for which you want to display the status.

Switch **/status**

Requests the status of any redirected parallel printers. The **mode** command, when used without this switch, displays the status of all installed devices except redirected parallel printers. You can abbreviate the **/status** switch as **/sta**.

Example To display the status of the console, type the following command:

```
mode con
```

■ MS-DOS
☐ Batch
☐ CONFIG.SYS
☐ Internal
■ External
■ Network

Mode (Redirect Printing)

Redirects output from a parallel port to a serial communications port.

For an introduction to configuring printers by using the **mode** command, see Chapter 11, "Customizing Your System."

Syntax **mode lpt*n*[:]=com*m*[:]**

Parameters **lpt*n***

Specifies the parallel port. Valid values for *n* are in the range 1 through 3.

com*m*

Specifies the serial port. Valid values for *m* are in the range 1 through 4.

Examples Suppose you want to set up your system so that it sends parallel-printer output to a serial printer. To do this, you must use the **mode** command twice. The first time, you use **mode** to configure the serial port; the second time, you use **mode** to redirect parallel-printer output to the serial port you specified in the first **mode** command.

For example, if your serial printer operates at 4800 baud with even parity and is connected to the COM1 port (the first serial connection on your computer), you would type the following two commands:

```
mode com1 48,e,,,b

mode lpt1=com1
```

If you redirect parallel-printer output from LPT1 to COM1 but then decide that you want to print a file by using LPT1, use the following command before you print the file. This command prevents MS-DOS from redirecting the file from LPT1 to COM1.

```
mode lpt1
```

- ■ MS-DOS
- □ Batch
- □ CONFIG.SYS
- □ Internal
- ■ External
- ■ Network

Mode (Set Device Code Pages)

Prepares, selects, refreshes, or displays the numbers of the code pages for parallel printers or the console.

For an introduction to using code pages, see Chapter 13, "Customizing for International Use."

Syntax

mode *device* **codepage prepare**=((*yyy* [...]) [*drive*:][*path*]*filename*)

mode *device* **codepage select**=*yyy*

mode *device* **codepage refresh**

mode *device* **codepage** [**/status**]

Parameters

device

Specifies the device for which you want to prepare or select a code page. Valid names for *device* are **con**, **lpt1**, **lpt2**, and **lpt3**.

codepage prepare

Prepares code pages for the specified device. You must prepare a code page for a device before you can use the code page with that device. After you use the **codepage prepare** form of the **mode** command, use the **codepage select** form of **mode** to specify the code page you want to use. You can abbreviate **codepage** and **prepare** as **cp** and **prep**, respectively.

yyy

Specifies the number of the code page to prepare or select. The following list shows each code page that MS-DOS supports and its country or language:

437 United States
850 Multilingual (Latin I)
852 Slavic (Latin II)
860 Portuguese
863 Canadian-French
865 Nordic

[*drive*:][*path*]*filename*

Specifies the location and name of the code-page-information (.CPI) file that MS-DOS uses to prepare a code page for the specified device.

codepage select

Specifies (selects) which code page to use with the specified device. Before selecting a code page, you must use the **codepage prepare** form of the **mode** command to prepare a code page. You can abbreviate **codepage** and **select** as **cp** and **sel**, respectively.

codepage refresh

Reinstates the prepared code pages if they are lost as the result of a hardware problem or other error. You can abbreviate **codepage** and **refresh** as **cp** and **ref**, respectively.

codepage

Displays the numbers of the code pages, if any, that are prepared or selected for the specified device.

Switch

/status

Displays the numbers of the current code pages prepared or selected for the specified device. You can abbreviate this switch as **/sta**. Whether or not you specify the **/status** switch, typing the **mode** command with a device name and the **codepage** parameter displays the numbers of the code pages that are prepared or selected for the specified device.

Note

MS-DOS includes five .CPI files, which correspond to specific devices, as follows:

File	Device
EGA.CPI	Enhanced graphics adapter (EGA) or IBM Personal System/2
4201.CPI	IBM Proprinters II and III Model 4201 IBM Proprinters II and III XL Model 4202
4208.CPI	IBM Proprinter X24E Model 4207 IBM Proprinter XL24E Model 4208
5202.CPI	IBM Quietwriter III printer
LCD.CPI	IBM PC Convertible liquid crystal display

Related Commands

For information about other code-page commands, see the **nlsfunc** and **chcp** commands.

■ MS-DOS
☐ Batch
☐ CONFIG.SYS
☐ Internal
■ External
■ Network

Mode (Set Display Mode)

Selects the active display adapter and its display mode, or reconfigures the active display adapter.

For an introduction to using the **mode** command, see Chapter 11, "Customizing Your System."

Syntax

mode [*display-adapter*][,*shift*[,**t**]]

mode [*display-adapter*][,*n*]

mode con[**:**] [**cols**=*c*] [**lines**=*n*]

Parameters

display-adapter

Specifies a setting category. The following list shows the value(s) associated with each setting category for *display-adapter*:

40 or **80**

Indicates the number of characters per line.

bw40 or **bw80**

Specifies a color graphics adapter (CGA) with color disabled, and specifies the number of characters per line.

co40 or **co80**

Specifies a color monitor with color enabled, and specifies the number of characters per line.

mono

Specifies a monochrome display adapter with a constant width of 80 characters per line.

shift

Specifies whether to shift the CGA screen to the left or to the right. Valid values for *shift* are **l** (for left) and **r** (for right).

t Enables you to align the screen by using a test pattern. MS-DOS prompts you to indicate whether the screen is aligned correctly.

con[**:**]

Refers to the monitor.

cols=*c*

Specifies the number of characters (columns) per line. Valid values are 40 and 80.

lines=*n*

Specifies the number of lines that can be displayed on the screen. Valid values for *n* are 25, 43, and 50. Not all display adapters support all three settings. To set the number of lines, you must have installed the ANSI.SYS device driver by using a **device** command in your CONFIG.SYS file.

■ MS-DOS
□ Batch
□ CONFIG.SYS
□ Internal
■ External
■ Network

Mode (Set Typematic Rate)

Sets the keyboard typematic rate, the rate at which MS-DOS repeats a character when you hold down the key for that character.

The typematic rate has two components, the rate and the delay. Some keyboards do not recognize this command.

Syntax

mode con[**:**] [**rate**=*r* **delay**=*d*]

Parameters

con[**:**]

Refers to the keyboard.

rate=*r*

Specifies the rate at which a character is repeated on the screen when you hold down a key. Valid values are in the range 1 through 32. These values are equal to approximately 2 to 30 characters per second, respectively. The default value is 20 for IBM AT-compatible keyboards, and 21 for IBM PS/2-compatible keyboards. If you set the rate, you must also set the delay.

delay=*d*

Specifies the amount of time that must elapse—after you press and hold down a key—before MS-DOS starts to repeat the character. Valid values for *d* are 1, 2, 3, and 4 (representing 0.25, 0.50, 0.75, and 1 second, respectively). The default value is 2. If you set the delay, you must also set the rate.

■ MS-DOS
☐ Batch
☐ CONFIG.SYS
☐ Internal
■ External
■ Network

More

Displays one screen of output at a time.

The **more** command reads standard input from a pipe or redirected file and displays one screen of information at a time. This command is commonly used to view long files. For an introduction to the **more** command, see Chapter 7, "Advanced Command Techniques."

Syntax

more < [*drive*:][*path*]*filename*

command-name ¦ **more**

Parameters

[*drive*:][*path*]*filename*

Specifies the location and name of a file that supplies data you want to display.

command-name

Specifies the command that supplies data you want to display.

Notes

Sources of data

When using the redirection character (<), you must specify a filename as the source. When using the pipe (¦), you can use such commands as **dir**, **sort**, and **type**. Before using a pipe for redirection, you should set the TEMP environment variable in your AUTOEXEC.BAT file.

Using the redirection characters

For an introduction to using the redirection characters with the command, see Chapter 7, "Advanced Command Techniques."

Examples

Suppose you have a long file named CLIENTS.NEW that you want to view on your screen. Either of the following two commands redirects the file through the **more** command to begin displaying the contents of the file:

```
more < clients.new

type clients.new | more
```

The **more** command displays the first screen of information from CLIENTS.NEW and then prompts you with the following message:

```
-- More --
```

You can then press any key to see the next screen of information.

Related Commands

For information about displaying the contents of a directory, see the **dir** command.

For information about displaying the contents of a file, see the **type** command.

- ■ MS-DOS
- □ Batch
- □ CONFIG.SYS
- □ Internal
- ■ External
- ■ Network

Nlsfunc

Starts the Nlsfunc program, which loads country-specific information for national language support (NLS).

You can use the **nlsfunc** command either from the command line or within CONFIG.SYS to support the use of country-specific information and code-page switching. For an introduction to Nlsfunc, see Chapter 13, "Customizing for International Use."

Syntax

nlsfunc [[*drive*:][*path*]*filename*]

In your CONFIG.SYS file, use the following syntax:

install=[[*dos-drive*:]*dos-path*]**nlsfunc.exe** [*country-filename*]

Parameters

[*drive*:][*path*]*filename* or *country-filename*

Specifies the location and name of the file containing country-specific information. If you use this parameter in the **install** command, you must include the drive and directory.

[*dos-drive*:]*dos-path*

Specifies the location of NLSFUNC.EXE.

Note

The default value for [*drive*:][*path*]*filename* is defined by the **country** command in your CONFIG.SYS file. If no **country** command exists in CONFIG.SYS, Nlsfunc looks for COUNTRY.SYS in the root directory of the startup drive. Nlsfunc does not access the COUNTRY.SYS file until MS-DOS requests information from it. If MS-DOS cannot find the COUNTRY.SYS file when you install Nlsfunc, no error message is given. However, you will get an error message if you subsequently run a **chcp** command.

Examples

To use the default country-specific information found in the COUNTRY.SYS file, type the following command:

```
nlsfunc
```

Suppose you have a file called NEWCDPG.SYS that contains country-specific information. If you want to use the information from that file rather than the COUNTRY.SYS file, type the following command:

```
nlsfunc newcdpg.sys
```

Related Commands

For information about displaying the current code page, see the **chcp** command.

For information about preparing a code page, see the **mode** (set device code page) command.

■ MS-DOS
☐ Batch
☐ CONFIG.SYS
■ Internal
☐ External
■ Network

Path

Sets a search path for executable files.

MS-DOS uses the **path** command to search for executable files in the directories you specify. By default, the search path is the current directory only. For an introduction to the **path** command, see Chapter 5, "Working with Directories."

Syntax

path [[*drive*:]*path*[;...]]

To display the current search path, use the following syntax:

path

To clear all search-path settings other than the default setting (the current directory), use the following syntax:

path ;

Parameters

[drive:]path

Specifies a drive, directory, and any subdirectories to search.

;

When used as the only parameter, clears all search-path settings and specifies that MS-DOS is to search only the current directory.

Notes

Current directory

MS-DOS always searches in the current directory first, before it searches directories in the search path.

Length limit for the path command

The maximum length of the path command is 127 characters. To fit more directories in the search path, you can shorten directory names, use the **subst** command to redirect directories to logical drives (which shortens the entries on the **path** command line), or use the **append /x:on** command.

Files with the same name, different extensions

You might have some files in the same directory that share the same filename but have different extensions. For example, you might have a file named ACCNT.COM that starts an accounting program and another file named ACCNT.BAT that connects your system to the accounting system network.

MS-DOS searches for a file by using default filename extensions in the following order of precedence: .COM, .EXE, and .BAT. To run ACCNT.BAT when ACCNT.COM exists in the same directory, you must include the .BAT extension on the command line.

Two or more identical filenames in the path

You might have two or more files in the search path that have the same filename and extension. MS-DOS searches for the specified filename first in the current directory. Then it searches directories in the order in which they are listed in the **path** command.

Specifying multiple paths

To specify more than one path for MS-DOS to search, separate entries with a semicolon (;).

Using path in your AUTOEXEC.BAT file

If you place the **path** command in your AUTOEXEC.BAT file, MS-DOS automatically initiates the specified search path every time you start your computer. For more information about using **path** in your AUTOEXEC.BAT file, see Chapter 11, "Customizing Your System," and Chapter 12, "Optimizing Your System."

Example

The following command specifies that MS-DOS is to search three directories to find external commands (the three paths for these directories are C:\USER\TAXES, B:\USER\INVEST, and B:\BIN):

```
path c:\user\taxes;b:\user\invest;b:\bin
```

Related Command

For information about setting a search path for data files, see the **append** command.

- ☐ MS-DOS
- ■ Batch
- ☐ CONFIG.SYS
- ■ Internal
- ☐ External
- ☐ Network

Pause

Suspends processing of a batch program and displays a message prompting the user to press any key to continue.

For an introduction to using batch programs, see Chapter 10, "Working with Batch Programs."

Syntax

pause

Notes

Prompting the user to continue the program

MS-DOS displays the following message in response to the **pause** command:

```
Press any key to continue . . .
```

Dividing a batch file into sections

If you press CTRL+C to stop a batch program, MS-DOS displays the following message:

```
Terminate batch job (Y/N)?
```

If you press Y (for yes) in response to this message, the batch program ends and control returns to the operating system. Therefore, you can insert the **pause** command before a section of the batch file you may not want to process. While **pause** suspends processing of the batch program, you can press CTRL+C and then Y to stop the batch program.

Example

Suppose you want a batch program to prompt the user to change disks in one of the drives. To do this, you might create the following file:

```
@echo off
:begin
copy a:*.*
echo Please put a new disk into drive A
pause
goto begin
```

In this example, all the files on the disk in drive A are copied to the current directory. After the displayed comment prompts you to place another disk in drive A, the **pause** command suspends processing so that you can change disks and then press any key to resume processing. This particular batch program runs in an endless loop. The **goto begin** command sends the command interpreter to the *begin* label of the batch file. To stop this batch program, press CTRL+C and then Y.

- ■ MS-DOS
- □ Batch
- □ CONFIG.SYS
- □ Internal
- ■ External
- ■ Network

Print

Prints a text file while you are using other MS-DOS commands.

This command can print in the background if you have an output device connected to one of your system's serial or parallel ports. For an introduction to the **print** command, see Chapter 4, "Working with Files."

Syntax

print [**/d**:*device*] [**/b**:*size*] [**/u**:*ticks1*] [**/m**:*ticks2*] [**/s**:*ticks3*] [**/q**:*qsize*] [**/t**] [[*drive*:][*path*]*filename*[...]] [**/c**] [**/p**]

To install **print** with the default parameters or to display the contents of the print queue on your screen without affecting the queue, use the following syntax:

print

Parameter

[*drive*:][*path*]*filename*

Specifies the location and name of a file or set of files you want to print. You can include multiple files (usually as many as 13) on one command line.

Switches

/d:*device*

Specifies the name of the print device. Valid values for parallel ports are **lpt1**, **lpt2**, and **lpt3**. Valid values for serial ports are **com1**, **com2**, **com3**, and **com4**. The default value is **prn**. The values **prn** and **lpt1** refer to the same parallel port. The **/d** switch must precede any filename used on the command line.

/b:*size*

Sets the size (in bytes) of the internal buffer, which is used to store data before it is sent to the printer. The minimum and default value for *size* is 512; the maximum value is 16384. Increasing this value decreases the amount of memory available for other purposes but may speed up the **print** command.

/u:*ticks1*

Specifies the maximum number of clock ticks **print** is to wait for a printer to be available (clock ticks occur about 18 times per second). If the printer is not available within the time specified, the job does not print. Values for *ticks1* must be in the range 1 through 255. The default value is 1.

/m:*ticks2*

Specifies the maximum number of clock ticks **print** can take to print a character on the printer. Values for *ticks2* must be in the range 1 through 255. The default value is 2. If a character is printed too slowly, MS-DOS displays an error message.

/s:*ticks3*

Specifies the number of clock ticks the MS-DOS scheduler allocates for background printing. Values for *ticks3* must be in the range 1 through 255 . The default value is 8. Increasing this value can speed up printing while slowing down other programs.

/q:*qsize*

Specifies the maximum number of files allowed in the print queue. Values for *qsize* must be in the range 4 through 32. The default value is 10.

/t Removes all files from the print queue.

/c Removes files from the print queue. You can use the **/c** and **/p** switches on the same command line.

When the **/c** switch precedes the list of filenames on the command line, it applies to all files whose names follow the **/c** switch, until **print** encounters a **/p** switch, in which case the **/p** switch applies to the file whose name precedes the **/p** switch.

When the **/c** switch follows a filename, it applies to the file whose name precedes the **/c** switch and all files whose names follow the **/c** switch, until **print** encounters a **/p** switch, in which case the **/p** switch applies to the file whose name precedes the **/p** switch.

/p Adds files to the print queue. You can use the **/c** and **/p** switches on the same command line.

When the **/p** switch precedes the list of filenames on the command line, it applies to all files whose names follow the **/p** switch, until **print** encounters a **/c** switch, in which case the **/c** switch applies to the file whose name precedes the **/c** switch.

When the **/p** switch follows a filename, it applies to the file whose name precedes the **/p** switch and all files whose names follow the **/p** switch, until **print** encounters a **/c** switch, in which case the **/c** switch applies to the file whose name precedes the **/c** switch.

Notes

Length of a print queue entry

Each print queue entry can contain a maximum of 64 characters. Each queue entry includes the drive letter, directory, and any subdirectories.

Limitations on switches

You can only use the **/d**, **/b**, **/u**, **/m**, **/s**, and **/q** switches the first time you use the **print** command after starting MS-DOS. To use one of these switches after using **print**, you need to restart MS-DOS.

Printing files generated by programs

Many programs have their own print commands. You should use a program's print command to print files that you create with the program.

Examples

The following command sets up the print queue for printing on LPT1:

```
print /d:lpt1
```

The following command removes the PENCIL.TST file from the print queue:

```
print a:pencil.tst /c
```

The next command shows how to remove the file PENCIL.TST from the queue and add the file PEN.TST to the queue:

```
print pencil.tst /c pen.tst /p
```

The following three examples use switches that work only with the first **print** command you use after starting MS-DOS.

To specify that the **print** command is to wait 60 clock ticks for a printer to be available and that the MS-DOS scheduler is to allocate 25 clock ticks to the **print** command for background printing rather than the default value of 8 clock ticks, type the following command:

```
print /u:60 /s:25
```

The following example specifies that **print** has 4 clock ticks available to print each character rather than the default value of 2 clock ticks:

```
print /m:4
```

To change the default maximum number of files for the print queue, use the **print** command with the **/q** switch, as the following example shows:

```
print /q:32
```

Related Commands

For information about configuring a printer connected to a parallel port, see the **mode** (configure printer) command.

For information about displaying the status of a printer, see the **mode** (display device status) command.

For information about configuring a printer connected to a serial port, see the **mode** (redirect printing) command.

For information about preparing printers for code-page switching, see the **mode** (set device code pages) command.

■ MS-DOS
☐ Batch
☐ CONFIG.SYS
■ Internal
☐ External
■ Network

Prompt

Changes the MS-DOS command prompt.

You can customize the command prompt to display any text you want, including such information as the name of the current directory, the time and date, and the MS-DOS version number. For an introduction to the **prompt** command, see Chapter 5, "Working with Directories."

Syntax

prompt [*text*]

Parameter

text

Specifies any text and information you want included in your system prompt.

The following list shows the character combinations you can include instead of, or in addition to, any character string(s) in the *text* parameter. The list includes a brief description of the text or information that each character combination adds to your command prompt.

$q = (equal sign)
$$ $ (dollar sign)

$t	Current time
$d	Current date
$p	Current drive and path
$v	MS-DOS version number
$n	Current drive
$g	> (greater-than sign)
$l	< (less-than sign)
$b	¦ (pipe)
$_	ENTER-LINEFEED
$e	ASCII escape code (code 27)
$h	Backspace (to delete a character that has been written to the **prompt** command line)

Notes

Using the prompt command without the *text* parameter

When you use the **prompt** command without specifying a value for *text*, **prompt** resets the command prompt to the default setting, the current drive letter followed by a greater-than sign (>).

Using the $p value for *text*

If you include the **$p** character in the *text* parameter, MS-DOS reads your disk after you enter each command to determine the current drive and path. This can take extra time, especially for floppy disk drives.

Examples

The following example sets the command prompt to display the current drive and path followed by the greater-than sign (>):

```
prompt $p$g
```

The following command displays a two-line prompt in which the current time appears on the first line and the current date appears on the second line:

```
prompt time is: $t$_date is: $d
```

If your CONFIG.SYS file loads ANSI.SYS, you can use ANSI escape sequences in your prompts. The following command, for example, displays your prompt in reverse video mode and returns to usual video mode for other text:

```
prompt $e[7m$n:$e[m
```

The characters following the escape code (**$e**) are ANSI escape sequences. For information about ANSI.SYS and ANSI escape sequences, see Chapter 11, "Customizing Your System."

Related Commands

For information about setting the current date and time, see the **date** and **time** commands.

For information about adding the **prompt** command to your AUTOEXEC.BAT file, see Chapter 11, "Customizing Your System."

- ■ MS-DOS
- □ Batch
- □ CONFIG.SYS
- □ Internal
- ■ External
- ■ Network

Qbasic

Starts MS-DOS QBasic, a program that reads instructions written in the Basic computer language and interprets them into executable computer code.

The QBasic program provides a complete environment for programming in the Basic language. QBasic includes extensive online Help. For more information about using QBasic, press ENTER immediately after starting QBasic or press F1 any time while running QBasic.

Syntax

qbasic [**/b**] [**/editor**] [**/g**] [**/h**] [**/mbf**] [**/nohi**] [[**/run**][*drive*:][*path*]*filename*]

Parameter

[*drive*:][*path*]*filename*

Specifies the location and name of the file to load when QBasic starts.

Switches

/b Displays QBasic in black and white if you have a color monitor.

/editor

Invokes MS-DOS Editor, a full-screen text editor.

/g Provides the fastest update of a CGA monitor.

/h Displays the maximum number of display lines possible on your screen.

/mbf

Converts the built-in functions MKS$, MKD$, CVS, and CVD to MKSMBF$, MKDMBF$, CVSMBF, and CVDMBF, respectively.

/nohi

Allows the use of a monitor that does not support high-intensity video. Do not use this switch with COMPAQ laptop computers.

/run

Runs the specified Basic program before displaying it. You must specify a filename.

Notes

Cannot use MS-DOS Editor if MS-DOS QBasic is not present

To use MS-DOS Editor, you must have the QBASIC.EXE file in the current directory or in your search path or in the same directory as the EDIT.COM file. If you delete QBASIC.EXE to save space on your hard disk, you cannot use MS-DOS Editor.

Running consecutive Basic programs

You can run consecutive Basic programs from a batch file by using the Basic **system** statement and the **qbasic** command with the **/run** switch. A **system** statement returns control to MS-DOS after a Basic program has run, instead of returning to QBasic. This allows you to run more than one Basic program from a batch file without having to intervene.

Converting a GW-BASIC program to run in QBasic

For a Basic program that helps convert GW-BASIC programs to QBasic, see the file REMLINE.BAS (included with QBasic).

Display of shortcut keys

Some monitors may not support the display of shortcut keys by default. If your monitor does not display shortcut keys, use the **/b** switch (for CGA monitors) and the **/nohi** switch (for systems that do not support bold characters).

■ MS-DOS
☐ Batch
☐ CONFIG.SYS
☐ Internal
■ External
☐ Network

Recover

Recovers readable information from a bad or defective disk.

The **recover** command reads a file sector by sector and recovers data from the good sectors. Data in bad sectors is lost. All recovered data is restored to the root directory. For an introduction to the **recover** command, see Chapter 6, "Managing Disks."

CAUTION Since the root directory can hold only a finite number of entries, some files may be lost. If you need to recover all files on a disk, recover them one at a time. Do not attempt to recover all the files in a directory or on a disk at one time unless absolutely necessary.

Syntax

recover [*drive*:][*path*]*filename*

To recover all files on a disk when the disk directory is unusable, use the following syntax:

recover *drive*:

Parameters

[drive:][path]filename

Specifies the location and name of the file you want to recover. (Use this syntax to recover a single file.)

drive:

Specifies the drive from which you want to recover all files.

Notes

Limitation on *[drive:][path]filename*

You cannot use wildcards (* and ?) with the **recover** command. You must specify either a file or a drive.

Locating recovered files

When you recover an entire disk, each recovered file is placed in the root directory in a FILE*nnnn*.REC file, where *nnnn* is a 4-digit number. The first recovered file is named FILE0001.REC, the next recovered file is named FILE0002.REC, and so on.

Reentering lost data

Since all data in bad sectors is lost when you recover a file, you should recover files one at a time. This method allows you to edit each file and reenter missing information after you recover the file.

Recovering bad sectors

Bad sectors reported by **chkdsk** were marked as "bad" when your disk was first prepared for operation. They pose no danger, and **recover** will not affect them.

Limitations with networks and assigned drives

The **recover** command cannot recover files on a network drive. **Recover** also does not work on a drive formed by the **assign**, **join**, or **subst** command.

Recover vs. backup and restore

The **recover** command does not work with the **backup** or **restore** command. You must use **restore** with backup files you created by using the **backup** command.

Related Command

For information about checking your disk for bad sectors, see the **chkdsk** command.

☐ MS-DOS
■ Batch
■ CONFIG.SYS
☐ Internal
☐ External
☐ Network

Rem

Enables you to include comments (remarks) in a batch file or in your CONFIG.SYS file.

MS-DOS ignores any batch command or CONFIG.SYS line that begins with **rem**. For an introduction to using batch programs, see Chapter 10, "Working with Batch Programs." For an introduction to using a CONFIG.SYS file, see Chapter 11, "Customizing Your System."

Syntax

rem [*comment*]

Parameter

comment

Specifies any string of characters you want to include as a comment.

Notes

Using the echo command to display comments

The **rem** command does not display comments on the screen. You must use the **echo on** command in your batch or CONFIG.SYS file in order to display comments on the screen.

Restrictions on batch-file comments

You cannot use a redirection character (> or <) or pipe (¦) in a batch-file comment.

Using rem to add vertical spacing

Although you can use **rem** without a comment to add vertical spacing to a batch file, you can also use blank lines. MS-DOS ignores the blank lines when processing the batch program.

Examples

The following example shows a batch file that uses remarks for both explanations and vertical spacing:

```
@echo off
rem  This batch program formats and checks new disks.
rem  It is named CHECKNEW.BAT.
rem
echo Insert new disk in drive B.
pause
format b: /v
chkdsk b:
```

Suppose you want to include in your CONFIG.SYS file an explanatory comment before the **country** command. To do this, add the following lines to CONFIG.SYS:

```
rem Set country code to France
country=033
```

Related Command

For information about displaying messages, see the **echo** command.

- ■ MS-DOS
- ☐ Batch
- ☐ CONFIG.SYS
- ■ Internal
- ☐ External
- ■ Network

Rename (ren)

Changes the name of a file or files.

You can rename all files matching the specified filename. You cannot use the **rename** command to rename files across drives, to move files to a different directory location, or to rename subdirectories. For an introduction to using the **rename** command, see Chapter 4, "Working with Files."

Syntax

rename [*drive*:][*path*]*filename1 filename2*

ren [*drive*:][*path*]*filename1 filename2*

Parameters

[*drive*:][*path*]*filename1*

Specifies the location and name of the file or set of files you want to rename.

filename2

Specifies the new name for the file or, if you use wildcards, the new names for the files. (You cannot specify a new drive or path.)

Notes

Using wildcards with rename

You can use wildcards (* and ?) in either filename parameter. If you use wildcards in *filename2*, the characters represented by the wildcards will be identical to the corresponding characters in *filename1*.

Rename will not work if *filename2* already exists

If, for *filename2*, you specify a filename that already exists, **rename** displays the following message:

```
Duplicate file name or file not found
```

Examples

Suppose you want to change the extensions of all the filenames in the current directory that have the extension .TXT; for example, suppose you want to change the .TXT extensions to .DOC extensions. To make this change, type the following command:

```
ren *.txt *.doc
```

To rename a file named CHAP10 (on drive B) to PART10, type the following command:

```
ren b:chap10 part10
```

The newly renamed file PART10 remains on drive B.

Related Commands

For information about renaming a disk, see the **label** command.

For information about copying files to a different drive or directory, see the **copy** command.

For information about copying entire directories to a new location, see the **xcopy** command.

■ MS-DOS
☐ Batch
☐ CONFIG.SYS

☐ Internal
■ External

■ Network

Replace

Replaces files in the destination directory with files in the source directory that have the same name. You can also use **replace** to add unique filenames to the destination directory. For an introduction to using the **replace** command, see Chapter 5, "Working with Directories."

Syntax

replace [*drive1*:][*path1*]*filename* [*drive2*:][*path2*] [**/a**] [**/p**] [**/r**] [**/w**]

replace [*drive1*:][*path1*]*filename* [*drive2*:][*path2*] [**/p**] [**/r**] [**/s**] [**/w**] [**/u**]

Parameters

[*drive1*:][*path1*]*filename*

Specifies the location and name of the source file or set of files.

[*drive2*:][*path2*]

Specifies the location of the destination file. You cannot specify a filename for files you replace. If you specify neither a drive nor a directory, **replace** uses the current drive and directory as the destination.

Switches

/a Adds new files to the destination directory instead of replacing existing files. You cannot use this switch with the **/s** or **/u** switch.

/p Prompts you for confirmation before replacing a destination file or adding a source file.

/r Replaces read-only files as well as unprotected files. If you do not specify this switch but attempt to replace a read-only file, an error results and stops the replacement operation.

/s Searches all subdirectories of the destination directory and replaces matching files. You cannot use the **/s** switch with the **/a** switch. The **replace** command does not search subdirectories specified in *path1*.

/w Waits for you to insert a disk before **replace** begins to search for source files. If you do not specify **/w**, **replace** begins replacing or adding files immediately after you press ENTER.

/u Replaces (updates) only those files on the destination directory that are older than those in the source directory. You cannot use the */u* switch with the */a* switch.

Notes

Replace messages

As **replace** adds or replaces files, MS-DOS displays their filenames on the screen. After the **replace** command is finished, MS-DOS displays a summary line in one of the following formats:

```
nnn files added
nnn files replaced

no file added
no file replaced
```

Replacing files on a floppy disk system

If you have a floppy disk system and need to switch disks during the **replace** operation, you can specify the **/w** switch so that **replace** will wait for you to switch disks, as necessary.

Limitations on replace

You cannot use the **replace** command to update hidden files or system files such as IO.SYS and MSDOS.SYS. For information about changing hidden and system attributes, see the **attrib** command.

Replace exit codes

The following list shows each exit code and a brief description of its meaning:

0 **Replace** successfully replaced or added the files.

2 **Replace** could not find the source files.

3 **Replace** could not find the source or destination path.

5 The user does not have access to the files you want to replace.

8 There is insufficient system memory to carry out the command.

11 The user used the wrong syntax on the command line.

You can use the **errorlevel** parameter on the **if** command line in a batch program to process exit codes returned by **replace**. For an example of a batch program that processes exit codes, see the **backup** command. For more information about batch programs, see Chapter 10, "Working with Batch Programs."

Examples

Suppose that several directories on drive C contain different versions of a file named PHONES.CLI, which contains client names and phone numbers. To replace all of these files with the latest version of the PHONES.CLI file from the disk in drive A, type the following command:

```
replace a:\phones.cli c:\ /s
```

Suppose you want to add new printer device drivers to a directory on drive C named TOOLS, which already contains several printer device-driver files for a word processor. To do this, type the following command:

```
replace a:*.prd c:\tools /a
```

This command searches the current directory on drive A for any files that have the extension .PRD and then adds these files to the TOOLS directory on drive C. Because the **/a** switch is included, **replace** adds only those files from drive A that do not exist on drive C.

Related Command

For information about changing file attributes, see the **attrib** command.

■ MS-DOS
☐ Batch
☐ CONFIG.SYS
☐ Internal
■ External
■ Network

Restore

Restores files that were backed up by using the **backup** command.

You can restore files from similar or dissimilar disk types. For an introduction to the **restore** command, see Chapter 6, "Managing Disks."

Syntax

restore *drive1*: *drive2*:[*path*[*filename*]] [**/s**] [**/p**] [**/b:***date*] [**/a:***date*] [**/e:***time*] [**/l:***time*] [**/m**] [**/n**] [**/d**]

Parameters

drive1:

Specifies the drive on which the backed-up files are stored.

drive2:

Specifies the drive to which the backed-up files will be restored.

path

Specifies the directory to which the backed-up files will be restored. You must specify the same directory from which the files were backed up.

filename

Specifies the names of the backed-up files you want to restore.

Switches

/s Restores all subdirectories.

/p Prompts you for permission to restore files that are read-only (that have the read-only attribute set) or that have changed since the last backup (that have the archive attribute set).

/b:*date*

Restores only those files last modified on or before the specified date. The format of *date* varies according to the **country** setting in your CONFIG.SYS file. For information about specifying *date*, see the **date** command.

/a:*date*

Restores only those files last modified on or after the specified date. The format of *date* varies according to the **country** setting in your CONFIG.SYS file. For information about specifying *date*, see the **date** command.

/e:*time*

Restores only those files last modified at or earlier than the specified time. The format of *time* varies according to the **country** setting in your CONFIG.SYS file. For information about specifying *time*, see the **time** command.

/l:*time*

Restores only those files last modified at or later than the specified time. The format of *time* varies according to the **country** setting in your CONFIG.SYS file. For information about specifying *time*, see the **time** command.

/m Restores only those files modified since the last backup.

/n Restores only those files that no longer exist on the destination disk.

/d Displays a list of the files on the backup disk that match the names specified in *filename* without restoring any files. Even though no files are being restored, you must specify *drive2* when you use **/d**.

Notes

Checking restored files

Once a file has been restored, you can use the **dir** or **type** command to make sure the file was restored properly.

Limitations on restore

You cannot use the **restore** command to restore system files (IO.SYS and MSDOS.SYS). **Restore** does not work with drives that have been redirected with the **assign**, **join**, or **subst** command.

Compatibility with previous versions of backup

The MS-DOS version 5.0 **restore** command can restore files that were backed up by using any previous version of the MS-DOS **backup** command.

Restore exit codes

The following list shows each exit code and a brief description of its meaning:

0 **Restore** successfully restored the file or files.

1 **Restore** could not find the files to restore.

3 The user pressed CTRL+C to stop the restoring operation.

4 **Restore** stopped because of an error.

You can use the **errorlevel** parameter on the **if** command line in a batch program to process exit codes returned by **restore**. For an example of a batch program that processes exit codes, see the **backup** command. For more information about batch programs, see Chapter 10, "Working with Batch Programs."

Listing the names of backed-up files

Use the **/d** switch to see a list of the backed up files. If you specify *filename* with the **/d** switch, **restore** displays a list of the backed up files that match the name you specify. If you use the **/d** switch, **restore** does not restore any files.

Examples

To restore the file INVEST.MNT from the backup disk in drive A to the IRS directory on drive C, type the following command:

```
restore a: c:\irs\invest.mnt
```

MS-DOS prompts you to insert the backup disk into drive A. Once the backup disk is in drive A, press ENTER to continue.

Suppose you backed up all of the files in the directory \USER\ADAMS on drive C. To restore these files, insert the backup disk in drive A and type the following command:

```
restore a: c:\user\adams\*.*
```

It is important that you specify *.* for *filename*. Otherwise, the **restore** command attempts to restore a file named ADAMS in the USER directory.

To restore a complete hard disk from a backup disk (or disks) in drive A, type the following command:

```
restore a: c:\*.* /s
```

The **/s** switch and the wildcards (*.*) specify that **restore** is to restore all backed-up files to their original directories and subdirectories on drive C.

Related Command

For information about backing up files, see the **backup** command.

- ■ MS-DOS
- □ Batch
- □ CONFIG.SYS
- ■ Internal
- □ External
- ■ Network

Rmdir (rd)

Deletes (removes) a directory.

Before you can delete a directory, you must delete its files and subdirectories; the directory must be empty except for the "." and ".." symbols. For an introduction to the **rmdir** command, see Chapter 5, "Working with Directories."

Syntax

rmdir [*drive*:]*path*

rd [*drive*:]*path*

Parameter

[*drive*:]*path*

Specifies the location and name of the directory you want to delete.

Notes

Cannot delete directory with hidden or system files

You cannot delete a directory that contains files, including hidden or system files. If you attempt to do so, MS-DOS displays the following message:

```
Invalid path, not directory,
or directory not empty
```

Use the **dir** command to list hidden and system files and the **attrib** command to remove hidden and system attributes from files. For more information, see those commands.

Using the backslash character with the *path* parameter

If you insert a backslash (\) before the first directory name in *path*, MS-DOS treats the directory as a subdirectory of the root directory—regardless of your current directory. If you do not insert a backslash before the first directory name in *path*, MS-DOS treats the directory as a subdirectory of the current directory.

Deleting the current directory

You cannot use **rmdir** to delete the current directory. You must first change to a different directory (not a subdirectory of the current directory) and then use **rmdir** with a path. If you attempt to delete the current directory, MS-DOS displays a message in the following format:

```
Attempt to remove current directory - drive:path
```

MS-DOS also displays this message if you attempt to delete a directory that has been redirected by using the **subst** command.

Example

To delete a directory named \USER\SMITH, first ensure that the directory is empty, as in following example:

```
dir \user\smith /a
```

MS-DOS should display only the "." and ".." symbols.

Then, from any directory except \USER\SMITH, type the following command:

```
rmdir \user\smith
```

Related Commands

For information about creating a directory, see the **mkdir** command.

For information about hidden files, see the **attrib** command and the **dir** command (the **/a** switch).

- ■ MS-DOS
- ☐ Batch
- ☐ CONFIG.SYS
- ■ Internal
- ☐ External
- ■ Network

Set

Displays, sets, or removes MS-DOS environment variables.

You use environment variables to control the behavior of some batch files and programs and to control the way MS-DOS appears and works. The **set** command is often used in the AUTOEXEC.BAT file to set environment variables each time you start MS-DOS.

Syntax

set [*variable*=[*string*]]

To display the current environment settings, use the following syntax:

set

Parameters

variable

Specifies the variable you want to set or modify.

string

Specifies the string you want to associate with the specified variable.

Notes

Displaying the current environment settings

When you type the **set** command alone, MS-DOS displays the current environment settings. These settings usually include the COMSPEC and PATH environment variables that MS-DOS uses to help find programs on disk. PROMPT and DIRCMD are two other environment variables that MS-DOS uses. For more information about DIRCMD, see the **dir** command. For information about adding the **prompt** command to your AUTOEXEC.BAT file, see Chapter 11, "Customizing Your System."

Using parameters

When you use a **set** command and specify values for both *variable* and *string*, MS-DOS adds the specified *variable* value to the environment and associates *string* with that variable. If *variable* already exists in the environment, the new *string* value replaces the old *string* value.

If you specify only a variable and an equal sign (without a string) for the **set** command, MS-DOS clears the *string* value associated with the variable (as if the variable is not there at all).

Using set in batch files

When creating batch files, you can use the **set** command to create variables and use them in the same way as you would the numbered variables **%0** through **%9**. You can also use the variables **%0** through **%9** as input for the **set** command. For information about batch programs, see Chapter 10, "Working with Batch Programs."

Calling a set variable from a batch file

When you call a *variable* value from a batch file, you must enclose the value with percent signs (%). For example, if your batch program creates an environment variable named BAUD, you can use the string associated with BAUD as a replaceable parameter by inserting **%baud%** on the command line.

Effect of set on environment space

After you use a **set** command, MS-DOS might display the following message:

```
Out of environment space
```

This message means the available environment space is insufficient to hold the new variable definition. For information about how to increase the environment space, see the **command** command.

Examples

To set an environment variable named INCLUDE so that the string C:\INC (the INC directory on drive C) is associated with it, type the following command:

```
set include=c:\inc
```

You can then use the string C:\INC in batch files by enclosing the name INCLUDE with percent signs (%). For example, you might include the following command in a batch file in order to display the contents of the directory associated with the INCLUDE environment variable:

```
dir %include%
```

When MS-DOS processes this command, the string C:\INC replaces **%include%**.

Another possible use for the **set** command is in a batch program that adds a new directory to the PATH environment variable, as the following example shows:

```
@echo off
rem ADDPATH.BAT adds a new directory
rem to the PATH environment variable.
set path=%1;%path%
set
```

Related Commands

For information about setting environment variables that MS-DOS uses to control its own operations, see the **path**, **prompt**, **shell**, and **dir** commands.

- ■ MS-DOS
- ☐ Batch
- ☐ CONFIG.SYS
- ☐ Internal
- ■ External
- ■ Network

Setver

Sets the MS-DOS version number that MS-DOS version 5.0 reports to a program.

You can also use this command to display and modify the *version table*, which lists names of programs and the number of the MS-DOS version with which they are designed to run. If you are using a program that has not been updated for MS-DOS version 5.0, you can add its name to the version table by using the **setver** command.

Syntax

setver [*drive*:*path*] [*filename n.nn*]

setver [*drive*:*path*] [*filename* [**/delete** [**/quiet**]]

To display the current version table, use the following syntax:

setver [*drive*:*path*]

MS-DOS displays two columns: the left column lists the names of the program files; the right column lists the corresponding MS-DOS version with which each file is set to run.

Parameters

[drive:path]

Specifies the location of the SETVER.EXE file.

filename

Specifies the name of the program file (.EXE or .COM) that you want to add to the version table. You cannot use a wildcard (* or ?).

n.nn

Specifies the MS-DOS version (for example, 3.3 or 4.01) that MS-DOS version 5.0 reports to the specified program file.

Switches

/delete

Deletes the version-table entry for the specified program file. You can abbreviate this switch as **/d**.

/quiet

Hides the message typically displayed during deletion of an entry from the version table.

Notes

Loading the version table into memory

Before you can use the **setver** command, the version table must be loaded into memory by a device command in your CONFIG.SYS file. By default, the MS-DOS Setup program modifies your CONFIG.SYS file to ensure that the version table is loaded into memory each time you start your system. For information about loading the version table, see SETVER.EXE in Chapter 15, "Device Drivers."

Setting the version number of a command interpreter

You can use **setver** to set the version number of a command interpreter; however, if you set the version number of the MS-DOS version 5.0 command interpreter (COMMAND.COM), you may not be able to start your system.

Using the version table

Many programs designed to run with a previous version of MS-DOS will run correctly with MS-DOS version 5.0. In some cases, however, a program might not run correctly unless its name is included in the version table. The table indicates to the program that it is running with the MS-DOS version for which it was designed, even though it is running with MS-DOS version 5.0. By interpreting MS-DOS version 5.0 as the earlier version, the program will probably run correctly; however, using **setver** will not solve the problem if the program is not compatible with MS-DOS version 5.0.

Setver confirmation

If you make changes to the version table and no errors are detected, MS-DOS displays the following message:

```
WARNING - The application you are adding to the MS-DOS version
table may not have been verified by Microsoft on this version of
MS-DOS. Please contact your software vendor for information on
whether this application will operate properly under this version
of MS-DOS. If you execute this application by instructing MS-DOS to
report a different MS-DOS version number, you may lose or corrupt
data, or cause system instabilities. In that circumstance, Microsoft
is not responsible for any loss or damage.

Version table successfully updated
The version change will take effect the next time you restart your
system
```

Restarting after updating the version table

When you update the version table by adding or deleting entries, you must restart your system before the changes will take effect.

Updating existing entries

If you specify a filename that is already in the version table, the new entry replaces the existing entry.

Setver exit codes

The following list shows each exit code and a brief description of its meaning:

0 **Setver** successfully completed its task.

1 The user specified an invalid command switch.

2 The user specified an invalid filename.

3 There is insufficient system memory to carry out the command.

4 The user specified an invalid version-number format.

5 **Setver** could not find the specified entry in the version table.

6 **Setver** could not find the SETVER.EXE file.

7 The user specified an invalid drive.

8 The user specified too many command-line parameters.

9 **Setver** detected missing command-line parameters.

10 **Setver** detected an error while reading the SETVER.EXE file.

11 The SETVER.EXE file is corrupt.

12 The specified SETVER.EXE file does not support a version table.

13 There is insufficient space in the version table for a new entry.

14 **Setver** detected an error while writing to the SETVER.EXE file.

You can use the **errorlevel** parameter on the **if** command line in a batch program to process exit codes returned by **setver**. For an example of a batch program that processes exit codes, see the **backup** command. For more information about batch programs, see Chapter 10, "Working with Batch Programs."

Examples

Suppose you have a program file named MYPROG.EXE that runs with MS-DOS version 3.30. To run MYPROG.EXE, you must first use the **setver** command to create an entry in the version table that will cause MYPROG.EXE to interpret MS-DOS version 5.0 as version 3.30:

```
setver myprog.exe 3.30
```

To delete the MYPROG.EXE entry from the version table (without otherwise affecting the MYPROG.EXE file), type the following command:

```
setver myprog.exe /delete
```

To list the contents of the version table on drive C, type the following command:

```
setver c:
```

■ MS-DOS
☐ Batch
☐ CONFIG.SYS
☐ Internal
■ External
■ Network

Share

Starts the Share program, which installs file-sharing and locking capabilities on your hard disk.

Syntax

share [**/f:***space*] [**/l:***locks*]

In your CONFIG.SYS file, use the following syntax:

install=[[*drive***:**]*path*]**share.exe** [**/f:***space*] [**/l:***locks*]

Parameter

[*drive***:**]*path*

Specifies the location of the SHARE.EXE file.

Switches

/f:*space*

Allocates file space (in bytes) for the MS-DOS storage area used to record file-sharing information. The default value is 2048.

/l:*locks*

Sets the number of files that can be locked at one time. The default value is 20.

Notes

Common use of Share

Typically, you use Share in a network or multitasking environment in which programs share files. Share loads the code that supports file-sharing and locking in these environments. Once you install Share, MS-DOS uses the code loaded by Share to validate all read and write requests from programs.

Allocating space for file-sharing information

When deciding how many bytes to allocate for file sharing, note that each open file requires enough space for the length of the full path and filename. The average length of a filename and its path is 20 bytes.

Examples

The following example shows how you can use the **install** command in your CONFIG.SYS file to load Share, with the default values for the **/f** and **/l** switches. MS-DOS searches for the file SHARE.EXE in the DOS directory on drive C.

```
install=c:\dos\share.exe
```

The following example allocates 4096 bytes for storing file-sharing information and specifies that 25 files can be locked at one time. Again, MS-DOS searches for SHARE.EXE in the DOS directory on drive C:

```
install=c:\dos\share.exe /f:4096 /l:25
```

☐ MS-DOS
☐ Batch
■ CONFIG.SYS
☐ Internal
☐ External
☐ Network

Shell

Specifies the name and location of the command interpreter you want MS-DOS to use.

If you want to use your own command interpreter (instead of COMMAND.COM), you can specify its name by adding a **shell** command to your CONFIG.SYS file. For an introduction to using CONFIG.SYS, see Chapter 11, "Customizing Your System."

Syntax

shell=[[*drive*:]*path*]*filename* [*parameters*]

Parameters

[[*drive*:]*path*]*filename*

Specifies the location and name of the command interpreter you want MS-DOS to use.

parameters

Specifies any command-line parameters or switches that can be used with the specified command interpreter.

Notes

Default setting

The default command interpreter for MS-DOS is COMMAND.COM. If you do not use a **shell** command in your CONFIG.SYS file, MS-DOS searches for COMMAND.COM in the root directory of your startup drive. You need to use the **shell** command if you want to specify a COMMAND.COM file that is not in the root directory or if you do not want to use the default environment size for COMMAND.COM. For information about COMMAND.COM switches, see the **command** command.

Using switches with a command interpreter

The **shell** command itself does not accept any switches, but if the command interpreter does, you can include them on the **shell** command line.

Examples

Suppose the file NEWSHELL.COM is in a directory named BIN on your startup drive, and suppose you want to use NEWSHELL.COM as your command interpreter. To do this, add the following command to your CONFIG.SYS file:

```
shell=\bin\newshell.com
```

Suppose you add the line **shell=newcmdp.com** to your CONFIG.SYS file, and suppose the NEWCMDP.COM command interpreter accepts the switches **/c**, **/p**, and **/e**. You can now use any of these switches on the **shell** command line. Thus, the following command would be valid:

```
shell=newcmdp.com /c /p /e
```

The **shell** command is the preferred method of using **command** to increase the size of the environment. To increase the environment size to 512 bytes, add the following command to your CONFIG.SYS file:

```
shell=command.com /e:512 /p
```

To start an MS-DOS command interpreter located in the directory OLD on drive C, add the following command to your CONFIG.SYS file:

```
shell=c:\old\command.com c:\old /e:256 /p
```

Shift

- ☐ MS-DOS
- ■ Batch
- ☐ CONFIG.SYS
- ■ Internal
- ☐ External
- ☐ Network

Changes the position of replaceable parameters in a batch file.

For an introduction to using batch files, see Chapter 10, "Working with Batch Programs."

Syntax

shift

Notes

How the shift command works

The **shift** command changes the values of the replaceable parameters **%0** through **%9**, by copying each parameter into the previous one. In other words, the value of **%1** is copied to **%0**, the value of **%2** is copied to **%1**, and so on. This is useful for writing a batch file that performs the same operation on any number of parameters.

Working with more than 10 command-line parameters

You can also use the **shift** command to create a batch file that can accept more than 10 parameters. If you specify more than 10 parameters on the command line, those that appear after the tenth (**%9**) will be shifted one at a time into **%9**.

Shifting parameters back

There is no backward **shift** command. Once you carry out the **shift** command, you cannot recover the first parameter (**%0**) that existed before the shift.

Example

The following batch file, MYCOPY.BAT, shows how to use the **shift** command with any number of parameters. It copies a list of files to a specific directory. The parameters are the directory name followed by any number of filenames.

```
@echo off
rem MYCOPY.BAT copies any number of files
rem to a directory.
rem The command uses the following syntax:
rem mycopy dir file1 file2 ...
set todir=%1
:getfile
shift
if "%1"=="" goto end
copy %1 %todir%
goto getfile
:end
set todir=
echo All done
```

- ■ MS-DOS
- □ Batch
- □ CONFIG.SYS
- □ Internal
- ■ External
- ■ Network

Sort

Reads input, sorts data, and writes the results to the screen, to a file, or to another device.

Sort acts as a filter, reading characters in a specified column and rearranging them in ascending or descending order. For an introduction to the **sort** command, see Chapter 7, "Advanced Command Techniques."

Syntax

sort [**/r**] [**/+***n*] [<] [*drive1***:**][*path1*]*filename1* [> [*drive2***:**][*path2*]*filename2*]

[*command* ¦] **sort** [**/r**] [**/+***n*] [> [*drive2***:**][*path2*]*filename2*]

Parameters

[*drive1***:**][*path1*]*filename1*

Specifies the location and name of the file whose data you want to sort.

[drive2:][path2]filename2

Specifies the location and name of a file in which the sorted output is to be stored.

command

Specifies a command whose output is the data you want to sort.

Switches

/r Reverses the order of the sorting operation; that is, sorts from Z to A, and then from 9 to 0.

/+*n* Sorts the file according to the character in column *n*. If you do not use this switch, the **sort** command sorts data according to the characters in column 1.

Notes

Specifying a source

Unless you specify the *command* or *filename* parameter, **sort** acts as a filter and takes input from the MS-DOS standard input (usually from the keyboard, from a pipe, or from a file).

Using redirection symbols with sort

You can use the pipe (¦) or the less-than sign (<) to direct data through the **sort** command from *command* or *filename*. If you want to display the information one screen at a time or direct the information to a file, you can also specify the **more** command or a filename. You can use the greater-than sign (>) to direct the sorted output to a file.

Before using a pipe for redirection, you should set the TEMP environment variable in your AUTOEXEC.BAT file. For an introduction to using redirection symbols, see Chapter 7, "Advanced Command Techniques."

Collating sequence

The **sort** program uses the collating-sequence table corresponding to the country code and code-page settings. Characters greater than ASCII code 127 are sorted based on information in the COUNTRY.SYS file or in an alternate file specified by the **country** command in your CONFIG.SYS file.

Uppercase vs. lowercase

Sort does not distinguish between uppercase and lowercase letters.

Limits on file size

The **sort** command can handle files as large as 64K.

Examples

The following command reads the file EXPENSES.TXT, sorts it in reverse order, and displays it on your screen:

```
sort /r < expenses.txt
```

Suppose you want to search a large file named MAILLST.TXT for the text "Jones", and suppose you want to sort the results of the search. To do this, use the pipe (¦) to direct the output of a **find** command to the **sort** command, as shown in the following example:

```
find "Jones" maillst.txt ¦ sort
```

The command produces a sorted list of lines that contain the specified text.

To sort keyboard input and display the results alphabetically on the screen, you can first use the **sort** command with no parameters, as the following example shows:

```
sort
```

Then type the text you want sorted, pressing ENTER at the end of each line. When you have finished typing text, press CTRL+Z, and then press ENTER. The **sort** command displays the text you typed, sorted alphabetically. You could also redirect sorted keyboard input to a file.

Related Command

For information about displaying information one screen at a time, see the **more** command.

☐ MS-DOS
☐ Batch
■ CONFIG.SYS

☐ Internal
☐ External

☐ Network

Stacks

Supports the dynamic use of data stacks to handle hardware interrupts.

For an introduction to using the CONFIG.SYS file, see Chapter 11, "Customizing Your System."

Syntax

stacks=*n*,*s*

Parameters

n Specifies the number of stacks. Valid values for *n* are 0 and numbers in the range 8 through 64.

s Specifies the size (in bytes) of each stack. Valid values for *s* are 0 and numbers in the range 32 through 512.

Notes

Default settings

The default settings for the **stack** command are as follows:

Computer	Stacks
IBM PC, IBM PC/XT, IBM PC-Portable	0,0
Other	9,128

Special cases for stack allocation

Upon receiving a hardware interrupt, MS-DOS allocates one stack from the specified number of stacks. When you specify 0 for the *n* and *s* values, MS-DOS allocates no stacks. If the values are 0, each running program must have enough stack space to accommodate the computer's hardware interrupt drivers. Many computers operate correctly, saving some memory for programs, with *n* and *s* values of 0. If, however, your computer becomes unstable when you set these values to 0, return to the default values.

Example

To allocate 8 stacks of 512 bytes each for hardware-interrupt handling, add the following command to your CONFIG.SYS file:

```
stacks=8,512
```

- ■ MS-DOS
- ☐ Batch
- ☐ CONFIG.SYS
- ☐ Internal
- ■ External
- ☐ Network

Subst

Associates a path with a drive letter.

The drive letter you assign represents a virtual drive, because you can use the drive letter in commands as if it represented a physical drive. For an introduction to the **subst** command, see Chapter 6, "Managing Disks."

Syntax

subst [*drive1*: [*drive2*:]*path*]

subst *drive1*: **/d**

To display the names of the virtual drives in effect, use the following syntax:

subst

Parameters

drive1:

Specifies the virtual drive to which you want to assign a path.

drive2:

Specifies the physical drive that contains the specified path (if different from the current drive).

path

Specifies the path that you want to assign to a virtual drive.

Switch

/d Deletes a virtual drive.

Notes

Using other commands with subst

The following commands do not work, or should not be used, on drives used in the **subst** command:

assign	**diskcopy**	**mirror**
backup	**fdisk**	**recover**
chkdsk	**format**	**restore**
diskcomp	**label**	**sys**

Valid *drive1* values

The *drive1* parameter must be within the range specified by the **lastdrive** command. If not, **subst** displays the following error message:

```
Invalid parameter - drive1:
```

Ensuring compatibility with future versions of MS-DOS

To ensure compatibility with future versions of MS-DOS, you should use **subst** rather than the **assign** command.

Examples

The following command creates a virtual drive Z for the path B:\USER\BETTY\FORMS:

```
subst z: b:\user\betty\forms
```

Now, instead of typing the full path, you can reach this directory by typing the letter of the virtual drive, followed by a colon, as in the following example:

```
z:
```

This example works only if you have included the line **lastdrive=z** in your CONFIG.SYS file to define Z as the highest letter that MS-DOS recognizes as a disk drive.

Related Commands

For information about joining a disk drive to a directory, see the **join** command.

For information about increasing the number of available drive letters, see the **lastdrive** command.

☐ MS-DOS
☐ Batch
■ CONFIG.SYS
☐ Internal
☐ External
☐ Network

Switches

Forces an enhanced keyboard to behave like a conventional keyboard.

You use this command in your CONFIG.SYS file. For an introduction to using the CONFIG.SYS file, see Chapter 11, "Customizing Your System."

Syntax

switches=/k

Notes

When to use the switches command

If you have a program that does not correctly interpret input from an enhanced keyboard, add this command to your CONFIG.SYS file so your enhanced keyboard will use conventional keyboard functions.

Using the /k switch with ANSI.SYS

If you use the **switches=/k** command and you install the ANSI.SYS device driver, use the **/k** switch on the **device** command line for ANSI.SYS.

Example

If you want MS-DOS to use conventional keyboard functions even though you are using an enhanced keyboard, add the following command to your CONFIG.SYS file:

```
switches=/k
```

- ■ MS-DOS
- ☐ Batch
- ☐ CONFIG.SYS
- ☐ Internal
- ■ External
- ☐ Network

Sys

Copies MS-DOS system files and the MS-DOS command interpreter (COMMAND.COM) to the disk in a drive you specify.

The two system files (IO.SYS and MSDOS.SYS) are hidden files and do not typically appear when you type the **dir** command. For an introduction to the **sys** command, see Chapter 6, "Managing Disks."

Syntax

sys [*drive1*:][*path*] *drive2*:

Parameters

[*drive1*:][*path*]

Specifies the location of the system files. If you do not specify a path, MS-DOS searches the root directory on the current drive for the system files.

drive2:

Specifies the drive to which you want to copy the system files. These files can be copied only to a root directory, not to a subdirectory.

Notes

How the sys command copies files

The **sys** command copies the files in the following order: IO.SYS, MSDOS.SYS, and COMMAND.COM.

No requirement for contiguous files

MS-DOS no longer requires the two system files to be contiguous. This means that when you want to copy a new version of MS-DOS to a disk containing system files for MS-DOS version 3.3 or earlier, you need not reformat the disk.

Using the sys command on assigned drives and networks

The **sys** command does not work on drives that have been redirected by using the **assign**, **join**, or **subst** command. **Sys** also does not work on network drives.

Examples

To copy the MS-DOS system files and command interpreter from the disk in the current drive to a disk in drive A, type the following command:

```
sys a:
```

To copy the MS-DOS system files and command interpreter from the disk in drive D to a disk in drive A, type the following the command:

```
sys d:\ a:
```

Related Commands

For information about copying files, see the **copy** and **xcopy** commands.

- ■ MS-DOS
- ☐ Batch
- ☐ CONFIG.SYS
- ■ Internal
- ☐ External
- ■ Network

Time

Displays the system time or sets your computer's internal clock.

MS-DOS uses time information to update the directory whenever you create or change a file. For an introduction to the **time** command, see Chapter 2, "Command-Line Basics."

Syntax

time [*hours*:[*minutes*[:*seconds*[.*hundredths*]]][**a**|**p**]]

To display the current time or to display a prompt by which you can change the current time, use the following syntax:

time

Parameters

hours

Specifies the hour. Valid values are in the range 0 through 23.

minutes

Specifies minutes. Valid values are in the range 0 through 59.

seconds

Specifies seconds. Valid values are in the range 0 through 59.

hundredths

Specifies hundredths of a second. Valid values are in the range 0 through 99.

a|p Specifies A.M or P.M. for the 12-hour time format. If you type a valid 12-hour time but do not type **a** or **p**, **time** uses **a** (for A.M.).

Notes

Specifying an invalid time format

If you specify the time in an invalid format, MS-DOS displays the following message and then waits for you to specify the time:

```
Invalid time
Enter new time:_
```

Changing the time format

You can change the **time** format by changing the **country** setting in your CONFIG.SYS file. For more information, see the **country** command. Depending on the country code, MS-DOS will display the time in the 12-hour format or the 24-hour format. If you are setting the time in the 12-hour format, be sure to specify **p** for hours after noon.

Ensuring that MS-DOS prompts you for the time

If you want MS-DOS to prompt you for the current time whenever you start your system, you can add the **time** command to your AUTOEXEC.BAT file. MS-DOS will automatically prompt you for the time and date if you do not have an AUTOEXEC.BAT file. For more information, see Chapter 11, "Customizing Your System."

Examples

To set your computer's clock to 1:36 P.M., use either of the following commands:

```
time 13:36

time 1:36p
```

Related Command

For information about changing the current date, see the **date** command.

For information about changing the time format, see the **country** command.

■ MS-DOS
☐ Batch
☐ CONFIG.SYS
☐ Internal
■ External
■ Network

Tree

Graphically displays the directory structure of a path or of the disk in a drive.

For an introduction to the **tree** command, see Chapter 5, "Working with Directories."

Syntax

tree [*drive*:][*path*] [**/f**] [**/a**]

Parameters

drive:

Specifies the drive that contains the disk for which you want to display the directory structure.

path

Specifies the directory for which you want to display the directory structure.

Switches

/f Displays the names of the files in each directory.

/a Specifies that **tree** is to use text characters instead of graphic characters to show the lines linking subdirectories. Use this switch with code pages that do not support graphic characters and to send output to printers that don't properly interpret the graphic characters.

Note

The structure that **tree** displays depends upon the parameters you specify on the command line. If you do not specify a drive or path, **tree** displays the tree structure beginning with the current directory of the current drive.

Examples

To display the names of all the subdirectories on the disk in your current drive, type the following command:

```
tree \
```

To display, one screen at a time, the files in all the directories on drive C, type the following command:

```
tree c:\ /f ¦ more
```

To print the same list that the previous example displayed, type the following command:

```
tree c:\ /f > prn
```

For information about pipes and command redirection, see Chapter 7, "Advanced Command Techniques."

Related Command

For information about displaying the contents of a directory, see the **dir** command.

- ■ MS-DOS
- ☐ Batch
- ☐ CONFIG.SYS
- ■ Internal
- ☐ External
- ■ Network

Type

Displays the contents of a text file.

Use the **type** command to view a text file without modifying it. For an introduction to the **type** command, see Chapter 4, "Working with Files."

Syntax

type [*drive*:][*path*]*filename*

Parameter

[*drive*:][*path*]*filename*

Specifies the location and name of the file that you want to view.

Notes

Displaying binary files

If you display a binary file or a file created by a program, you may see strange characters on the screen, including formfeed characters and escape-sequence symbols. These characters represent control codes used in the binary file. In general, you should avoid using the **type** command to display binary files.

Changing the contents of a file

For information about using MS-DOS to change the contents of a file, see Chapter 9, "Working with MS-DOS Editor."

Examples

If you want to display the contents of a file named HOLIDAY.MAR, type the following command:

```
type holiday.mar
```

If the file you want to display is long, you can use the **more** command along with **type**, as shown in the following command, to display the file's contents one screen at a time:

```
type holiday.mar ¦ more
```

Before using a pipe (¦) for redirection, you should set the TEMP environment variable in your AUTOEXEC.BAT file.

For information about using pipes and command redirection, see Chapter 7, "Advanced Command Techniques."

Related Commands

For information about displaying filenames and file sizes, see the **dir** command.

For information about displaying text files one screen at a time, see the **more** command.

■ MS-DOS
☐ Batch
☐ CONFIG.SYS
☐ Internal
■ External
☐ Network

Undelete

Restores files previously deleted with the **del** command.

For an introduction to the **undelete** command, see Chapter 4, "Working with Files."

Syntax

undelete [[*drive*:][*path*]*filename*] [**/list**|**/all**] [**/dos**|**/dt**]

Parameter

[*drive*:][*path*]*filename*

Specifies the location and name of the file or set of files you want to recover. By default, **undelete** restores all deleted files in the current directory.

Switches

/list

Lists the deleted files that are available to be recovered, but does not recover any files. The [*drive*:][*path*]*filename* parameter and the **/dt** and **/dos** switches control the listing produced by this switch.

/all

Recovers deleted files without prompting for confirmation on each file. **Undelete** uses the deletion-tracking file if it is present. Otherwise, **undelete** recovers files from the DOS directory, supplying a number sign (#) for the missing first character in the filename. If a duplicate filename already exists, this switch next tries each of the following characters, in the order listed, until the result is a unique filename: #%&-0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ.

/dos

Recovers only those files that are internally listed as deleted by MS-DOS, prompting for confirmation on each file. If a deletion-tracking file exists, this switch causes **undelete** to ignore it.

/dt

Recovers only those files listed in the deletion-tracking file produced by the **mirror** command, prompting for confirmation on each file. For more information about deletion-tracking files, see the **mirror** command and Chapter 4, "Working with Files."

Note

Use any one of the following switches: **/dos**, **/dt**, or **/all**. If you do not specify a switch, **undelete** uses the deletion-tracking file, if it is available. If the deletion-tracking file is not available, **undelete** attempts to recover files by using the directory listing of deleted files. Recovering deleted files by using the deletion-tracking file is usually more reliable than doing so by using the directory listing of deleted files.

CAUTION **Undelete** cannot restore a directory that has been removed, and it cannot retrieve a file if you have removed the directory that contained the file. If the directory was an immediate subdirectory of the root directory, you may be able to retrieve the directory and its files if you first use the **unformat** command described later in this chapter to restore the directory and then use **undelete** to retrieve the files. You must use caution because you can lose data if you use **unformat** incorrectly. Usually **unformat** can restore only immediate subdirectories of the root directory. However, when you use **unformat** to recover an accidentally formatted disk, **unformat** recovers all root-level files and subdirectory names.

Examples

The following command specifies that **undelete** is to recover all deleted files in the current directory one at a time and to prompt for confirmation on each file:

```
undelete
```

The following command specifies that **undelete** is to recover all deleted files with the .BAT extension in the root directory of drive C, without prompting for confirmation on each file:

```
undelete c:\*.bat /all
```

CAUTION Once you delete a file from your disk, you may not be able retrieve it. Although the **undelete** command can recover deleted files, it can do so with certainty only if no other files have been created or changed on the disk. If you accidentally delete a file that you want to keep, stop what you are doing and immediately use the **undelete** command to retrieve the file.

■ MS-DOS
☐ Batch
☐ CONFIG.SYS
☐ Internal
■ External
☐ Network

Unformat

Restores a disk erased by the **format** command or restructured by the **recover** command.

Unformat restores only local hard disk drives and floppy disk drives; it cannot be used on network drives. The **unformat** command can also rebuild a corrupted disk partition table on a hard disk drive. For an introduction to the **unformat** command, see Chapter 6, "Managing Disks."

Syntax

unformat *drive*: [**/j**]

unformat *drive*: [**/u**] [**/l**] [**/test**] [**/p**]

unformat [**/partn**] [**/l**]

Parameter

drive:

Specifies the drive that contains the disk on which you want to recover files.

Switches

/j Verifies that the file created by the **mirror** command has been saved and that it agrees with the system information on the disk. Like the **/test** switch, this parameter does not actually rebuild the disk. When the **/j** switch is used, it must be the only switch on the command line.

/u Unformats a disk without using the mirror file.

/l When used without the **/partn** switch, lists every file and subdirectory found by **unformat**. If you do not specify this switch, **unformat** lists only subdirectories and files that are fragmented. To suspend scrolling of the displayed list, press CTRL+S; to resume scrolling, press any key. Use the **/l** switch only if you do not want **unformat** to use the file created by the Mirror program.

When the **/l** switch is used with the **/partn** switch, **unformat** displays the partition table of the current drive. The size of the partition table is represented in megabytes and is based upon standard-sized sectors of 512 bytes. If the sectors on your hard disk are any other size, the displayed size in bytes may be incorrect. The total is the result of multiplying the number of sectors by 512.

/test

Shows how **unformat** would recreate the information on the disk, but does not actually unformat the disk. Use this switch only if you do not want **unformat** to use the file created by the Mirror program.

/p Sends output messages to the printer connected to LPT1.

/partn

Restores a corrupted partition table of a hard disk drive. This switch requires a PARTNSAV.FIL file that was created by using the **mirror** command with the **/partn** switch.

Notes

Limitation on the unformat command

If the **format** command was used with the **/u** switch, **unformat** cannot restore the disk to its previous condition.

Unformatting a disk by using a mirror file

When a file created by the Mirror program is available, **unformat** uses it to help restore the condition of the disk that existed before the disk was formatted or erased.

The Mirror program creates a file containing information about a disk's file allocation table and root directory. When this file is available, it provides the most reliable method for restoring a disk that was unintentionally formatted or erased. To enable **unformat** to recover files in the root directory that have been accidentally deleted, you should use the **mirror** command frequently to record the current state of your system. If your system contains a hard disk, it is highly recommended that you put the **mirror** command in your AUTOEXEC.BAT file to record the state of each hard disk in your system.

The effect of using the **format** command without the **/u** switch is the same as the effect of using the **mirror** command.

CAUTION **Unformat** attempts to restore the root directory of a disk to its condition when the mirror file was created. As a result, **unformat** cannot recover files in the root directory that were created or changed after the mirror file was created. Use the **unformat** command to recover files deleted from the root directory by the **del** command only if the **undelete** command was unable to restore them.

By default, **unformat** uses the mirror file if it is available. If you specify the **/l** or **/test** switch, **unformat** does *not* use the mirror file.

You can use the optional **/j** switch to verify that the mirror file exists and that it agrees with information on the formatted disk. With this switch, **unformat** does not actually rebuild the disk.

When you rebuild a disk by using the mirror file, **unformat** displays the time and date of the most recent (last) mirror file on the disk and the time and date of the previous (prior) mirror file on the disk. Press L to update the system area of your disk with the latest information; or press P to update your disk with information from the prior mirror file.

The only case in which you would want to use the prior mirror file is the following: you use the **mirror** command, then the disk is corrupted, then you use the **format** command. If you use the **mirror** command and the **format** command after the disk is corrupted, the **unformat** command will not work. **Unformat** searches the disk for the mirror file. Because **unformat** searches the disk directly, the disk does not have to be "readable" by MS-DOS for **unformat** to work. Do not use the **fdisk** command before using **unformat**; doing so can destroy information not saved by the Mirror program.

Unformatting a disk without a mirror file

If you do not have a mirror file or if your mirror file is very old, the **unformat** command can restore your disk by using information in the root directory and file allocation table on the disk. This method is slower and less reliable than restoring the disk by using the information in the mirror file, however, so you should use this method only if you cannot use a mirror file.

If you specify the **/l** switch, **unformat** attempts to restore your disk by using information in the root directory and file allocation table.

As **unformat** rebuilds the disk, it displays how many subdirectories it has found; and if you specified the **/l** switch, it shows you all files in each subdirectory.

If **unformat** finds a file that appears to be fragmented (that is, stored in separate places on the disk), it cannot recover the file because it cannot locate the remaining portions of the file. In this case, the **unformat** command prompts you to confirm whether you want **unformat** to truncate the file (that is, recover only the first part of the file that it can locate) or delete the file altogether.

If **unformat** does not prompt you for a specific file, that file is most likely intact. In certain circumstances, however, **unformat** may not recognize that a file is fragmented, even though it has located only a portion of the file. If this happens to a program file, the program does not run properly. If this happens to a data file, information is lost and the program that created the data file may not be able to read it. In these cases, your only recourse is to restore the files from your original floppy disks or backup files. You can reduce how often **unformat** fails to recognize fragmented files by using the Mirror program on a regular basis.

Restoring disk partition information

In addition to restoring a disk after accidental formatting or erasure, you can also use the **unformat** command to rebuild the disk partition table of a hard disk when the table has become corrupted. Corruption of the disk partition table might be the problem when MS-DOS displays the following message during an attempt by you or by a program to access a hard disk:

```
Invalid drive specification
```

This message indicates that MS-DOS cannot find the "logical" disk (which is defined by the partition table) on the physical hard disk.

To be able to rebuild the disk partition table, you must have saved this information on a separate disk by using the **mirror** command with the **/partn** switch. The Mirror program creates a file named PARTNSAV.FIL.

To rebuild the disk partition table, type the following command:

```
unformat /partn
```

Unformat prompts you to insert the floppy disk containing PARTNSAV.FIL and prompts you for the letter of the floppy disk drive. Insert the disk in the drive and type its letter at the prompt.

Next, **unformat** checks the drive parameters that were saved in PARTNSAV.FIL against the actual drive parameters. If they do not match exactly, **unformat** does not restore the information.

When you use **unformat** with the **/partn** switch, **unformat** prompts you to insert a system disk in drive A and press ENTER to restart. This is necessary to inform MS-DOS that the partition-table information has changed. You should then use **unformat** without the **/partn** switch to recover your directories and file allocation table.

Examples

To determine whether **unformat** can restore a formatted disk in drive C by using a mirror file, type the following command:

```
unformat c: /j
```

To restore a formatted disk in drive A by using a mirror file (if available), type the following command:

```
unformat a:
```

To determine whether **unformat** can restore a formatted disk in drive A without a mirror file, type the following command:

```
unformat a: /test
```

To restore a formatted disk in drive A without a mirror file, listing all files and subdirectories, type the following command:

```
unformat a: /l
```

Related Commands

For information about formatting a disk, see the **format** command.

For information about saving system information that can be used by **unformat** to restore a disk, see the **mirror** command.

■ MS-DOS
☐ Batch
☐ CONFIG.SYS

■ Internal
☐ External

■ Network

Ver

Displays the MS-DOS version number.

Syntax

ver

Example

When you enter the **ver** command, MS-DOS displays the following message:

```
MS-DOS Version 5.0
```

- ■ MS-DOS
- ☐ Batch
- ☐ CONFIG.SYS
- ■ Internal
- ☐ External
- ■ Network

Verify

Tells MS-DOS whether to verify that your files are written correctly to a disk.

You can use this command to make sure data is not written to a bad sector, for example. For an introduction to the **verify** command, see Chapter 6, "Managing Disks."

Syntax

verify [**on**|**off**]

Switch

on|**off**

Specifies whether MS-DOS is to verify that write operations are done correctly. The **on** value enables this verification process. The **off** value disables it.

Notes

Displaying the current status of the on|off switch

Use the **verify** command without a switch to find out whether verification is turned on.

How verify affects performance

Turning **verify** on slows down all disk write operations.

Verifying files as you copy them

To verify that files are copied correctly, you can also use the **/v** switch with the **copy** or **xcopy** command. For more information about the **/v** switch, see these commands.

Related Command

For information about checking a disk for bad sectors, see the **chkdsk** command.

■ MS-DOS
☐ Batch
☐ CONFIG.SYS
■ Internal
☐ External
■ Network

Vol

Displays the disk volume label and serial number, if they exist.

A serial number is displayed for a disk formatted with MS-DOS version 4.0 or later. For an introduction to the **vol** command, see Chapter 6, "Managing Disks."

Syntax

vol [*drive*:]

Parameter

drive:

Specifies the drive that contains the disk for which you want to display the volume label and serial number.

Note

To cause MS-DOS to display the volume label of the disk in the current drive, you can use the **vol** command with no parameter.

Related Commands

For information about assigning a volume label, see the **format** and **label** commands.

■ MS-DOS
☐ Batch
☐ CONFIG.SYS
☐ Internal
■ External
■ Network

Xcopy

Copies files (except hidden and system files) and directories, including subdirectories.

With this command, you can copy all the files in a directory, including the files in the subdirectories of that directory. For an introduction to the **xcopy** command, see Chapter 5, "Working with Directories."

Syntax

xcopy *source* [*destination*] [**/a**|**/m**] [**/d:***date*] [**/p**] [**/s** [**/e**]] [**/v**] [**/w**]

Parameters

source

Specifies the location and names of the files you want to copy. *Source* must include either a drive or a path.

destination

Specifies the destination of the files you want to copy. *Destination* can include a drive letter and colon, a directory name, a filename, or a combination.

Switches

/a Copies only source files that have their archive file attributes set. This switch does not modify the archive file attribute of the source file. For information about how to set the archive file attribute, see the **attrib** command.

/m Copies source files that have their archive file attributes set. Unlike the **/a** switch, the **/m** switch turns off archive file attributes in the files specified in *source*. For information about how to set the archive file attribute, see the **attrib** command.

/d:*date*

Copies only source files modified on or after the specified date. Note that the format of *date* depends on the **country** setting you are using. For information about changing the date format, see Chapter 13, "Customizing for International Use."

/p Prompts you to confirm whether you want to create each destination file.

/s Copies directories and subdirectories, unless they are empty. If you omit this switch, **xcopy** works within a single directory.

/e Copies any subdirectories, even if they are empty. You must use the **/s** switch with this switch.

/v Verifies each file as it is written to the destination file to make sure that the destination files are identical to the source files.

/w Displays the following message and waits for your response before starting to copy files.

```
Press any key to begin copying file(s)
```

Notes

Default value for *destination*

If you omit *destination*, the **xcopy** command copies the files to the current directory.

Specifying whether *destination* is a file or directory

If *destination* does not contain an existing directory and does not end with a backslash (\), **xcopy** prompts you with a message in the following format:

```
Does destination specify a file name
or directory name on the target
(F = file, D = directory)?
```

Press F if you want the file(s) to be copied to a file. Press D if you want the file(s) to be copied to a directory.

Xcopy does not copy hidden and system files

In previous versions of MS-DOS, **xcopy** copies hidden and system files. This is no longer the case in MS-DOS version 5.0. To remove the hidden or system attribute from a file, use the **attrib** command.

Xcopy sets archive attribute for destination files

Xcopy creates files with the archive attribute set, whether or not this attribute was set in the source file. For more information about file attributes, see the **attrib** command.

Xcopy vs. diskcopy

If you have a disk that contains files in subdirectories and you want to copy it to a disk that has a different format, you should use the **xcopy** command instead of **diskcopy**. Since the **diskcopy** command copies disks track by track, it requires that your source and destination disks have the same format. **Xcopy** has no such requirement. In general, use **xcopy** unless you need a complete disk image copy. However, **xcopy** will not copy hidden or system files such as IO.SYS and MSDOS.SYS. Therefore, use **diskcopy** to make copies of system disks.

Xcopy exit codes

The following list shows each exit code and a brief description of its meaning:

0 Files were copied without error.

1 No files were found to copy.

2 The user pressed CTRL+C to terminate **xcopy**.

4 Initialization error occurred. There is not enough memory or disk space, or you entered an invalid drive name or invalid syntax on the command line.

5 Disk write error occurred.

You can use the **errorlevel** parameter on the **if** command line in a batch program to process exit codes returned by **xcopy**. See the following "Examples" section.

Examples

The following example copies all the files and subdirectories (including any empty subdirectories) from the disk in drive A to the disk in drive B:

```
xcopy a: b: /s /e
```

The following example uses the **/d:** and **/v** switches:

```
xcopy a: b: /d:04/11/90 /v
```

In this example, only files on the disk in drive A that were written on or after 04/11/90 are copied to the disk in drive B. Once the files are written to the disk in drive B, the **xcopy** command compares the files on the two disks to make sure they are the same.

You can create a batch program to perform **xcopy** operations and use the batch **if** command to process the exit code in case an error occurs. For example, the following batch program uses replaceable parameters for the **xcopy** *source* and *destination* parameters:

```
@echo off
rem COPYIT.BAT transfers all source
rem files in all directories on the source
rem drive (%1) to the destination drive (%2)

xcopy %1 %2 /s /e

if errorlevel 4 goto lowmemory
if errorlevel 2 goto abort
if errorlevel 0 goto exit

:lowmemory
echo Insufficient memory to copy files or
echo invalid drive or command-line syntax.
goto exit

:abort
echo You pressed CTRL+C to end the copy operation.
goto exit

:exit
```

To use this batch program to copy all files in the C:\PRGMCODE directory and its subdirectories to drive B, type the following command:

```
copyit c:\prgmcode b:
```

The command interpreter substitutes C:\PRGMCODE for **%1** and B: for **%2**, then uses **xcopy** with the **/e** and **/s** switches. If **xcopy** encounters an error, the batch program reads the exit code and goes to the label indicated in the appropriate **if errorlevel** statement. MS-DOS displays the appropriate message and exits from the batch program.

For an introduction to using batch programs, see Chapter 10, "Working with Batch Programs."

Related Command

For information about copying individual files, see the **copy** command.

Chapter 15

Device Drivers

15

This chapter describes the device drivers supplied with MS-DOS version 5.0. For an introduction to installing and using device drivers, see Chapter 11, "Customizing Your System," and Chapter 12, "Optimizing Your System."

The following list shows the device drivers described in this chapter and a brief description of each:

ANSI.SYS

Defines functions that change display graphics, control cursor movement, and reassign keys.

DISPLAY.SYS

Supports code-page switching for the console.

DRIVER.SYS

Creates a logical drive that you can use to refer to a physical disk drive, and specifies parameters for a drive not supported by your hardware.

EGA.SYS

Saves and restores the display when Task Swapper is used with an EGA monitor.

EMM386.EXE

Simulates expanded memory in extended memory and provides access to the upper memory area on a computer with an 80386 or higher processor.

HIMEM.SYS

Manages the use of extended memory on a computer with an 80286 or higher processor and extended memory.

PRINTER.SYS

Supports code-page switching for printers.

RAMDRIVE.SYS

Creates a virtual disk drive in your system's random access memory (RAM) to simulate a hard disk drive.

SETVER.EXE

Loads the MS-DOS version table into memory.

SMARTDRV.SYS

Creates a disk cache in extended or expanded memory.

ANSI.SYS

Defines functions that change display graphics, control cursor movement, and reassign keys. The ANSI.SYS device driver supports the use of ANSI escape sequences to control your system's screen and keyboard. An ANSI escape sequence is a sequence of ASCII characters, the first two of which are the escape character (1Bh) and the left-bracket character (5Bh). The character or characters following the escape and left-bracket characters specify an alphanumeric code that controls a keyboard or display function. Case is significant for all characters you use in ANSI escape sequences. For an introduction to using escape sequences, see Chapter 11, "Customizing Your System."

Syntax

device=[*drive*:][*path*]**ansi.sys** [**/x**] [**/k**]

Parameter

[*drive*:][*path*]

Specifies the location of the ANSI.SYS file.

Switches

/x Remaps extended keys independently on 101-key keyboards.

/k Ignores extended keys on 101-key keyboards.

Notes

Remapping extended keys

If you have a keyboard with 101 keys, you may want to use the **/x** switch to remap certain extended keys. For example, there are two HOME keys on keyboards with 101 keys. One HOME key is on the numeric keypad and the other is in the block of cursor-control keys. To MS-DOS, the two HOME keys are the same, unless you specify the **/x** switch.

Ignoring extended keys

Some computers do not reliably detect all the extended-keyboard services of 101-key keyboards. You can use the **/k** switch to force ANSI.SYS to ignore extended keys.

Parameters used in ANSI escape sequences

Pn Numeric parameter. Specifies a decimal number.

Ps Selective parameter. Specifies a decimal number that you use to select a function. You can specify more than one function by separating the parameters with semicolons.

PL Line parameter. Specifies a decimal number that represents one of the lines on your display or on another device.

Pc Column parameter. Specifies a decimal number that represents one of the columns on your screen or on another device.

ANSI escape sequences for cursor movement, graphics, and keyboard settings

In the following list of ANSI escape sequences, the abbreviation **ESC** represents the ASCII escape character 27 (1Bh), which appears at the beginning of each escape sequence.

ESC[*PL*;*Pc*H

Cursor Position Moves the cursor to the specified position (coordinates). If you do not specify a position, the cursor moves to the home position—the upper-left corner of the screen (line 0, column 0). This escape sequence works the same way as the following Cursor Position escape sequence.

ESC[*PL*;*Pc*f

Cursor Position Works the same way as the preceding Cursor Position escape sequence.

ESC[*Pn*A

Cursor Up Moves the cursor up by the specified number of lines without changing columns. If the cursor is already on the top line, ANSI.SYS ignores this sequence.

ESC[*Pn*B

Cursor Down Moves the cursor down by the specified number of lines without changing columns. If the cursor is already on the bottom line, ANSI.SYS ignores this sequence.

ESC[*Pn*C

Cursor Forward Moves the cursor forward by the specified number of columns without changing lines. If the cursor is already in the rightmost column, ANSI.SYS ignores this sequence.

ESC[*Pn*D

Cursor Backward Moves the cursor back by the specified number of columns without changing lines. If the cursor is already in the leftmost column, ANSI.SYS ignores this sequence.

ESC[s

Save Cursor Position Saves the current cursor position. You can move the cursor to the saved cursor position by using the Restore Cursor Position sequence.

ESC[u

Restore Cursor Position Returns the cursor to the position stored by the Save Cursor Position sequence.

ESC[2J

Erase Display Clears the screen and moves the cursor to the home position (line 0, column 0).

ESC[K

Erase Line Clears all characters from the cursor position to the end of the line (including the character at the cursor position).

ESC[*Ps*;...;*Ps*m

Set Graphics Mode Calls the graphics functions specified by the following values. These specified functions remain active until the next occurrence of this escape sequence.

Text attributes

0	All attributes off
1	Bold on
4	Underscore (on monochrome display adapter only)
5	Blink on
7	Reverse video on
8	Concealed on

Foreground colors

30	Black
31	Red
32	Green
33	Yellow
34	Blue
35	Magenta
36	Cyan
37	White

Background colors

40	Black
41	Red
42	Green
43	Yellow
44	Blue
45	Magenta
46	Cyan
47	White

Parameters 30 through 47 meet the ISO 6429 standard.

ESC[=*Ps*h

Set Mode Changes the screen width or type to the mode specified by one of the following values:

0	40 × 25 monochrome (text)
1	40 × 25 color (text)
2	80 × 25 monochrome (text)
3	80 × 25 color (text)
4	320 × 200 4-color (graphics)
5	320 × 200 monochrome (graphics)
6	640 × 200 monochrome (graphics)
7	Enables line wrapping
13	320 × 200 color (graphics)
14	640 × 200 color (16-color graphics)
15	640 × 350 monochrome (2-color graphics)
16	640 × 350 color (16-color graphics)
17	640 × 480 monochrome (2-color graphics)
18	640 × 480 color (16-color graphics)
19	320 × 200 color (256-color graphics)

ESC[=*Ps*l

Reset Mode Resets the mode by using the same values that Set Mode uses, except for 7, which disables line wrapping. The last character in this escape sequence is a lowercase L.

ESC[*code*;*string*;...p

Set Keyboard Strings Redefines a keyboard key to a specified string. The parameters for this escape sequence are defined as follows:

- *Code* is one or more of the values listed in the following table. These values represent keyboard keys and key combinations. When using these values in a command, you must type the semicolons shown in this table in addition to the semicolons required by the escape sequence. The codes in parentheses are not available on some keyboards. ANSI.SYS might not interpret some of the codes in parentheses unless you specify the **/x** switch in the **device** command for ANSI.SYS.

- *String* is either the ASCII code for a single character or a string contained in quotation marks. For example, both 65 and "A" can be used to represent an uppercase A.

IMPORTANT Some of the values in the following table are not valid for all computers. Check your computer's documentation for values that are different.

ASCII Key Codes

Key	*Code*	SHIFT+*code*	CTRL+*code*	ALT+*code*
F1	0;59	0;84	0;94	0;104
F2	0;60	0;85	0;95	0;105
F3	0;61	0;86	0;96	0;106
F4	0;62	0;87	0;97	0;107
F5	0;63	0;88	0;98	0;108
F6	0;64	0;89	0;99	0;109
F7	0;65	0;90	0;100	0;110
F8	0;66	0;91	0;101	0;111
F9	0;67	0;92	0;102	0;112
F10	0;68	0;93	0;103	0;113
F11	0;133	0;135	0;137	0;139
F12	0;134	0;136	0;138	0;140
HOME	0;71	55	0;119	—
UP ARROW	0;72	56	(0;141)	—
PAGE UP	0;73	57	0;132	—
LEFT ARROW	0;75	52	0;115	—
RIGHT ARROW	0;77	54	0;116	—
END	0;79	49	0;117	—
DOWN ARROW	0;80	50	(0;145)	—
PAGE DOWN	0;81	51	0;118	—
INSERT	0;82	48	(0;146)	—
DELETE	0;83	46	(0;147)	—
HOME (gray key)	(224;71)	(224;71)	(224;119)	(224;151)
UP ARROW (gray key)	(224;72)	(224;72)	(224;141)	(224;152)
PAGE UP (gray key)	(224;73)	(224;73)	(224;132)	(224;153)
LEFT ARROW (gray key)	(224;75)	(224;75)	(224;115)	(224;155)
RIGHT ARROW (gray key)	(224;77)	(224;77)	(224;116)	(224;157)
END (gray key)	(224;79)	(224;79)	(224;117)	(224;159)

ASCII Key Codes (*continued*)

Key	*Code*	SHIFT+*code*	CTRL+*code*	ALT+*code*
DOWN ARROW (gray key)	(224;80)	(224;80)	(224;145)	(224;154)
PAGE DOWN (gray key)	(224;81)	(224;81)	(224;118)	(224;161)
INSERT (gray key)	(224;82)	(224;82)	(224;146)	(224;162)
DELETE (gray key)	(224;83)	(224;83)	(224;147)	(224;163)
PRINT SCREEN	—	—	0;114	—
PAUSE/BREAK	—	—	0;0	—
BACKSPACE	8	8	127	(0)
ENTER	13	—	10	(0,28)
TAB	9	0;15	(0;148)	(0;165)
NULL	0;3	—	—	—
A	97	65	1	0;30
B	98	66	2	0;48
C	99	66	3	0;46
D	100	68	4	0;32
E	101	69	5	0;18
F	102	70	6	0;33
G	103	71	7	0;34
H	104	72	8	0;35
I	105	73	9	0;23
J	106	74	10	0;36
K	107	75	11	0;37
L	108	76	12	0;38
M	109	77	13	0;50
N	110	78	14	0;49
O	111	79	15	0;24
P	112	80	16	0;25
Q	113	81	17	0;16
R	114	82	18	0;19
S	115	83	19	0;31
T	116	84	20	0;20
U	117	85	21	0;22

ASCII Key Codes (*continued*)

Key	*Code*	SHIFT+*code*	CTRL+*code*	ALT+*code*
V	118	86	22	0;47
W	119	87	23	0;17
X	120	88	24	0;45
Y	121	89	25	0;21
Z	122	90	26	0;44
1	49	33	—	0;120
2	50	64	0	0;121
3	51	35	—	0;122
4	52	36	—	0;123
5	53	37	—	0;124
6	54	94	30	0;125
7	55	38	—	0;126
8	56	42	—	0;126
9	57	40	—	0;127
0	48	41	—	0;129
-	45	95	31	0;130
=	61	43	—	0;131
[	91	123	27	0;26
]	93	125	29	0;27
\	92	124	28	0;43
;	59	58	—	0;39
'	39	34	—	0;40
,	44	60	—	0;51
.	46	62	—	0;52
/	47	63	—	0;53
`	96	126	—	(0;41)
ENTER (keypad)	13	—	10	(0;166)
/ (keypad)	47	47	(0;142)	(0;74)
* (keypad)	42	(0;144)	(0;78)	—
- (keypad)	45	45	(0;149)	(0;164)
+ (keypad)	43	43	(0;150)	(0;55)
5 (keypad)	(0;76)	53	(0;143)	—

Examples

To exchange the backslash and question-mark keys by using literal strings, type the following escape sequence:

```
ESC["\";"?"pESC["?";"\"p
```

To exchange the backslash and question-mark keys by using each key's ASCII value, type the following escape sequence:

```
ESC[92;63pESC[63;92p
```

To restore the backslash and question-mark keys to their original meanings, type the following escape sequence:

```
ESC[92;92pESC[63;63p
```

DISPLAY.SYS

Supports code-page switching for the console.

For an introduction to preparing the console for code pages, see Chapter 13, "Customizing for International Use."

Syntax

device=[*drive*:][*path*]**display.sys con**[:]=(*type*[,[*hwcp*][,*n*]])

device=[*drive*:][*path*]**display.sys con**[:]=(*type*[,[*hwcp*][,(*n*,*m*)]])

Parameters

[*drive*:][*path*]

Specifies the location of the DISPLAY.SYS file.

type

Specifies the display adapter in use. Valid values include **ega** and **lcd**. The **ega** value supports both EGA and VGA display adapters. If you omit the *type* parameter, DISPLAY.SYS checks the hardware to determine which display adapter is in use. You can also specify **cga** and **mono** as values for *type*, but they have no effect because code-page switching is not enabled for these devices.

hwcp

Specifies the number of the code page that your hardware supports. The following list shows the code pages that MS-DOS supports and the country or language for each:

437	United States
850	Multilingual (Latin I)
852	Slavic (Latin II)
860	Portuguese
863	Canadian-French
865	Nordic

For more information about code pages, see the Appendix, "Keyboard Layouts and Code Pages."

n Specifies the number of code pages the hardware can support in addition to the primary code page specified for the *hwcp* parameter. Valid values for *n* are in the range 0 through 6. This value depends on your hardware. For EGA display adapters, the maximum value for *n* is 6; for LCD display adapters, the maximum value for *n* is 1.

m Specifies the number of subfonts the hardware supports for each code page. The default value is 2 if *type* is **ega**, and 1 if *type* is **lcd**.

Notes

Using DISPLAY.SYS with monochrome or CGA display adapters

Because monochrome and CGA display adapters do not support code-page switching, using DISPLAY.SYS with either type of adapter has no effect.

Installing a third-party console driver

If you install both DISPLAY.SYS and a third-party console driver, such as VT52.SYS, the third-party device driver must be installed first. Otherwise, the third-party device driver may disable DISPLAY.SYS.

Example

Suppose you want DISPLAY.SYS to support an EGA display adapter with a United States code page and the potential for two or more code pages without subfonts. To do this and to specify that DISPLAY.SYS is in the DOS directory on drive C, add the following line to your CONFIG.SYS file:

```
device=c:\dos\display.sys con:=(ega,437,2)
```

DRIVER.SYS

Creates a logical drive that you can use to refer to a physical floppy disk drive.

A logical drive is a pointer to a physical disk drive in your system. The logical drive is associated with a drive letter (for example, A or B). You can specify parameters to describe the disk drive to MS-DOS.

For an introduction to using logical drives, see Chapter 11, "Customizing Your System."

Syntax

device=[*drive*:][*path*]**driver.sys /d:***number* [**/c**] [**/f:***factor*] [**/h:***heads*] [**/s:***sectors*] [**/t:***tracks*]

Parameter

[*drive*:][*path*]

Specifies the location of the DRIVER.SYS file.

Switches

/d:*number*

Specifies the number of the physical floppy disk drive. Valid values for *number* are in the range 0 through 127. The first physical floppy disk drive (drive A) is drive 0; a second physical floppy disk drive is drive 1; a third physical floppy disk drive, which must be external, is 2. For a computer with one floppy disk drive, both drives A and B are numbered 0; for a computer with multiple floppy disk drives, drive B is numbered 1.

/c Specifies that the physical disk drive can detect whether the drive door is closed (change-line support).

/f:*factor*

Specifies the type of disk drive. Valid values for *factor* are as follows:

0	160K/180K or 320K/360K
1	1.2 megabyte (MB)
2	720K (3.5-inch disk) or other
7	1.44 MB (3.5-inch disk)
9	2.88 MB (3.5-inch disk)

The default value for *factor* is 2.

Generally, if you use the **/f** switch, you can omit the **/h**, **/s**, and **/t** switches. Check the default values for these switches to make sure they are correct for the type of disk drive you are using. To determine the appropriate values for the disk drive, see the disk-drive manufacturer's documentation.

If you specify the **/h**, **/s**, and **/t** switches, you can omit the **/f** switch.

/h:*heads*

Specifies the number of heads in the disk drive. Valid values for *heads* are in the range 1 through 99. The default value is 2. To determine the correct value for your disk drive, see the disk-drive manufacturer's documentation.

/s:*sectors*

Specifies the number of sectors per track. Valid values for *sectors* are in the range 1 through 99. The default value depends on the value of **/f:***factor*, as follows:

/f:0	**/s:9**
/f:1	**/s:15**
/f:2	**/s:9**
/f:7	**/s:18**
/f:9	**/s:36**

To determine the correct value for your disk drive, see the disk-drive manufacturer's documentation.

/t:*tracks*

Specifies the number of tracks per side on the block device. Valid values for *tracks* are in the range 1 through 999. The default value is 80, unless **/f:***factor* is 0, in which case the default value is 40. To determine the correct value for your disk drive, see the disk-drive manufacturer's documentation.

Notes

Disk-drive change-line support

The term *change-line support* means that a physical disk drive can detect when the drive door is open. Change-line support allows faster MS-DOS operation with floppy disks. If you specify the **/c** switch, it indicates to MS-DOS that the physical disk drive can support change-line error detection. To determine whether your disk drive has change-line support, see the disk-drive manufacturer's documentation.

Modifying or redefining a supported physical disk drive

For information about modifying the parameters of a physical disk drive that is supported by your hardware, see the **drivparm** command in Chapter 14, "Commands." You can also use DRIVER.SYS to redefine a physical floppy disk drive. For information about redefining a floppy disk drive, see Chapter 11, "Customizing Your System."

Limitations on DRIVER.SYS

You cannot use DRIVER.SYS with hard disk drives. For information about substituting a logical drive letter for a hard disk drive, see the **subst** command in Chapter 14, "Commands."

Creating a duplicate logical drive

Suppose you want to use one physical floppy disk drive to copy files from one floppy disk to another. Because you cannot copy from and to the same logical drive by using the **copy** or **xcopy** command, you must assign a second drive letter to that physical drive.

If your system has just one physical floppy disk drive, you do not need to install DRIVER.SYS for this purpose. MS-DOS already assigns both logical drive A and logical drive B to that drive. Just copy files from drive A to drive B and switch disks when MS-DOS prompts you.

If your system has more than one floppy disk drive, then you need to use DRIVER.SYS to assign a second drive letter to the physical floppy disk drive.

Creating a new logical drive with different parameters

If you use DRIVER.SYS to assign a logical drive that has parameters different from those of the previously assigned logical drive, then the parameters of the previous logical drive will be invalid. Therefore, you should no longer use the drive letter corresponding to the previous logical drive.

Examples

To add an external 720K drive to your system, add the following line to your CONFIG.SYS file:

```
device=driver.sys /d:2
```

Since no location is specified, MS-DOS searches for DRIVER.SYS in the root directory of your startup drive.

Suppose you want to use a single 1.44-megabyte external disk drive to copy files from one floppy disk to another. To do this, you must add two identical **device** commands for DRIVER.SYS in your CONFIG.SYS file. This procedure assigns two logical drive letters to the same physical drive. You can then swap disks in the same drive during the copying process. The following example shows how to do this:

```
device=driver.sys /d:2 /f:7
device=driver.sys /d:2 /f:7
```

EGA.SYS

Saves and restores the display when Task Swapper is used with EGA monitors. If you have an EGA monitor, you must install the EGA.SYS device driver before using Task Swapper.

For an introduction to using Task Swapper, see Chapter 3, "MS-DOS Shell Basics."

Syntax

device=[*drive*:][*path*]**ega.sys**

Parameter

[*drive*:][*path*]

Specifies the location of the EGA.SYS file.

Note

If you are using a mouse on a system that has an EGA monitor, you can save memory by installing EGA.SYS before you install your mouse driver.

EMM386.EXE

Simulates expanded memory while using extended memory and provides access to the upper memory area on a computer that has an 80386 or higher processor.

EMM386.EXE uses extended memory to simulate expanded memory for programs that can use expanded memory. EMM386.EXE also makes it possible to load programs and device drivers into the upper memory area. For an introduction to using the EMM386.EXE device driver, see Chapter 12, "Optimizing Your System."

Syntax

device=[*drive*:][*path*]**emm386.exe** [**on**|**off**|**auto**] [*memory*] [**w=on**|**w=off**] [**m***x*|**frame**=*address*|**p***mmmm*] [**p***n*=*address*] [**x**=*mmmm*-*nnnn*] [**i**=*mmmm*-*nnnn*] [**b**=*address*] [**L**=*minXMS*] [**a**=*altregs*] [**h**=*handles*] [**d**=*nnn*] [**ram**] [**noems**]

Parameters

[*drive*:][*path*]

Specifies the location of the EMM386.EXE file.

[on|off|auto]

Activates the EMM386.EXE device driver (if set to **on**), or suspends the EMM386.EXE device driver (if set to **off**), or places the EMM386.EXE device driver in auto mode (if set to **auto**). Auto mode enables expanded-memory support only when a program calls for it. The default value is **on**. Use the **emm386** command to change this value after EMM386 has started.

memory

Specifies the amount of memory (in kilobytes) that you want to allocate to EMM386.EXE. Values for *memory* are in the range 16 through 32768. The default value is 256. EMM386.EXE rounds the value down to the nearest multiple of 16. If you are using expanded memory, this value is in addition to the memory used for low-memory backfilling.

Switches

w=on|w=off

Enables or disables support for the Weitek coprocessor. The default setting is **w=off**.

m*x* Specifies the address of the page frame. Valid values for *x* are in the range 1 through 14. The following list shows each value and its associated base address in hexadecimal format:

1 => C000h	8 => DC00h
2 => C400h	9 => E000h
3 => C800h	10 => 8000h
4 => CC00h	11 => 8400h
5 => D000h	12 => 8800h
6 => D400h	13 => 8C00h
7 => D800h	14 => 9000h

Values in the range 10 through 14 should be used only on computers with 512K of memory.

frame=*address*

Specifies the page-frame segment base directly. To specify a specific segment-base address for the page frame, use the **frame** switch and specify the address you want. Valid values for *address* are in the ranges 8000h through 9000h and C000h through E000h, in increments of 400h.

/p*mmmm*

Specifies the address of the page frame. Valid values for *mmmm* are in the ranges 8000h through 9000h and C000h through E000h, in increments of 400h.

p*n*=*address*

Specifies the segment address of a specific page, where *n* is the number of the page you are specifying and *address* is the segment address you want. Valid values for *n* are in the range 0 through 255. Valid values for *address* are in the ranges 8000h through 9C00h and C000h through EC00h, in increments of 400h.

The addresses for pages 0 through 3 must be contiguous in order to maintain compatibility with version 3.2 of the Lotus/Intel/Microsoft Expanded Memory Specification (LIM EMS).

If you use the **m***x* switch, the **frame** switch, or the **/p***mmmm* switch, you cannot specify the addresses for pages 0 through 3 for the **/p***mmmm* switch.

x=*mmmm-nnnn*

Prevents EMM386.EXE from using a particular range of segment addresses for an EMS page. Valid values for *mmmm* and *nnnn* are in the range A000h through FFFFh and are rounded down to the nearest 4-kilobyte boundary. The **x** switch takes precedence over the **i** switch if the two ranges overlap.

i=*mmmm-nnnn*

Specifies a range of segment addresses to be used (included) for an EMS page or for random-access memory (RAM). Valid values for *mmmm* and *nnnn* are in the range A000h through FFFFh and are rounded down to the nearest 4-kilobyte boundary. The **x** switch takes precedence over the **i** switch if the two ranges overlap.

b=*address*

Specifies the lowest segment address available for EMS "banking" (swapping of 16-kilobyte pages). Valid values are in the range 1000h through 4000h. The default value is 4000h.

L=*minXMS*

Ensures that the specified amount (in kilobytes) of extended memory will still be available after you load EMM386.EXE. The default value is 0.

a=*altregs*

Specifies how many fast alternate register sets (used for multitasking) you want to allocate to EMM386.EXE. Valid values are in the range 0 through 254. The default value is 7. Every alternate register set adds about 200 bytes to the size in memory of EMM386.EXE.

h=*handles*

Specifies how many handles EMM386.EXE can use. Valid values are in the range 2 through 255. The default value is 64.

d=*nnn*

Specifies how many kilobytes of memory should be reserved for buffered direct memory access (DMA). Discounting floppy-disk DMA, this value should reflect the largest DMA transfer that will occur while EMM386.EXE is active. Valid values for *nnn* are in the range 16 through 256. The default value is 16.

ram

Provides access to both expanded memory and the upper memory area.

noems

Provides access to the upper memory area but prevents access to expanded memory.

Notes

Must install HIMEM.SYS before EMM386.EXE

You must include a **device** command for the HIMEM.SYS device driver in your CONFIG.SYS file before the **device** command for EMM386.EXE.

Using EMM386.EXE memory switches

Unless you want to use EMM386.EXE to provide access to the upper memory area, you need not specify memory switches on the **device** command line. EMM386.EXE usually runs properly with the default values. In some cases, however, you might want to control how EMM386.EXE uses memory. For example, some programs will run better if you allocate more expanded memory. Also, you can control where EMM386.EXE puts the EMS page frame, or which segments it uses for EMS pages. You can use as many of these memory switches as you want, in any order you want. For more information, see Chapter 12, " Optimizing Your System."

CAUTION Use EMM386.EXE parameters carefully. You can disable your system if you use them incorrectly.

Using EMM386.EXE to provide access to the upper memory area

In addition to providing access to expanded memory, EMM386.EXE provides access to the upper memory area, which you can use to load certain programs and device drivers. You must use either the **ram** or **noems** switch to provide access to the upper memory area.

To give MS-DOS access to the upper memory area but not to expanded memory, use the **noems** switch. To give MS-DOS access to both the upper memory area and expanded memory, use the **ram** switch. The **ram** switch provides access to less of the upper memory area for running device drivers and programs than does the **noems** switch. In either case, you must include the **dos=umb** command in your CONFIG.SYS file. The device command for EMM386.EXE must precede any **devicehigh** commands.

If you are using a Virtual Control Program Interface (VCPI) application, such as Lotus 1-2-3 version 3.1, use the **/ram** switch to provide access to expanded memory. For more information about using the upper memory area, see Chapter 12, "Optimizing Your System."

Inadequate space for page frame

If EMM386.EXE is unable to find 64K of continguous space for the page frame, the following message is displayed:

```
Unable to set base address
```

Examples

To start EMM386 as an expanded-memory emulator, using the default values, add the following lines to your CONFIG.SYS file:

```
device=himem.sys
device=emm386.exe
```

Since no location is specified, MS-DOS searches for EMM386.EXE in the root directory of your startup drive.

To allocate 4096K of memory to EMM386.EXE and specify that the EMM386.EXE file is located in the DOS directory on drive C, add the following line to your CONFIG.SYS file:

```
device=c:\dos\emm386.exe 4096
```

To emulate expanded memory, specify the segment-base address D000h for the EMS page frame, and allocate 512K of memory to EMM386.EXE, use one of the following commands:

```
device=emm386.exe 512 frame=d000

device=emm386.exe 512 p0=d000 p1=d400 p2=d800 p3=dc00
```

Suppose that, in addition to specifying the conditions set in the preceding commands, you want to prevent EMM386 from using the memory address E000h through EC00h. To do this and to specify that EMM386 can use 127 handles, add the following line to your CONFIG.SYS file:

```
device=emm386.exe 512 frame=d000 x=e000-ec00 h=127
```

To provide access to the upper memory area but not emulate expanded memory, add the following line to your CONFIG.SYS file:

```
device=emm386.exe noems
```

To provide access to the upper memory area and emulate expanded memory, add the following line to your CONFIG.SYS file:

```
device=emm386.exe ram
```

HIMEM.SYS

Manages the use of extended memory.

HIMEM.SYS manages programs' use of extended memory and of the high memory area (HMA). This prevents programs from simultaneously using the same area of memory. You install the HIMEM.SYS device driver by adding a **device** command for it in your CONFIG.SYS file. The **device** command for HIMEM.SYS must precede any **device** commands for programs or device drivers that use extended memory (such as SMARTDRV.SYS, RAMDRIVE.SYS, and EMM386.EXE). For an introduction to using extended memory, see Chapter 12, "Optimizing Your System."

Syntax

device=[*drive*:][*path*]**himem.sys** [**/hmamin**=*m*] [**/numhandles**=*n*] [**/int15**=*xxxx*] [**/machine:***xxxx*] [**/a20control:on|off**] [**/shadowram:on|off**] [**/cpuclock:on|off**]

Parameter

[*drive*:][*path*]

Specifies the location of the HIMEM.SYS file.

Switches

/hmamin=*m*

Specifies the amount of memory (in kilobytes) a program must use before HIMEM.SYS permits the program to use the high memory area. Valid values for *m* are in the range 0 through 63. The default value is 0.

/numhandles=*n*

Specifies the maximum number of extended-memory-block (EMB) handles that can be used simultaneously. Valid values for *n* are in the range 1 through 128. The default value is 32. Each additional handle requires an additional 6 bytes of resident memory.

/int15=*xxxx*

Allocates the specified amount of extended memory (in kilobytes) for the Interrupt 15h interface. Some older programs use a conflicting extended-memory scheme. To use memory allocated by this switch, programs must recognize VDisk headers. To ensure enough memory is available, add 64 to the value you want to specify for *xxxx*. Valid values for *xxxx* are in the range 64 through 65535. If you specify a value less than 64, the value becomes 0. The default value is 0.

/machine:*xxxx*

Specifies the A20 handler to be used. An A20 handler is a part of your computer that gives it access to the high memory area. The *xxxx* value can be any of the following codes or their equivalent numbers:

Code	Number	A20 handler
at	1	IBM PC/AT
ps2	2	IBM PS/2
ptlcascade	3	Phoenix Cascade BIOS
hpvectra	4	HP Vectra (A and A+)
att6300plus	5	AT&T 6300 Plus
acer1100	6	Acer 1100
toshiba	7	Toshiba 1600 and 1200XE
wyse	8	Wyse 12.5 MHz 286
tulip	9	Tulip SX
zenith	10	Zenith ZBIOS
at1	11	IBM PC/AT
at2	12	IBM PC/AT (alternative delay)
css	12	CSS Labs
at3	13	IBM PC/AT (alternative delay)
philips	13	Philips
fasthp	14	HP Vectra

Typically, HIMEM.SYS detects which A20 handler is being used. You might have to specify a value for this setting if the A20 handler reports problems or if you have problems using MS-DOS in the high memory area. If the machine you are using is not listed, see the README.TXT online file for additional values. The default value for the **/machine:***xxxx* switch is **at** or **1**.

/a20control:on|off

Specifies whether HIMEM.SYS is to take control of the A20 line even if A20 was on when HIMEM.SYS was loaded. If you specify **/a20control:off**, HIMEM.SYS takes control of the A20 line only if A20 was off when HIMEM.SYS was loaded. The default setting is **/a20control:on**.

/shadowram:on|off

Specifies whether HIMEM.SYS is to switch off *shadow RAM*—random-access memory (RAM) used for read-only memory (ROM)—and add that RAM to its memory pool. If your computer has less than 2 megabytes of RAM, the default setting is **/shadowram:off**. This parameter is supported only on some computers.

/cpuclock:on|off

Specifies whether HIMEM.SYS is to affect the clock speed of your computer. If your computer's speed changes when you install HIMEM.SYS, specifying **/cpuclock:on** might correct the problem. Enabling this switch slows down HIMEM.SYS.

Notes

Default memory allocation

Only one program can use the high memory area at a time. If you omit the **/hmamin=***m* switch (or set it to 0), HIMEM.SYS reserves the HMA for the first program that requests it. HIMEM.SYS reserves the HMA for the first program that meets the memory requirements set by the **/hmamin=***m* switch. To ensure the most efficient use of your system's high memory area, you should set **/hmamin=***m* to the amount of memory required by the program that uses the most HMA memory.

Loading MS-DOS into the high memory area

HIMEM.SYS or another XMS driver must be loaded before you can load MS-DOS into the high memory area (HMA). You load MS-DOS into the HMA by using the **dos=high** command in your CONFIG.SYS file.

Examples

To install HIMEM.SYS, using the default values, add the following line to your CONFIG.SYS file:

```
device=himem.sys
```

Since no location is specified, MS-DOS searches for HIMEM.SYS in the root directory of your startup drive.

Suppose you want a program to use at least 40K of memory before it has access to the high memory area. To specify this and that HIMEM.SYS is located in the DOS directory of drive C, add the following line to your CONFIG.SYS file:

```
device=c:\dos\himem.sys /hmamin=40
```

To install HIMEM.SYS and specify the A20 handler for an IBM PS/2 computer, add either of the following lines to your CONFIG.SYS file:

```
device=himem.sys /machine:ps2

device=himem.sys /machine:2
```

Suppose you want to install HIMEM.SYS and allow simultaneous use of as many as 128 extended-memory handles. Also suppose that HIMEM.SYS is located in the DEVICES directory on drive D. To do this, add the following line to your CONFIG.SYS file:

```
device=d:\devices\himem.sys /numhandles=128
```

PRINTER.SYS

Supports code-page switching for the parallel ports PRN, LPT1, LPT2, and LPT3.

For an introduction to using code pages, see Chapter 13, "Customizing for International Use."

Syntax

device=[*drive*:][*path*]**printer.sys lpt***x*=(*type*[,[*hwcp*][,*n*]])

Parameters

[*drive*:][*path*]

Specifies the location of the PRINTER.SYS file.

lpt*x*

Specifies the number of the parallel port for which you want to support code-page switching.

type Specifies the printer in use. The following list shows valid values for *type* and the printers represented by each value:

4201	IBM Proprinters II and III Model 4201 IBM Proprinters II and III XL Model 4202
4208	IBM Proprinter X24E Model 4207 IBM Proprinter XL24E Model 4208
5202	IBM Quietwriter III Model 5202

hwcp

Specifies the code page your hardware supports. The following list shows the code pages that MS-DOS supports and the country or language for each:

437 United States
850 Multilingual (Latin I)
852 Slavic (Latin II)
860 Portuguese
863 Canadian-French
865 Nordic

n Specifies the number of code pages your hardware can support in addition to the code page specified in the *hwcp* parameter.

Example

The following command loads the PRINTER.SYS device driver for use with the IBM Proprinter X24E Model 4207, loads code page 850, and prepares PRINTER.SYS to support two additional code pages:

```
device=c:\dos\printer.sys lpt1:=(4208,850,2)
```

RAMDRIVE.SYS

Creates a RAM disk in your system's random-access memory (RAM) to simulate a hard disk drive. RAM disks are much faster than hard disk disks because the information they contain is always loaded into memory. RAM disks are temporary—any data you place on a RAM disk is lost when you turn off your computer. You can set up as many RAM disks as you want, limited only by the amount of memory your computer has. To do this, add one RAMDRIVE.SYS line to your CONFIG.SYS file for each additional RAM disk.

Syntax

device=[*drive*:][*path*]**ramdrive.sys** [*DiskSize SectorSize*] [**/e**|**/a**]

device=[*drive*:][*path*]**ramdrive.sys** [*DiskSize SectorSize NumEntries*] [**/e**|**/a**]

Parameters

[*drive*:][*path*]

Specifies the location of the RAMDRIVE.SYS file.

DiskSize

Specifies the size (in kilobytes) of the RAM disk. Valid values for *DiskSize* are in the range 16 through 4096. The default value is 64.

SectorSize

Specifies the disk sector size (in bytes). Valid values for *SectorSize* are 128, 256, and 512. The default value is 512. If you include a value for the *SectorSize* parameter, you must also include a value for the *DiskSize* parameter. Although you can change the *SectorSize* value, the default value is strongly recommended.

NumEntries

Specifies the number of files and directories you can create in the RAM disk's root directory. Valid values for *NumEntries* are in the range 2 through 1024. The default value is 64. If you include a value for the *NumEntries* parameter, you must also include values for the *DiskSize* and *SectorSize* parameters.

Switches

/e Creates the RAM disk in extended memory instead of in expanded or conventional memory.

/a Creates the RAM disk in expanded memory instead of in extended or conventional memory.

Notes

Using the *NumEntries* parameter

RAMDRIVE.SYS rounds the number you specify up to the nearest sector boundary. If there is not enough memory to create the RAM disk as specified, RAMDRIVE.SYS attempts to create it with a limit of 16 directory entries. This can result in a RAM disk with a different limit from the one you specified.

Using conventional memory

Although specifying a memory type is optional, it is strongly recommended. If you omit both the **/e** and **/a** switches, RAMDRIVE.SYS uses your system's conventional memory. It is not a good idea to use conventional memory for a RAM disk, because this reduces available work space for programs. However, if you don't have extended memory, expanded memory, or a hard disk drive, you might want to use conventional memory for a RAM disk. A RAM disk can increase the speed of a floppy disk system significantly enough that it may be worth the loss of some conventional memory.

Using extended memory

If your system has extended memory installed (starting at the 1-megabyte boundary), you can use this extended memory for one or more RAM disks. For RAMDRIVE.SYS to use extended memory, you must first install HIMEM.SYS or another extended-memory manager that conforms to the Lotus/Intel/Microsoft/AST eXtended Memory Specification (XMS). In your CONFIG.SYS file, the **device** command that installs the XMS extended-memory manager must precede the commands that install the RAM disk.

Using expanded memory

For RAMDRIVE.SYS to use expanded memory, you must configure your system so that it provides expanded memory. In your CONFIG.SYS file, the **device** command that installs the expanded-memory manager (such as EMM386.EXE) must precede the **device** command that installs RAMDRIVE.SYS. The expanded-memory manager must conform to the Lotus/Intel/Microsoft Expanded Memory Specification (LIM EMS).

Increasing the efficiency of a RAM disk

For the best results with a RAM disk, you can define a TEMP environment variable and set it to point to a subdirectory on the RAM disk. For more information about setting up a RAM disk, see Chapter 12, "Optimizing Your System."

Examples

To create a RAM disk in extended memory and allocate 64K (the default amount) of extended memory to RAMDRIVE.SYS, add the following line to your CONFIG.SYS file:

```
device=ramdrive.sys /e
```

Since no location is specified, MS-DOS searches for RAMDRIVE.SYS in the root directory of your startup drive.

Suppose you want to install RAMDRIVE.SYS in expanded memory and allocate 4 MB (4096K) of expanded memory to the RAM disk. To do this and to specify that RAMDRIVE.SYS is located in the DOS directory on drive C, add the following line to your CONFIG.SYS file:

```
device=c:\dos\ramdrive.sys 4096 /a
```

Now suppose you want to allocate 1024K of extended memory to RAMDRIVE.SYS and create a RAM disk that has 512-byte sectors and a limit of 1024 entries in its root directory. To do this and to specify that RAMDRIVE.SYS is located in the DEVICES directory on drive D, add the following line to your CONFIG.SYS file:

```
device=d:\devices\ramdrive.sys 1024 512 1024 /e
```

SETVER.EXE

Loads the MS-DOS version table into memory.

SETVER.EXE loads into memory the MS-DOS version table, which lists names of programs and the number of the MS-DOS version with which each program is designed to run. To display or modify the version table, use the **setver** command, which is described in Chapter 14, "Commands."

Syntax

device=[*drive*:][*path*]**setver.exe**

Parameter

[*drive*:][*path*]

Specifies the location of the SETVER.EXE file.

SMARTDRV.SYS

Creates a disk cache in extended or expanded memory.

A disk cache can significantly speed up MS-DOS disk operations. You can control the size of the SMARTDRV.SYS memory cache, and you can set up the disk cache in expanded or extended memory.

Syntax

device=[*drive*:][*path*]**smartdrv.sys** [*InitCacheSize*] [*MinCacheSize*] [**/a**]

Parameters

[*drive*:][*path*]

Specifies the location of the SMARTDRV.SYS file.

InitCacheSize

Specifies the initial size (in kilobytes) of the memory cache. Valid values for *InitCacheSize* are in the range 128 through 8192. The default value is 256.

MinCacheSize

Specifies a minimum cache size (in kilobytes). Some programs can reduce the cache size. If you don't specify a value, there is no minimum cache size (that is, a program can reduce the cache size to zero). This parameter is useful only if you are running Windows version 3.0 or later. For more information, see the *Microsoft Windows User's Guide*.

Switch

/a Specifies that you want SMARTDRV.SYS to put the disk cache in expanded memory. If you omit this switch, SMARTDRV.SYS puts the cache in extended memory.

Notes

Specifying the initial memory cache

SMARTDRV.SYS rounds the value you specify for the *InitCacheSize* parameter to the nearest multiple of 16. If you plan to run a program that uses expanded or extended memory, specify a cache size that leaves enough memory for that program after SMARTDRV.SYS is installed.

If there is not enough memory to create a cache of the size you specify, SMARTDRV.SYS creates a smaller cache, using the available memory.

Using extended memory

For SMARTDRV.SYS to use extended memory, you must first install HIMEM.SYS or another extended-memory manager that conforms to the Lotus/Intel/Microsoft/AST eXtended Memory Specification (XMS). In your CONFIG.SYS file, the **device** command that installs the extended-memory manager must precede the **device** command that installs SMARTDRV.SYS.

Using expanded memory

For SMARTDRV.SYS to use expanded memory, you must configure your system to provide expanded memory. In your CONFIG.SYS file, the **device** command that installs the expanded-memory manager (such as EMM386.EXE) must precede the **device** command that installs SMARTDRV.SYS. The expanded-memory manager must conform to the Lotus/Intel/Microsoft Expanded Memory Specification (LIM EMS).

Using SMARTDRV.SYS with an 80286 or 80386 computer

If you have an 80286 or 80386 computer, you will probably have the best results if you place the disk cache in extended memory.

Cannot run disk-compaction program

To avoid losing data, do not run a disk-compaction program while SMARTDRV.SYS is loaded.

Examples

To create a disk cache in extended memory and set a cache size of 256K (the default size), add the following line to your CONFIG.SYS file:

```
device=smartdrv.sys
```

Since no location is specified, MS-DOS searches for SMARTDRV.SYS in the root directory of your startup drive.

Suppose you want to create a disk cache in extended memory, allocate a cache size of 2048K, and ensure that programs cannot reduce the size of the cache to less than 1024K. To do this and to specify that SMARTDRV.SYS is located in the DOS directory on drive C, add the following line to your CONFIG.SYS file:

```
device=c:\dos\smartdrv.sys 2048 1024
```

Appendix

Appendix

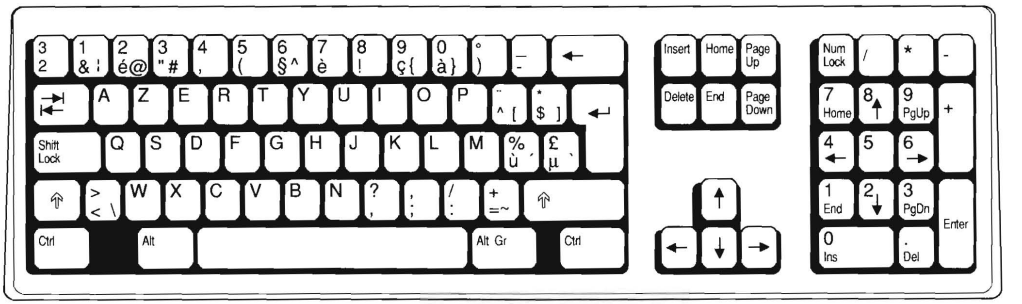
Keyboard Layouts and Code Pages

This appendix contains the following information:

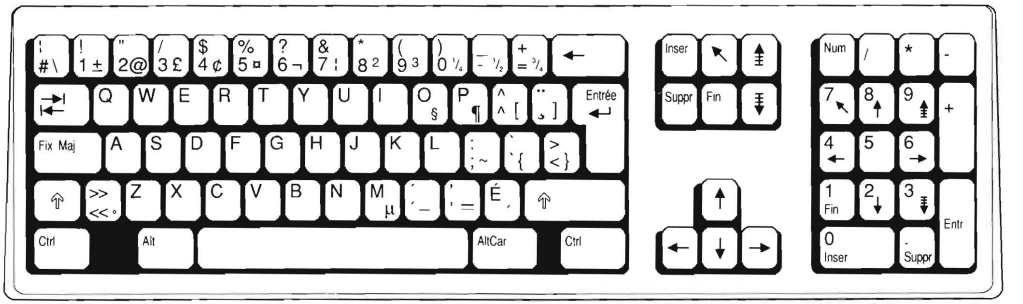
- Twenty keyboard layouts
- An explanation of how to use keys that contain more than two characters
- An explanation of how to create accented characters on a Brazilian keyboard
- An explanation of how to type characters that are not on your keyboard
- Tables that illustrate the six code pages that are supplied with MS-DOS

For information on changing your keyboard layout or active code page, see Chapter 13, "Customizing for International Use."

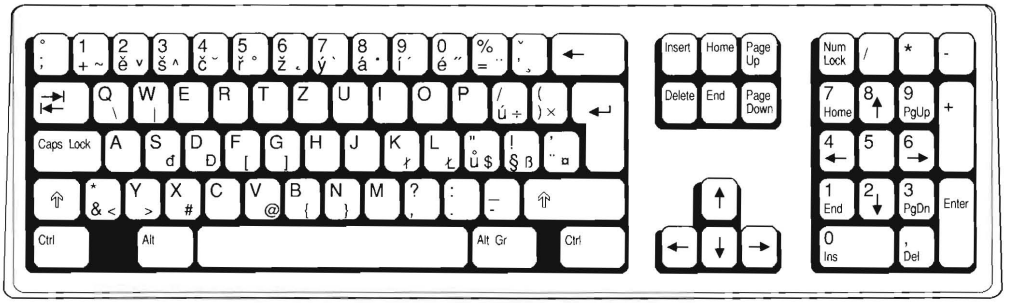
Keyboard Layouts



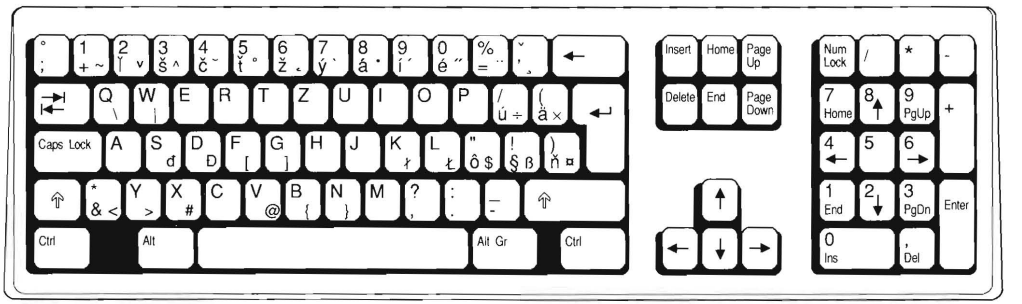
Belgium



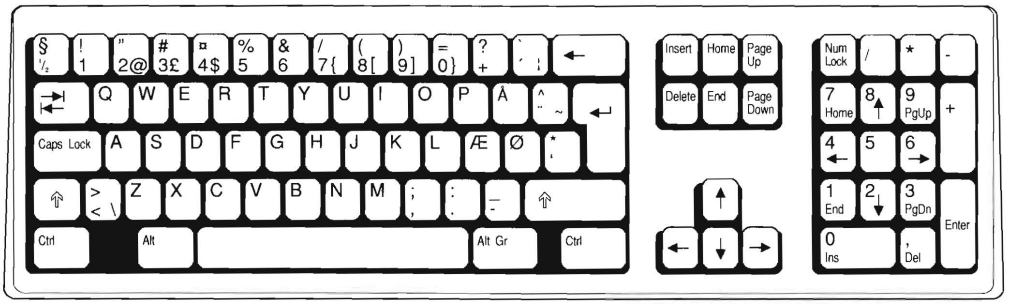
Canada



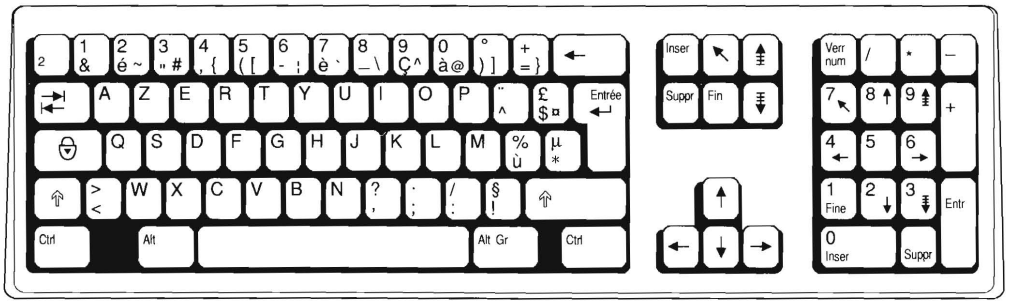
Czechoslovakia (Czech)



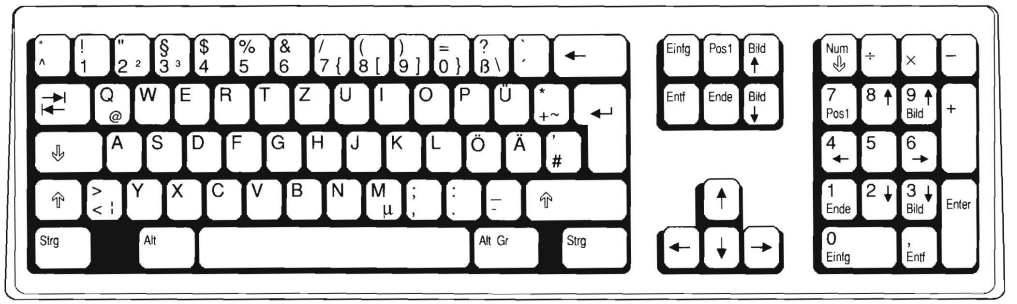
Czechoslovakia (Slovak)



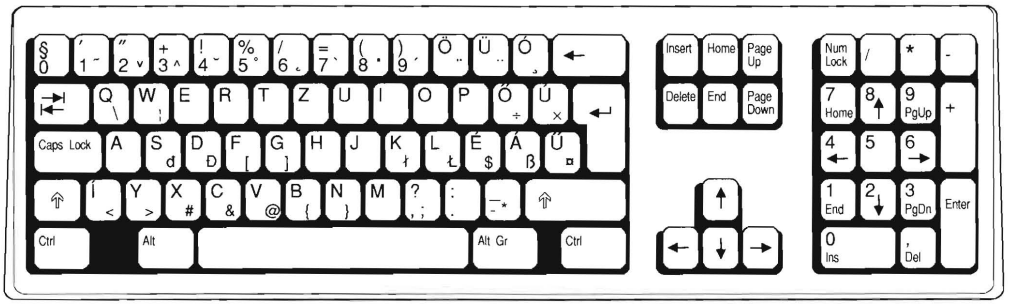
Denmark



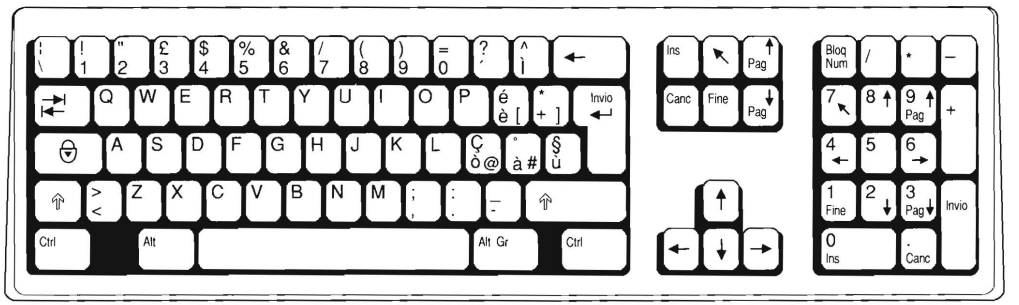
France



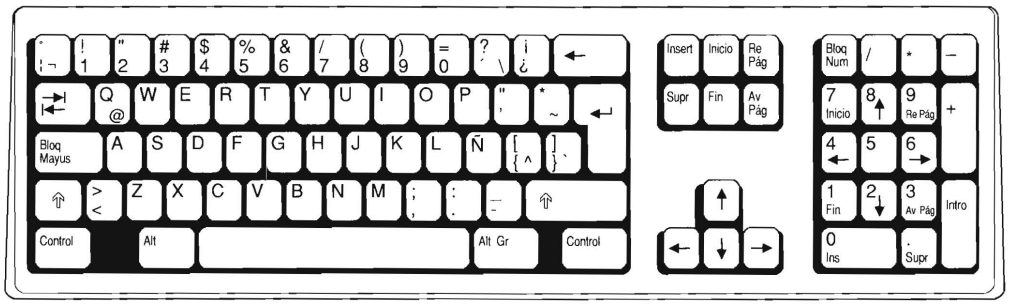
Germany



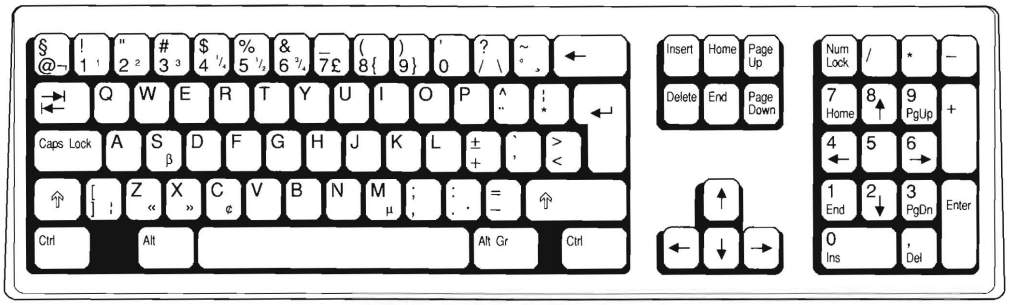
Hungary



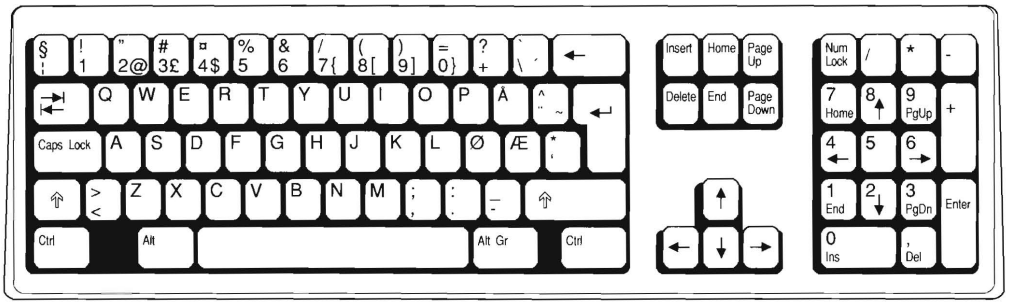
Italy



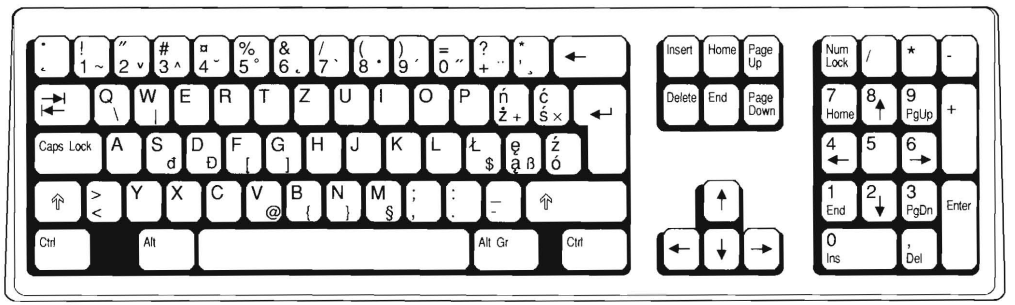
Latin America



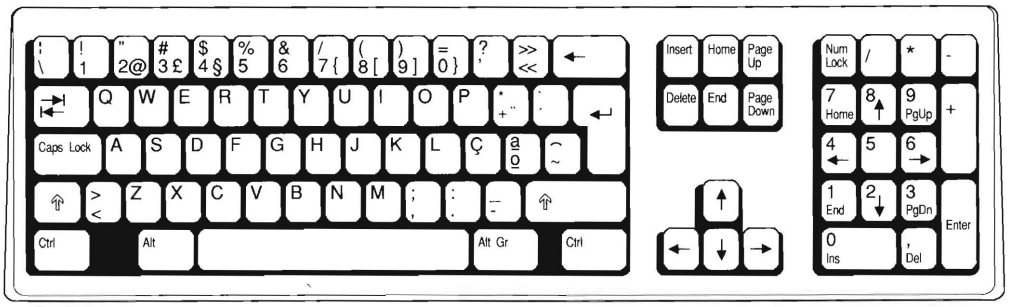
Netherlands



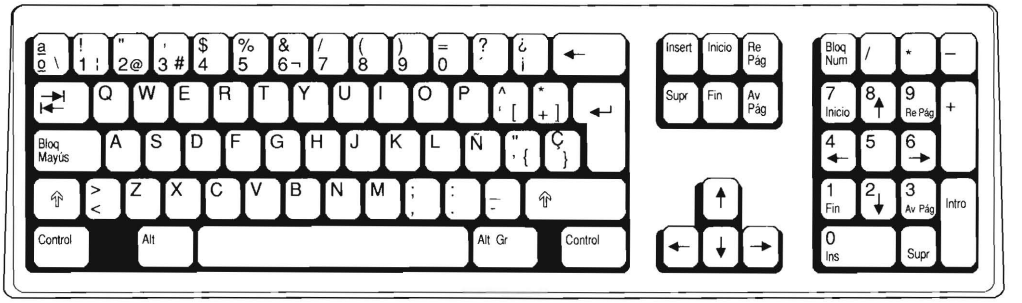
Norway



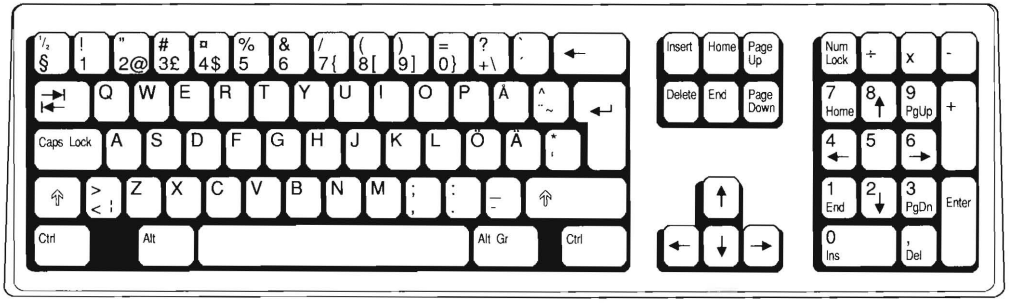
Poland



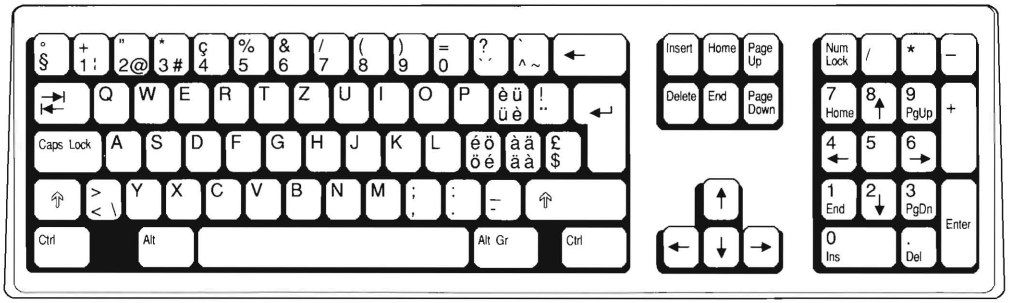
Portugal



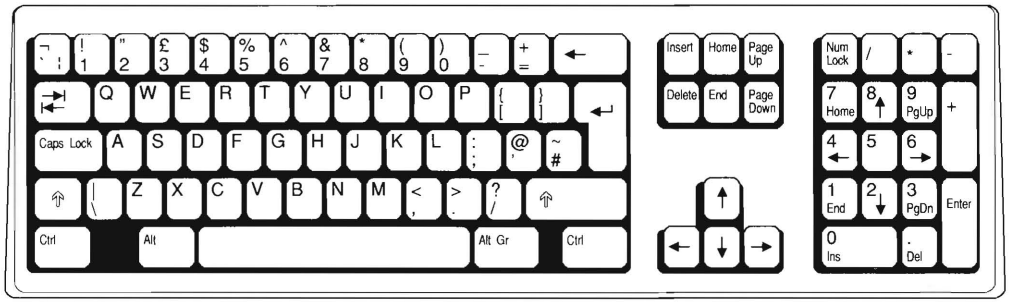
Spain



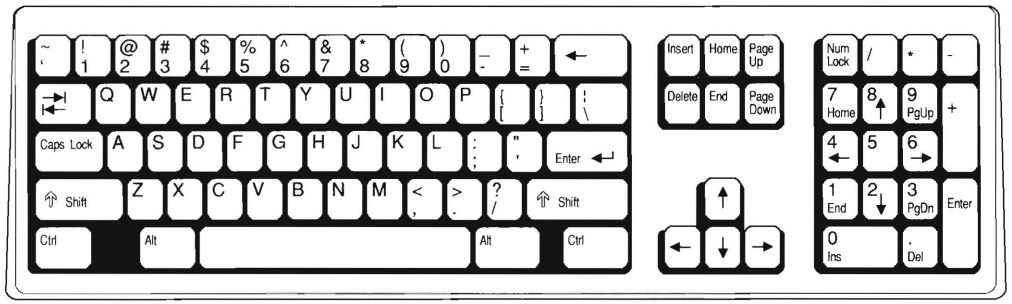
Sweden/Finland



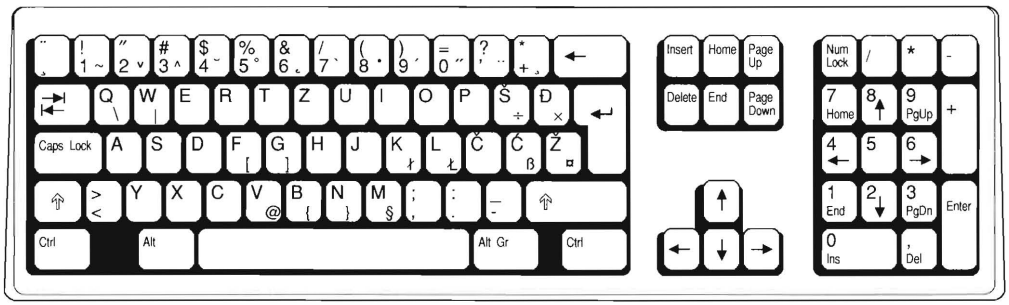
Switzerland



United Kingdom



United States



Yugoslavia

Using Keys That Contain More Than Two Characters

Some keyboards include keys that have three or four characters on each key. You can type the lower-left character by pressing the key. To type the upper-left character, press the SHIFT key and the character key. To type the lower-right and upper-right characters, use the key combinations described in the following table.

Country	Lower-right character Standard keyboard	Lower-right character Enhanced keyboard	Upper-right character All keyboards
Canada (Fr.)	ALT+SHIFT	ALT+GR	None
Denmark	ALT	ALT+GR	ALT+SHIFT
Finland	ALT	ALT+GR	ALT+SHIFT
Norway	ALT	ALT+GR	ALT+SHIFT
Sweden	ALT	ALT+GR	ALT+SHIFT
All others	ALT+CTRL	ALT+GR	None

Creating Accented Characters on a Brazilian Keyboard

If you install the Brazilian keyboard driver supplied with MS-DOS version 5.0, you can create accented characters on any standard U.S. keyboard. Use the following procedures:

▶ To create a character containing the acute accent:

- Type an apostrophe ('), and then type the character above which the acute accent should appear.

▶ To create a character containing the diaeresis:

- Type a double quotation mark ("), and then type the character above which the diaeresis should appear.

▶ To create a character containing the tilde:

- Type the tilde accent (~), and then type the character above which that accent should appear.

▶ To create the cedilla character:

- Type **'c**

If you type an invalid key sequence, you will hear a beep and MS-DOS will display the two characters separately.

Code-Page Tables

A code page consists of a set of characters that have corresponding numbers. MS-DOS version 5.0 reserves characters 0 through 31 for control characters. Your keyboard contains characters 32 through 127. Characters 128 and above are *extended characters*.

To produce extended characters and the few control characters that MS-DOS displays, use the following procedure.

▶ To type an ANSI character that is not on your keyboard:

1. Locate the character on the code-page table that matches the active code page.

   To display the active code page, use the **chcp** command.

2. Hold down ALT and use the numeric keypad to type the number that corresponds to the character you want.

For example, to type the Î character when using code page 850, hold down ALT while you type **215** on the numeric keypad.

The following are code-page tables for the six code pages included with MS-DOS version 5.0.

0		32		64	@	96	`	128	Ç	160	á	192	└	224	α
1	☺	33	!	65	A	97	a	129	ü	161	í	193	┴	225	ß
2	☻	34	"	66	B	98	b	130	é	162	ó	194	┬	226	Γ
3	♥	35	#	67	C	99	c	131	â	163	ú	195	├	227	π
4	♦	36	$	68	D	100	d	132	ä	164	ñ	196	─	228	Σ
5	♣	37	%	69	E	101	e	133	à	165	Ñ	197	┼	229	σ
6	♠	38	&	70	F	102	f	134	å	166	ª	198	╞	230	µ
7	•	39	'	71	G	103	g	135	ç	167	º	199	╟	231	τ
8	◘	40	(	72	H	104	h	136	ê	168	¿	200	╚	232	Φ
9	○	41	)	73	I	105	i	137	ë	169	⌐	201	╔	233	Θ
10	◙	42	*	74	J	106	j	138	è	170	¬	202	╩	234	Ω
11	♂	43	+	75	K	107	k	139	ï	171	½	203	╦	235	δ
12	♀	44	,	76	L	108	l	140	î	172	¼	204	╠	236	∞
13	♪	45	-	77	M	109	m	141	ì	173	¡	205	═	237	φ
14	♫	46	.	78	N	110	n	142	Ä	174	«	206	╬	238	ε
15	☼	47	/	79	O	111	o	143	Å	175	»	207	╧	239	∩
16	►	48	0	80	P	112	p	144	É	176	░	208	╨	240	≡
17	◄	49	1	81	Q	113	q	145	æ	177	▒	209	╤	241	±
18	↕	50	2	82	R	114	r	146	Æ	178	▓	210	╥	242	≥
19	‼	51	3	83	S	115	s	147	ô	179	│	211	╙	243	≤
20	¶	52	4	84	T	116	t	148	ö	180	┤	212	╘	244	⌠
21	§	53	5	85	U	117	u	149	ò	181	╡	213	╒	245	⌡
22	▬	54	6	86	V	118	v	150	û	182	╢	214	╓	246	÷
23	↨	55	7	87	W	119	w	151	ù	183	╖	215	╫	247	≈
24	↑	56	8	88	X	120	x	152	ÿ	184	╕	216	╪	248	°
25	↓	57	9	89	Y	121	y	153	Ö	185	╣	217	┘	249	∙
26	→	58	:	90	Z	122	z	154	Ü	186	║	218	┌	250	·
27	←	59	;	91	[	123	{	155	¢	187	╗	219	█	251	√
28	∟	60	<	92	\	124	¦	156	£	188	╝	220	▄	252	ⁿ
29	↔	61	=	93	]	125	}	157	¥	189	╜	221	▌	253	²
30	▲	62	>	94	^	126	~	158	₧	190	╛	222	▐	254	■
31	▼	63	?	95	_	127	⌂	159	ƒ	191	┐	223	▀	255	

437 English

0		32		64	@	96	`	128	Ç	160	á	192	└	224	Ó
1	☺	33	!	65	A	97	a	129	ü	161	í	193	┴	225	ß
2	☻	34	"	66	B	98	b	130	é	162	ó	194	┬	226	Ô
3	♥	35	#	67	C	99	c	131	â	163	ú	195	├	227	Ò
4	♦	36	$	68	D	100	d	132	ä	164	ñ	196	─	228	õ
5	♣	37	%	69	E	101	e	133	à	165	Ñ	197	┼	229	Õ
6	♠	38	&	70	F	102	f	134	å	166	ª	198	ã	230	µ
7	•	39	'	71	G	103	g	135	ç	167	º	199	Ã	231	þ
8	◘	40	(	72	H	104	h	136	ê	168	¿	200	╚	232	Þ
9	○	41	)	73	I	105	i	137	ë	169	®	201	╔	233	Ú
10	◙	42	*	74	J	106	j	138	è	170	¬	202	╩	234	Û
11	♂	43	+	75	K	107	k	139	ï	171	½	203	╦	235	Ù
12	♀	44	,	76	L	108	l	140	î	172	¼	204	╠	236	ý
13	♪	45	-	77	M	109	m	141	ì	173	¡	205	═	237	Ý
14	♫	46	.	78	N	110	n	142	Ä	174	«	206	╬	238	¯
15	☼	47	/	79	O	111	o	143	Å	175	»	207	¤	239	´
16	►	48	0	80	P	112	p	144	É	176	░	208	ð	240	
17	◄	49	1	81	Q	113	q	145	æ	177	▒	209	Ð	241	±
18	↕	50	2	82	R	114	r	146	Æ	178	▓	210	Ê	242	‗
19	‼	51	3	83	S	115	s	147	ô	179	│	211	Ë	243	¾
20	¶	52	4	84	T	116	t	148	ö	180	┤	212	È	244	¶
21	§	53	5	85	U	117	u	149	ò	181	Á	213	ı	245	§
22	▬	54	6	86	V	118	v	150	û	182	Â	214	Í	246	÷
23	↨	55	7	87	W	119	w	151	ù	183	À	215	Î	247	¸
24	↑	56	8	88	X	120	x	152	ÿ	184	©	216	Ï	248	°
25	↓	57	9	89	Y	121	y	153	Ö	185	╣	217	┘	249	¨
26	→	58	:	90	Z	122	z	154	Ü	186	║	218	┌	250	·
27	←	59	;	91	[	123	{	155	ø	187	╗	219	█	251	¹
28	∟	60	<	92	\	124	¦	156	£	188	╝	220	▄	252	³
29	↔	61	=	93	]	125	}	157	Ø	189	¢	221	¦	253	²
30	▲	62	>	94	^	126	~	158	×	190	¥	222	Ì	254	■
31	▼	63	?	95	_	127	⌂	159	ƒ	191	┐	223	▀	255	

850 Multilingual (Latin I)

0		32		64	@	96	`	128	Ç	160	á	192	└	224	Ó
1	☺	33	!	65	A	97	a	129	ü	161	í	193	┴	225	ß
2	☻	34	"	66	B	98	b	130	é	162	ó	194	┬	226	Ô
3	♥	35	#	67	C	99	c	131	â	163	ú	195	├	227	Ń
4	♦	36	Ş	68	D	100	d	132	ä	164	Ą	196	─	228	ń
5	♣	37	⅋	69	E	101	e	133	ů	165	ą	197	┼	229	ň
6	♠	38	&	70	F	102	f	134	ć	166	Ž	198	Ă	230	Š
7	•	39	'	71	G	103	g	135	ç	167	ž	199	ă	231	š
8	◘	40	(	72	H	104	h	136	ł	168	Ę	200	╚	232	Ŕ
9	○	41	)	73	I	105	i	137	ë	169	ę	201	╔	233	Ú
10	◙	42	*	74	J	106	j	138	Ő	170		202	╩	234	ŕ
11	♂	43	+	75	K	107	k	139	ő	171	ź	203	╦	235	Ű
12	♀	44	,	76	L	108	l	140	î	172	Č	204	╠	236	ý
13	♪	45	-	77	M	109	m	141	Ź	173	ş	205	═	237	Ý
14	♫	46	.	78	N	110	n	142	Ä	174	«	206	╬	238	ţ
15	☼	47	/	79	O	111	o	143	Ć	175	»	207	¤	239	´
16	►	48	0	80	P	112	p	144	É	176	░	208	đ	240	˝
17	◄	49	1	81	Q	113	q	145	Ĺ	177	▒	209	Đ	241	˛
18	↕	50	2	82	R	114	r	146	í	178	▓	210	Ď	242	ˇ
19	‼	51	3	83	S	115	s	147	ô	179	│	211	Ë	243	˘
20	¶	52	4	84	T	116	t	148	ö	180	┤	212	ď	244	˙
21	§	53	5	85	U	117	u	149	Ľ	181	Á	213	Ň	245	§
22	▬	54	6	86	V	118	v	150	ľ	182	Â	214	Í	246	÷
23	↨	55	7	87	W	119	w	151	Ś	183	Ě	215	Î	247	¸
24	↑	56	8	88	X	120	x	152	ś	184	Ş	216	ě	248	°
25	↓	57	9	89	Y	121	y	153	Ö	185	╣	217	┘	249	¨
26	→	58	:	90	Z	122	z	154	Ü	186	║	218	┌	250	·
27	←	59	;	91	[	123	{	155	Ť	187	╗	219	█	251	ű
28	∟	60	<	92	\	124	¦	156	ť	188	╝	220	▄	252	Ř
29	↔	61	=	93	]	125	}	157	Ł	189	Ż	221	Ţ	253	ř
30	▲	62	>	94	^	126	~	158	×	190	ż	222	Ů	254	■
31	▼	63	?	95	_	127	⌂	159	č	191	┐	223	▀	255	

852 Slavic (Latin II)

0		32		64	@	96	`	128	Ç	160	á	192	└	224	α
1	☺	33	!	65	A	97	a	129	ü	161	í	193	┴	225	ß
2	☻	34	"	66	B	98	b	130	é	162	ó	194	┬	226	Γ
3	♥	35	#	67	C	99	c	131	â	163	ú	195	├	227	π
4	♦	36	$	68	D	100	d	132	ã	164	ñ	196	─	228	Σ
5	♣	37	%	69	E	101	e	133	à	165	Ñ	197	┼	229	σ
6	♠	38	&	70	F	102	f	134	Á	166	ª	198	╞	230	µ
7	•	39	'	71	G	103	g	135	ç	167	º	199	╟	231	τ
8	◘	40	(	72	H	104	h	136	ê	168	¿	200	╚	232	Φ
9	○	41	)	73	I	105	i	137	Ê	169	⌐	201	╔	233	Θ
10	◙	42	*	74	J	106	j	138	è	170	¬	202	╩	234	Ω
11	♂	43	+	75	K	107	k	139	Ì	171	½	203	╦	235	δ
12	♀	44	,	76	L	108	l	140	Ô	172	¼	204	╠	236	∞
13	♪	45	-	77	M	109	m	141	ì	173	¡	205	═	237	ø
14	♫	46	.	78	N	110	n	142	Ã	174	«	206	╬	238	ε
15	☼	47	/	79	O	111	o	143	Â	175	»	207	╧	239	∩
16	►	48	0	80	P	112	p	144	É	176	░	208	╨	240	≡
17	◄	49	1	81	Q	113	q	145	À	177	▒	209	╤	241	±
18	↕	50	2	82	R	114	r	146	È	178	▓	210	╥	242	≥
19	‼	51	3	83	S	115	s	147	ô	179	│	211	╙	243	≤
20	¶	52	4	84	T	116	t	148	õ	180	┤	212	╘	244	⌠
21	§	53	5	85	U	117	u	149	ò	181	╡	213	╒	245	⌡
22	▬	54	6	86	V	118	v	150	Ú	182	╢	214	╓	246	÷
23	↨	55	7	87	W	119	w	151	ù	183	╖	215	╫	247	≈
24	↑	56	8	88	X	120	x	152	Ì	184	╕	216	╪	248	°
25	↓	57	9	89	Y	121	y	153	Õ	185	╣	217	┘	249	∙
26	→	58	:	90	Z	122	z	154	Ü	186	║	218	┌	250	·
27	←	59	;	91	[	123	{	155	¢	187	╗	219	█	251	√
28	∟	60	<	92	\	124	¦	156	£	188	╝	220	▄	252	ⁿ
29	↔	61	=	93	]	125	}	157	Ù	189	╜	221	▌	253	²
30	▲	62	>	94	^	126	~	158	₧	190	╛	222	▐	254	■
31	▼	63	?	95	_	127	⌂	159	Ó	191	┐	223	▀	255	

860 Portugal

0		32		64	@	96	`	128	Ç	160	á	192	└	224	α
1	☺	33	!	65	A	97	a	129	ü	161	í	193	┴	225	ß
2	☻	34	"	66	B	98	b	130	é	162	ó	194	┬	226	Γ
3	♥	35	#	67	C	99	c	131	â	163	ú	195	├	227	π
4	♦	36	$	68	D	100	d	132	ä	164	ñ	196	─	228	Σ
5	♣	37	%	69	E	101	e	133	à	165	Ñ	197	┼	229	σ
6	♠	38	&	70	F	102	f	134	å	166	ª	198	╞	230	µ
7	•	39	'	71	G	103	g	135	ç	167	º	199	╟	231	τ
8	◘	40	(	72	H	104	h	136	ê	168	¿	200	╚	232	Φ
9	○	41	)	73	I	105	i	137	ë	169	⌐	201	╔	233	Θ
10	◙	42	*	74	J	106	j	138	è	170	¬	202	╩	234	Ω
11	♂	43	+	75	K	107	k	139	ï	171	½	203	╦	235	δ
12	♀	44	,	76	L	108	l	140	î	172	¼	204	╠	236	∞
13	♪	45	-	77	M	109	m	141	ì	173	¾	205	═	237	φ
14	♫	46	.	78	N	110	n	142	Ä	174	«	206	╬	238	ε
15	☼	47	/	79	O	111	o	143	Å	175	»	207	╧	239	∩
16	►	48	0	80	P	112	p	144	É	176	░	208	╨	240	≡
17	◄	49	1	81	Q	113	q	145	æ	177	▒	209	╤	241	±
18	↕	50	2	82	R	114	r	146	Æ	178	▓	210	╥	242	≥
19	‼	51	3	83	S	115	s	147	ô	179	│	211	╙	243	≤
20	¶	52	4	84	T	116	t	148	ö	180	┤	212	╘	244	⌠
21	§	53	5	85	U	117	u	149	ò	181	╡	213	╒	245	⌡
22	▬	54	6	86	V	118	v	150	û	182	╢	214	╓	246	÷
23	↨	55	7	87	W	119	w	151	ù	183	╖	215	╫	247	≈
24	↑	56	8	88	X	120	x	152	ÿ	184	╕	216	╪	248	°
25	↓	57	9	89	Y	121	y	153	Ö	185	╣	217	┘	249	∙
26	→	58	:	90	Z	122	z	154	Ü	186	║	218	┌	250	·
27	←	59	;	91	[	123	{	155	¢	187	╗	219	█	251	√
28	∟	60	<	92	\	124	¦	156	£	188	╝	220	▄	252	n
29	↔	61	=	93	]	125	}	157	¥	189	╜	221	▌	253	²
30	▲	62	>	94	^	126	~	158	₧	190	╛	222	▐	254	■
31	▼	63	?	95	_	127	⌂	159	ƒ	191	┐	223	▀	255	

863 Canadian-French

0		32		64	@	96	`	128	Ç	160	á	192	└	224	α
1	☺	33	!	65	A	97	a	129	ü	161	í	193	┴	225	ß
2	☻	34	"	66	B	98	b	130	é	162	ó	194	┬	226	Γ
3	♥	35	#	67	C	99	c	131	â	163	ú	195	├	227	π
4	♦	36	$	68	D	100	d	132	ä	164	ñ	196	─	228	Σ
5	♣	37	%	69	E	101	e	133	à	165	Ñ	197	┼	229	σ
6	♠	38	&	70	F	102	f	134	å	166	ª	198	╞	230	µ
7	•	39	'	71	G	103	g	135	ç	167	º	199	╟	231	τ
8	◘	40	(	72	H	104	h	136	ê	168	¿	200	╚	232	Φ
9	○	41	)	73	I	105	i	137	ë	169	⌐	201	╔	233	Θ
10	◙	42	*	74	J	106	j	138	è	170	¬	202	╩	234	Ω
11	♂	43	+	75	K	107	k	139	ï	171	½	203	╦	235	δ
12	♀	44	,	76	L	108	l	140	î	172	¼	204	╠	236	∞
13	♪	45	-	77	M	109	m	141	ì	173	¡	205	═	237	ø
14	♫	46	.	78	N	110	n	142	Ä	174	«	206	╬	238	ε
15	☼	47	/	79	O	111	o	143	Å	175	»	207	╧	239	∩
16	►	48	0	80	P	112	p	144	É	176	░	208	╨	240	≡
17	◄	49	1	81	Q	113	q	145	æ	177	▒	209	╤	241	±
18	↕	50	2	82	R	114	r	146	Æ	178	▓	210	╥	242	≥
19	‼	51	3	83	S	115	s	147	ô	179	│	211	╙	243	≤
20	¶	52	4	84	T	116	t	148	ö	180	┤	212	╘	244	⌠
21	§	53	5	85	U	117	u	149	ò	181	╡	213	╒	245	⌡
22	▬	54	6	86	V	118	v	150	û	182	╢	214	╓	246	÷
23	↨	55	7	87	W	119	w	151	ù	183	╖	215	╫	247	≈
24	↑	56	8	88	X	120	x	152	ÿ	184	╕	216	╪	248	°
25	↓	57	9	89	Y	121	y	153	Ö	185	╣	217	┘	249	∙
26	→	58	:	90	Z	122	z	154	Ü	186	║	218	┌	250	·
27	←	59	;	91	[	123	{	155	ø	187	╗	219	█	251	√
28	∟	60	<	92	\	124	¦	156	£	188	╝	220	▄	252	n
29	↔	61	=	93	]	125	}	157	Ø	189	╜	221	▌	253	2
30	▲	62	>	94	^	126	~	158	₧	190	╛	222	▐	254	■
31	▼	63	?	95	_	127	⌂	159	ƒ	191	┐	223	▀	255	

865 Nordic

Index

A

A (append) command, Edlin, 463
A (assemble) command, Debug, 401 to 403
A20 handler, 611
Active partition
  described, 149
  setting, 157
Active Task List, MS-DOS Shell
  adding programs to, 57
  described, 31
  quitting programs, 58
  switching between programs, 56 to 57
Adding files with Replace, 123, 552 to 554
Advanced properties, MS-DOS Shell, 198
All Files command, MS-DOS Shell, 42
Allocation units
  defined, 127
  recovering, 294 to 295
Alphabetizing data in files, 568 to 570
ANSI characters not on keyboard, 631
ANSI commands. *See* ANSI escape sequences
ANSI escape sequences
  ASCII codes for, described, 262
  defined, 593
  described, 262
  driver support for, 593
  for assigning commands to keys, 266
  for background colors, 271, 595
  for creating screen displays, 268
  for cursor movement, 267, 594
  for display mode, 596
  for foreground colors, 271, 595
  for line wrap, 596
  for redefining keys, 265 to 266, 596
  for repositioning the command prompt, 268
  for screen attributes, 269 to 272, 595
  parameters used in, 593
  running from files, 264
  running from the command prompt, 263
  using in the command prompt, 546
ANSI.SYS device driver
  described, 261
  Device command syntax for, 593
  required for setting display mode, 527
Append command
  See also Path command
  restriction with reassigned drives, 365
  syntax and explanation, 363 to 365
APPEND environment variable, 364
Appending directories, 363 to 365
Appending files by using Copy, 80, 389 to 393
Application shortcut keys, defining, 188, 196
Applications
  See also Programs
  data files created by, 71
  .EXE files for, described, 71
Archive file attribute
  See also Backup, Restore, Xcopy commands
  backing up files, 370
  copying files by using Xcopy, 369, 589
  purpose of, 94
  restoring files, 555
  viewing or setting, 368
Arithmetic, hexadecimal, 411
Arrow keys, 7
ASCII codes
  assigning to keys, 265 to 266
  described, 262
ASCII files
  comparing, 488 to 491
  copying, 390
  creating. *See* MS-DOS Editor
  described, 71
  viewing, 75
Assign command
  See also Subst command
  restrictions with other commands, 366
  syntax and explanation, 366 to 367
  using with the Mirror command, 523
Assigned drives. *See* Reassigned drives
Assigning a path to a virtual drive, 571
Associate command, MS-DOS Shell, 58 to 60
Asterisk (*) wildcard, 73 to 75
Attrib command
  introduction to using, 94 to 95
  syntax and explanation, 368 to 369
Attributes. *See* File attributes
AUTOEXEC.BAT file
  changing, 286
  creating a startup procedure, 246 to 249
  defining macros in, 177
  described, 245
  freeing conventional memory, 286

AUTOEXEC.BAT file (*continued*)
loading programs
Fastopen, 487
into the upper memory area, 519
Keyb, 515
Mirror, 88
Nlsfunc, 345
presetting Dir command options, 439 to 440
running programs from, 247
saving disk information, 524, 582
setting
a date prompt, 398
a search path, 124, 246, 248
a time prompt, 576
environment variables, 559
the command prompt, 107, 248
starting MS-DOS Shell, 247
viewing contents of, 76
AUX port, using for command input and output, 397

B

Background colors, escape sequences for, 271, 595
Backing up files
See also Backup command
adding files to a backup disk, 139
described, 136
in a directory and its subdirectories, 138
in one directory, 137
using MS-DOS Shell, 140
using wildcards, 139
viewing filenames on a backup disk, 144
Backup command
See also Restore command
affected by country code, 394
introduction to using, 136 to 140
restriction with reassigned drives, 372
restriction with system files, 372
syntax and explanation, 370 to 374
versions compatible with Restore command, 556
Backup Fixed Disk command, MS-DOS Shell, 140
Bad sectors. *See* Defective sectors
.BAK files, Edlin, 467
Basic language interpreter, 547
.BAT files. *See* Batch programs
Batch commands
defined, 358
described, 226
listed, 359
Batch parameters with the Call command, 377
Batch programs
adding blank lines to, 550
avoiding endless loop with the Call command, 377
Batch programs (*continued*)
batch commands, described, 226
calling environment variables from, 560
calling from another batch program, 233, 377
colon for labels, 236
creating a menu system, 238 to 244
creating conditions in, 235 to 237, 242 to 244
creating
by using Doskey, 174
by using the Copy command, 228 to 229
defining macros in, 176
described, 71, 225
displaying commands and messages, 228 to 230
dividing into sections, 541
including comments in, 232, 550
naming conventions for, 227
processing exit codes by using errorlevel, 373
repositioning replaceable parameters, 567
run after .COM and .EXE files, 540
running, 227
running a command for a set of files, 496
running ANSI escape sequences, 264
running in a startup command, 192
stopping
temporarily, 227, 231, 541
to discontinue processing, 227, 541
to display a prompt, 231
testing, 228
tools for creating, 226
using more than 10 parameters in, 567
using replaceable parameters in, 234 to 235, 242
using Set to create variables in, 560
using the Goto command, 236 to 237, 242 to 244
versus macros, 175 to 176
Baud rate for serial ports, 257, 530
.BIN files, converting .EXE files to, 484
Binary files
comparing, 488 to 491
converting executable files to, 483 to 484
copying, 390
viewing, 578
Black and white
starting MS-DOS Editor in, 459
starting MS-DOS Shell in, 455
Blank lines, in batch programs, 550
Block devices, defining parameters for, 455 to 456
Bold type in command syntax, defined, xiii
Brackets in command syntax, defined, 361
Break command
in the CONFIG.SYS file, 250
introduction to using, 253
syntax and explanation, 374

Breakpoints, specifying in Debug, 410
Buffer cache. *See* Secondary buffer cache
Buffer, MS-DOS Editor
  described, 214
  transferring text to and from, 214 to 215
Buffers. *See* Disk buffers
Buffers command
  freeing conventional memory, 284
  in the CONFIG.SYS file, 250
  introduction to using, 251
  syntax and explanation, 375
  using to increase system speed, 300
Buttons
  in MS-DOS Editor, 208
  in MS-DOS Shell, 35 to 36
Bytes
  available, 382
  defined, 126

C

C (compare) command, Debug, 403
C (copy) command, Edlin, 464
Cache
  disk cache. *See* SMARTDrive cache
  secondary buffer cache, 301, 376
Calculations, performing in Debug, 411
Call command
  in startup commands for program items, 192
  introduction to using, 233
  syntax and explanation, 377
Calls, overriding with Debug, 402
Canceling a command, 23
Capitalization of commands, 20
Cd command. *See* Chdir command
Central processing unit, described, 4
CGA. *See* Color graphics adapters
Change Attributes command, MS-DOS Shell, 96
Change command, MS-DOS Editor, 216
Change-line support, specifying
  for a logical drive, 603
  for a physical drive, 457
Character sets
  See also Code pages
  described, 334
  graphics mode for extended characters, 504
  provided by MS-DOS, 632
Chcp command
  See also Country, Nlsfunc, Device, Mode commands
  introduction to using, 347 to 349
  syntax and explanation, 378
Chdir (cd) command
  introduction to using, 115 to 116
  syntax and explanation, 379 to 381
  versus Path command, 124
Check boxes
  in MS-DOS Editor, 208
  in MS-DOS Shell, 35, 39
Checking disks
  for defective sectors, 383
  for logical errors, 381 to 383
Child directory, defined, 102
.CHK files, 295
Chkdsk command
  See also Recover command
  bad sectors not affected by Restore, 549
  increasing disk space, 294 to 295
  keeping track of free disk space, 278
  restrictions on using
    with networks, 382
    with open files, 382
    with reassigned drives, 382
  syntax and explanation, 381 to 383
  using before formatting a hard disk, 160, 299
  viewing information about floppy disks, 126
Choosing
  buttons
    in MS-DOS Editor, 208
    in MS-DOS Shell, 36
  commands
    in MS-DOS Editor, 206
    in MS-DOS Shell, 33 to 34
Clear command, MS-DOS Editor, 215
Clearing
  check boxes
    in MS-DOS Editor, 208
    in MS-DOS Shell, 39
  the MS-DOS Editor window, 215
  the screen, 383
Cls command
  syntax and explanation, 383
  using in customized menus, 239, 244
Code pages
  changing for all devices, 347 to 348, 378
  changing for one device, 348
  country codes for, 394
  DISPLAY.SYS device driver, 341, 527, 600
  hardware
    defined, 334
    specifying in the Device command, 342
  loading, 346 to 347

Code pages (*continued*)
loading national language support for, 345
prepared
and monochrome or CGA monitors, 339
defined, 334
provided by MS-DOS, 339, 632
setting up, described, 340
preparing devices for code pages
keyboard and screen, 341
printers, 343 to 345
PRINTER.SYS device driver, 343, 613 to 614
reinstating after losing, 534
sample language changes, 351 to 353
specifying
in the Country command, 336, 393
in the Graftabl command, 504
in the Keyb command, 338
viewing information about, 349 to 350
Code-page-information (.CPI) files, 334, 534
Collapse Branch command, MS-DOS Shell, 46
Collating sequence table, 569
Colon (:) in batch programs, 236
Color graphics adapters
configuring, 535
using with DISPLAY.SYS, 601
Color monitors, described, 4
Colors
ANSI escape sequences for, 271, 595
changing
in MS-DOS Editor, 221
in MS-DOS Shell, 183 to 184
Colors command, MS-DOS Shell, 183
.COM files
converting .EXE files to, 484
described, 71
run before .EXE and .BAT files, 540
COM ports, configuring, 257, 529
Combining files, 80, 389 to 393
Command buttons in MS-DOS Shell, 36
Command command
See also Shell command
syntax and explanation, 384 to 386
Command environment. *See* Environment
Command interpreter
copying, 574
defined, 384
displaying complete error messages, 386
increasing the environment size, 386, 567
quitting, 485
running multiple command environments, 385
setting the version number, 562
specifying, 566
Command line. *See* MS-DOS command line
Command prompt
changing the appearance of, 106 to 107
defined, 19
list of variations of, 545
repositioning, 268
switching to from MS-DOS Shell, 30
using ANSI escape sequences in, 546
COMMAND.COM file
and programs loaded with Install, 510
and the primary DOS partition, 147
copying when formatting a disk, 133, 500
described, 24
quitting the command interpreter, 485
restriction on backing up, 372
starting a new command environment, 384
Commands
ANSI, 262
assigning to keys, 266
batch commands
defined, 358
described, 226
listed, 359
canceling, 23
carrying out, 14, 19
charts in command descriptions, 357
choosing
in MS-DOS Editor, 206
in MS-DOS Shell, 33 to 34
combining with redirection characters, 165
CONFIG.SYS commands
defined, 358
described, 250
listed, 360
Debug commands, listed, 360
defined, 19
displaying when a batch program runs, 229 to 230
editing
using Doskey, 21, 169 to 174
using editing keys, 21, 166 to 168
Edlin commands, listed, 360
external, defined, 24, 358
filter, described, 163
indicating the disk drive, 23
internal, defined, 24, 358
interpreting MS-DOS response to, 22 to 23
MS-DOS commands
defined, 357
listed, 358
requesting online Help for, 24, 362
MS-DOS Shell, online Help for, 62
name, defined, 19
network, defined, 358
parts of, described, 19 to 20

Commands (*continued*)
redirecting input from a file, 163
redirecting output to a file or printer, 162
startup commands
for program items, 191 to 195
for the AUTOEXEC.BAT file, 246 to 249
stopping and restarting, 23
syntax conventions defined, 360 to 362
types of, described, 357
typing and correcting mistakes, 20 to 21
typing multiple commands per line, 170
viewing or repeating
using Doskey, 170 to 172, 449
using editing keys, 167
Comments, including in batch programs, 232, 550
Comp command
See also Diskcomp and Fc commands
syntax and explanation, 387 to 389
Compacting your hard disk, 297 to 298
Comparing
files
using the Comp command, 387 to 389
using the Fc command, 93, 488 to 491
floppy disks, using the Diskcomp command, 441 to 443
portions of memory, 403
Compressed files, expanding, 485
Computer memory. *See* Memory
COMSPEC environment variable, 384, 560
Conditional processing, 235 to 237, 242 to 244, 509
CONFIG.SYS commands
defined, 358
described, 250
listed, 360
CONFIG.SYS file
ANSI.SYS required for setting display mode, 527
assigning two letters to one drive, 261
changing, 249, 286
changing the default country setting, 393
commands for system configuration, 250
configuring ports, 255 to 258
described, 245
DISPLAY.SYS for code-page switching, 527
enabling CTRL+C checking, 253, 374
freeing conventional memory, 284
including comments in, 550
increasing the environment size, 386, 567
installing
ANSI.SYS, 593
DISPLAY.SYS, 341, 600
DRIVER.SYS, 259, 602
EGA.SYS, 605
EMM386.EXE, 290, 605
HIMEM.SYS, 281, 610
CONFIG.SYS file (*continued*)
installing (*continued*)
PRINTER.SYS, 343, 613
RAMDRIVE.SYS, 310, 614
SETVER.EXE, 617
SMARTDRV.SYS, 306, 617
the Fastopen program, 303, 487
the Keyb program, 339, 515
the Nlsfunc program, 345, 538
the Share program, 565 to 566
loading device drivers, 251, 433, 435
loading memory-resident programs, 510, 518
reconfiguring devices with Drivparm, 457
samples of, 254 to 255
setting the last drive letter, 253, 517
setting the number and size of stacks, 570
setting the number of buffers, 252
setting the number of open files, 252, 493
setting up for the upper memory area, 317
specifying a command interpreter, 566
specifying a file for sort sequence, 569
switching keyboard functions, 573
using Fcbs for older programs, 492
Configuring
a hard disk. *See* Fdisk program
MS-DOS for your system, 249 to 255
printers, 255, 527
serial ports, 257, 529
your system. *See* Customizing your system
Confirmation command, MS-DOS Shell, 60
Conventional memory
and disk buffers, 376
described, 275
freeing for use by programs
described, 281
running MS-DOS in extended memory, 282
streamlining your AUTOEXEC.BAT file, 286
streamlining your CONFIG.SYS file, 284
using the upper memory area, 313 to 316
specifying in MS-DOS Shell for programs, 199
using for a RAM disk, 615
viewing the status of, 519
Conventions used in this guide
for command syntax, 360 to 362
for key combinations, xiii
for types of commands, 357
typographic, xiii
Copy command
See also Sys command, Xcopy command
accidentally copying over a file, 77
creating small batch programs, 228 to 229
introduction to using, 77 to 81
MS-DOS Editor, 215

Copy command (*continued*)
MS-DOS Shell, 82, 118, 188
syntax and explanation, 389 to 393
using with Del to move files, 92
versus Diskcopy command, 446
versus Xcopy command, 118
Copying
directories, 118 to 120, 587 to 590
directories, by using MS-DOS Shell, 118
files, 77 to 81, 389 to 393
files, by using MS-DOS Shell, 82
floppy disks, 444 to 446
from the keyboard to a file, 80
from the keyboard to a printer, 81
IO.SYS, MSDOS.SYS system files, 574 to 575
lines, using Edlin, 464
memory, by using Debug, 415
program items among program groups, 188
subdirectories, 119
text, using MS-DOS Editor, 215
Country (or language)
See also Customizing for international use
changing without changing code pages, 350
configuring a keyboard for, 337, 513 to 516
loading country-specific information, 538
sample language changes, 350 to 353
supported by MS-DOS version 5.0, 333, 394
Country command
See also Keyb, Mode, Nlsfunc commands
changing date and time formats, 335
effects on other commands, 394
in the CONFIG.SYS file, 250
introduction to using, 334
specifying a file for sort sequence, 569
syntax and explanation, 393 to 396
COUNTRY.SYS file
collating sequence used by Sort, 569
default file for country information, 393, 539
specifying the path for, 336
.CPI files, 334, 534
CPU. *See* Central processing unit
Create Directory command, MS-DOS Shell, 114
CTRL+BREAK key combination
canceling a command, 23
stopping a batch program, 227
stopping the printer retry option, 256
CTRL+C checking, setting or clearing, 253, 374
CTRL+C key combination
breaking a time-out loop, 529
canceling a command, 23
skipping macro commands, 175
stopping a batch program, 227
stopping a macro, 175
CTRL+S key combination
stopping a batch program temporarily, 227
stopping a command temporarily, 23
Ctty command
See also Command and Mode commands
syntax and explanation, 396
Current directory
appending directories to, 363 to 365
changing
on another drive, 380
using MS-DOS Shell, 44
using the Chdir command, 115 to 116
described, 105 to 106
removing, 558
represented by a period (.), 103
using from another drive, 380
using in the command prompt, 106 to 107
Current drive
changing, 24, 104
defined, 23
in MS-DOS Shell, 43
using in the command prompt, 546
Cursor movement
ANSI escape sequences for, 594
in MS-DOS Editor, 212 to 213
repositioning the cursor, 267
Cursor, on the MS-DOS command line, 20
Customizing for international use
See also Code pages
changing code pages, 347 to 348
changing date and time formats, 335
described, 334
loading code pages, 346 to 347
loading the Nlsfunc program, 345
preparing devices for code pages
keyboard and screen, 341
printers, 343 to 345
rearranging the keyboard keys, 337 to 339
sample language changes, 350 to 353
supported countries and languages, 333
viewing code page information, 349 to 350
Customizing MS-DOS Editor, 221
Customizing MS-DOS Shell
adding passwords for program items, 198
adding program groups, 184 to 186
adding program items to groups, 186 to 188
changing
color schemes, 183 to 184
group properties, 201
program item properties, 190
screen modes, 184
copying program items among groups, 188

Customizing MS-DOS Shell (*continued*)
creating Help messages
for program groups, 185
for program items, 198
creating startup commands, 191
defining application shortcut keys, 196
deleting a program group, 186
deleting program items from groups, 189
pausing after quitting a program, 197
preventing program switching, 201
rearranging items in program groups, 189
reserving shortcut keys for programs, 200
running batch programs in a startup command, 192
setting video mode for programs, 200
specifying memory for programs, 199
using a startup directory for programs, 195
using replaceable parameters, 192 to 195
Customizing your system
adding floppy disk drives, 258 to 261
assigning commands to keys, 266
changing screen attributes, 269 to 272
configuring MS-DOS, 249 to 255
configuring ports and printers, 255 to 258
creating a startup procedure, 246 to 249
creating menus by using batch programs, 238 to 244
creating screen displays, 268
described, 245
increasing the environment size, 386, 567
redefining keys, 265 to 266
repositioning the command prompt, 268
repositioning the cursor, 267
using ANSI.SYS, 261
using batch-program messages, 229
Cut command, MS-DOS Editor, 214

D

D (delete) command, Edlin, 466
D (dump) command, Debug, 405
Data bits for serial ports, 257, 530
Data files, described, 71
Data stacks, specifying size and number of, 570
Date
changing or viewing, 398
displaying in the command prompt, 546
format
changing by using the Country command, 335
table of country codes for, 395
of file creation, viewing, 72
Date command
affected by country code, 394
in the AUTOEXEC.BAT file, 246 to 247
syntax and explanation, 398
Deallocating handles to expanded memory, 429
Debug commands
A (assemble), 401 to 403
C (compare), 403
D (dump), 405
E (enter), 406 to 408
F (fill), 408
G (go), 409 to 411
H (hex), 411
I (input), 412
L (load), 413 to 415
M (move), 415
N (name), 416 to 418
O (output), 418
P (proceed), 419
Q (quit), 420
R (register), 421 to 423
S (search), 423
T (trace), 424
U (unassemble), 425 to 426
W (write), 427 to 428
XA (allocate expanded memory), 428
XD (deallocate expanded memory), 429
XM (map expanded-memory pages), 430
XS (display expanded-memory status), 431
Debug program
address parameter, defined, 401
allocating expanded-memory pages, 428
assembling executable machine code, 401 to 403
assigning a filename and parameters, 416 to 418
breakpoints, restrictions on setting, 410
comparing portions of memory, 403
copying a block of memory, 415
deallocating handles to expanded memory, 429
disassembling bytes, 425 to 426
entering data into memory, 406 to 408
executing instructions, 419
loading a file or sectors into memory, 413 to 415
mapping expanded-memory pages, 430
mnemonics, 402
operation codes, 401, 410
performing hexadecimal arithmetic, 411
pseudoinstructions, 402
quitting, 420
range parameter, defined, 401
running the program currently in memory, 409 to 411
searching for byte values, 423
sending a byte value to an output port, 418
separating command parameters, 400
starting, 399
tracing an instruction, 424

Debug program (*continued*)
viewing
a byte at a port address, 412
expanded-memory status, 431
memory contents, 405
register and flag contents, 421 to 423
word and byte memory locations, 402
writing a file to disk, 427 to 428
Debug prompt, 399
Decoded instructions, viewing, 424
Default drive. *See* Current drive
Defective sectors
described, 127
logical versus physical disk errors, 383
recovering data from a defective disk, 145, 548 to 549
reported by the Chkdsk command, 383
reported by the Format command, 130
Defragmentation utilities, 297
Del command
See also Mirror, Rmdir, Undelete commands
introduction to using, 88 to 89
restoring files deleted with, 90 to 91, 579 to 581
syntax and explanation, 432
using to rename a directory, 120
using with Copy to move files, 92
Delete command, MS-DOS Shell
deleting a directory, 117
deleting a program group, 186
deleting a program item, 189
deleting files, 89
Deleting
directories, 116, 558 to 559
directories, by using MS-DOS Shell, 117
files, 86 to 89, 432
files, by using MS-DOS Shell, 89
lines, by using Edlin, 466
logical drives, 157 to 158
macros, 178
program groups, 186
program items from groups, 189
text, by using MS-DOS Editor, 215
virtual drives, 146, 572
Deletion tracking
recovering deleted files, 90, 579 to 581
removing from memory, 523 to 524
setting up, 86, 522 to 525
Deselect All command, MS-DOS Shell, 53
Destination, defined, 362
Device command
See also Device drivers
in the CONFIG.SYS file, 250
introduction to using, 251
syntax and explanation, 433
Device drivers
ANSI.SYS, 261, 593 to 597
described, 245, 591
DISPLAY.SYS, 341, 600
DRIVER.SYS, 258 to 261, 602 to 604
EGA.SYS, 605
EMM386.EXE, 289 to 290, 605 to 609
HIMEM.SYS, 279 to 281, 610 to 613
installable, 245, 434
loading into memory, 251, 433
order in the CONFIG.SYS file, 285
PRINTER.SYS, 343, 613 to 614
RAMDRIVE.SYS, 309 to 313, 614 to 616
running in the upper memory area, 323 to 325, 435
SETVER.EXE, 617
SMARTDRV.SYS, 304 to 308, 617 to 619
specifying amount of memory for, 436
trouble loading into a UMB, 329
Devicehigh command
See also Dos and Mem commands
freeing conventional memory, 323 to 325
in the CONFIG.SYS file, 250
syntax and explanation, 435
Devices
changing for command input and output, 385, 396
copying files to and from, 390
.CPI files for, 535
defined, 245
preparing and selecting code pages for, 533
viewing the status of, 531
Dialog boxes, described
in MS-DOS Editor, 207
in MS-DOS Shell, 34 to 40
Diamond, in MS-DOS Shell menus, 33
Dir command
See also Set command, Tree command
comparing and tracking files, 72
introduction to using, 107 to 111
keeping track of free disk space, 278
presetting options for, 439 to 440
redirecting output to the More command, 537
syntax and explanation, 437 to 441
using with appended directories, 364
versus Tree command, 111
DIRCMD environment variable, 439 to 440, 560
Direct memory access, 608
Directories
adding files by using Replace, 123, 552 to 554
appending, 363 to 365
arranging a directory display, 109
backing up, 136 to 140, 370 to 374
copying, 118 to 120, 587 to 590
creating, 113 to 114, 525

Directories (*continued*)
creating as you copy files, 119
current
changing, 115 to 116, 379 to 381
changing on another drive, 380
described, 105 to 106
displaying in the command prompt, 106 to 107
represented by a period (.), 103
using from another drive, 380
deleting all files from, 89
described, 16, 101
naming conventions for, 102
organizing files with, 101
parent, 102 to 103
path for, described, 103 to 104
period (.) and double period (..), 103
recovering from a defective disk, 145, 548 to 549
removing, 116 to 117, 558 to 559
renaming, 120 to 121
restoring, 141 to 144, 555 to 557
root
changing to, 380
defined, 101, 134
size limitations, 102
sorting a directory display, 110
specifying in a search path, 123, 539
startup, for program items, 188, 195
updating, 122
viewing
a group of related filenames, 109
information about, 100
list of filenames in, 108, 437 to 441
one screen at a time, 109
the name of, 379 to 381
using MS-DOS Shell, 41, 43 to 46
using the Tree command, 111, 577
Directory tree, 29, 41, 101 to 102
Disk buffers
default setting for, 376
defined, 251
freeing conventional memory, 284
how MS-DOS uses, 376
specifying the number of, 252, 375
speeding up your system, 300
Disk cache. *See* SMARTDrive cache
Disk capacity, specifying, 130
Disk drives
3.5-inch, support for, 456
adding floppy disk drives, 258 to 261
assigning a physical drive number, 455
assigning two letters to one drive, 261
change-line support, 457, 603
Disk drives (*continued*)
current
changing, 24, 104
changing in MS-DOS Shell, 43
defined, 23
defining parameters
for a logical drive, 602
for a physical drive, 455 to 456
described, 10, 127
indicating on the command line, 23
inserting and removing disks, 12
joining to a directory on another drive, 511 to 513
logical, 155 to 160
logical versus physical, 258
maximum number accessible, 517
reassigned. *See* Reassigned drives
redirecting disk operations with Assign, 366
substituting a path for a drive letter, 146
virtual, 571
Disk errors
See also Recover command
finding with Chkdsk, 381 to 383
fixing, 381 to 383
logical versus physical, 383
Disk organizers, 297
Disk space
and file size, 72
increasing
by deleting unnecessary files, 292
by recovering allocation units, 294 to 295
by using a RAM disk, 309 to 313
keeping track of, 278
on a newly formatted disk, 130
viewing a status report of, 382
Disk Utilities group, MS-DOS Shell, 30
Disk volume label. *See* Volume label
Disk-caching programs
See also SMARTDRV.SYS device driver
described, 304
versus secondary buffer cache, 301
Disk-compaction programs
restriction with SMARTDRV.SYS, 618
restriction with the Fastopen program, 487
using to speed up your system, 297 to 298
Diskcomp command
See also Comp and Fc commands
restriction with networks, 443
restriction with reassigned drives, 443
syntax and explanation, 441 to 443
Diskcopy command
See also Copy command, Xcopy command
restriction with reassigned drives, 366, 572

Diskcopy command (*continued*)
syntax and explanation, 444 to 446
versus Xcopy command, 589
Disks
See also Floppy disks, Hard disks
backing up, 136 to 140, 370 to 374
checking, 381 to 382
comparing floppy disks track by track, 441 to 443
copying, 444 to 446
described, 125 to 126
formatting
creating a system disk, 133 to 134
described, 127 to 128
preparing for MS-DOS files, 129 to 130, 498 to 502
specifying disk capacity, 130
using MS-DOS Shell, 131 to 132
when backing up files, 370
when copying a floppy disk, 444 to 446
fragmented, 296, 446
labeling backup disks, 372
recovering data from a defective disk, 145, 548 to 549
renaming, 134
restoring, 132, 581 to 585
saving information about, 522 to 525, 582
verifying a write, 586
viewing a status report of, 381 to 383
viewing information about, 100
viewing the directory structure of, 111, 577
volume label for, 129, 134 to 135, 516
volume serial number created by Diskcopy, 445
Display adapters
configuring, 535
supported by DISPLAY.SYS, 600
Display command
MS-DOS Editor, 221
MS-DOS Shell, 184
Display mode
See also Graphics mode, Text mode
ANSI escape sequences for, 595 to 596
ANSI.SYS requirement, 527
selecting, 535
DISPLAY.SYS device driver
Device command syntax for, 600
preparing your console for code pages, 341 to 343
required for code-page switching, 527
Displaying. *See* Viewing
DMA. *See* Direct memory access
Dos command
See also Devicehigh and Loadhigh commands
freeing conventional memory, 283 to 284
in the CONFIG.SYS file, 250
linking to the upper memory area, 318
required for Devicehigh, 436
Dos command (*continued*)
required for Loadhigh, 519
running MS-DOS in conventional memory, 288
syntax and explanation, 447 to 448
DOS partitions. *See* Fdisk program
Doskey command
See also Doskey program
in the AUTOEXEC.BAT file, 248
syntax and explanation, 448 to 449, 451 to 453
Doskey program
creating macros, 176
deleting macros, 178
deleting stored commands, 174
described, 169
editing ANSI escape sequences, 263
editing macros, 178
editing previous commands, 173 to 174
installing, 169
macros, described, 175 to 176
redirection characters for macros, 180
running macros, 177
saving commands in a batch program, 174
saving macros, 178
starting, 21
typing multiple commands per line, 170
using replaceable parameters in macros, 179 to 180
viewing previous commands, 170 to 172
Dosshell command
in the AUTOEXEC.BAT file, 247
starting MS-DOS Shell, 27
syntax, 454
Double period (..) for parent directory, 103
Drive icons in the MS-DOS Shell window, 29
Drive letters
and hard disk partitions, described, 156 to 160
assigning two letters to one drive, 261
described, 10
setting the maximum number of, 517
substituting with a path, 146, 571
Drive number and type, specifying, 455
Drive, in command syntax, defined, 361
DRIVER.SYS device driver
See also Drivparm command, Subst command
assigning two letters to one drive, 261
described, 258
Device command syntax for, 602 to 604
installing, 259 to 260
restriction on using with hard disks, 604
Drivparm command
in the CONFIG.SYS file, 250
syntax and explanation, 455 to 456
Dual File Lists command, MS-DOS Shell, 41

E

E (end) command, Edlin, 467
E (enter) command, Debug, 406 to 408
Echo command
  for ANSI escape sequences, 264
  in the AUTOEXEC.BAT file, 247 to 248
  introduction to using, 229 to 230
  restriction on using in macros, 176
  syntax and explanation, 457
Echoing. *See* Echo command
Edit command
  See also MS-DOS Editor
  starting MS-DOS Editor, 204
  syntax and explanation, 459
EDIT.HLP file, 211
Editing
  commands
    using Doskey, 21, 169 to 174
    using editing keys, 21, 166 to 168
  files. *See* MS-DOS Editor or Edlin
  macros, 178
  your AUTOEXEC.BAT file, 286
  your CONFIG.SYS file, 249, 286
Editing keys
  for commands, 21, 166 to 168
  for Doskey, 173 to 174
  for MS-DOS Editor, 212
Editor. *See* MS-DOS Editor
Edlin
  backup (.BAK) files, 467
  copying lines, 464
  deleting lines, 466
  editing large files in stages, 463, 481
  freeing space in memory, 481
  inserting text, 467
  insufficient memory to load entire file, 463
  listing lines in a file, 469
  merging files, 480
  moving lines, 470
  paging through a file, 472
  quitting
    and saving changes, 467
    without saving changes, 473
  replacing text, 474 to 476
  searching for text, 478 to 479
  starting, 460
  viewing a specific line, 462
Edlin commands
  A (append), 463
  C (copy), 464
  D (delete), 466
  E (end), 467
Edlin commands (*continued*)
  I (insert) command, 467
  L (list), 469
  line, 462
  M (move), 470
  P (page), 472
  Q (quit), 473
  R (replace), 474 to 476
  S (search), 478 to 479
  T (transfer), 480
  W (write), 481
Edlin prompt, 461
EGA. *See* Enhanced graphics adapters
EGA.SYS device driver, 605
Ellipsis
  in command syntax, defined, 361
  in MS-DOS Shell menus, 33
EMB handles, 610
EMM386 command, 482
EMM386.EXE device driver
  Device command syntax for, 605 to 609
  emulating expanded memory, 289 to 290
  freeing extended memory, 287
  installing, 290
  managing the upper memory area, 318
  required for Devicehigh, 436
  required for Loadhigh, 519
EMS banking, 607
EMS memory. *See* Expanded memory
EMS pages, described, 276
Enable Task Swapper command, MS-DOS Shell, 56
End-of-file character
  adding to a copied file, 391
  copying up to, 391
Enhanced graphics adapters
  using with prepared code pages, 341
  using with Task Swapper, 605
Enhanced keyboards, 573
ENTER-LINEFEED in the command prompt, 546
Entering a command, 19
Environment
  increasing the size of, 386, 567
  insufficient space for environment variable, 560
  running multiple command interpreters, 385
  specifying the size of, 384
Environment variables
  and programs loaded with Install, 510
  APPEND, 364
  COMSPEC, 384, 560
  described, 559
  DIRCMD, 439 to 440, 560
  in batch programs, 235
  PATH, 560

Environment variables (*continued*)
presetting Dir command options, 439 to 440
PROMPT, 560
restriction on using in macros, 176
setting or viewing, 559 to 560
setting TEMP
before using a pipe, 163
to a RAM disk, 312, 616
TMP, 312
Erase command. *See* Del command
Error messages
displayed when a batch program runs, 228
displaying on a floppy disk system, 386
Errorlevel in batch programs, 373, 509
Escape sequences. *See* ANSI escape sequences
.EXE files
converting to binary format, 483 to 484
described, 71
loading into memory with Debug, 414
run after .COM files, 540
Exe2bin command, 483 to 484
Executable files
See also .EXE files, Programs
described, 71
search path for, 123
setting MS-DOS version for, 561 to 564
testing, 399
Exit codes, errorlevel processing with, 373, 509
Exit command
See also Command command
MS-DOS Editor, 211
MS-DOS Shell, 66
syntax and explanation, 485
Expand All command, MS-DOS Shell, 46
Expand Branch command, MS-DOS Shell, 46
Expand command, syntax and explanation, 485
Expand One Level command, MS-DOS Shell, 46
Expanded memory
allocating with Debug, 428
creating a SMARTDrive cache in, 308, 617
deallocating handles to, 429
described, 276
emulating, 289 to 290, 605 to 609
enabling or disabling, 482
freeing for use by programs, 288
mapping, 430
memory manager
required for RAMDRIVE.SYS, 616
required for SMARTDRV.SYS, 618
using for a RAM disk, 311, 616
viewing the status of, 431, 520
Expanded Memory Specification (LIM EMS), 276, 520
Expanding compressed files, 485
Extended characters
displaying in graphics mode, 504
producing, 631
Extended DOS partition
and drive letters, described, 156 to 160
creating, 154 to 155
creating or changing logical drives, 155
deleting logical drives from, 157 to 158
described, 147 to 148
Extended keys, remapping, 593
Extended memory
creating a SMARTDrive cache in, 308, 617
described, 275
emulating expanded memory, 289 to 290, 605 to 609
freeing for use by programs, 287
memory manager
installing HIMEM.SYS, 281, 610
required for RAMDRIVE.SYS, 615
required for SMARTDRV.SYS, 618
using HIMEM.SYS, 279 to 281
running MS-DOS in, 282
specifying in MS-DOS Shell for programs, 199
using for a RAM disk, 311, 615 to 616
viewing the status of, 520
Extended Memory Specification (XMS), 275
Extending selections in MS-DOS Shell, 50 to 52
External commands, defined, 24, 358

F

F (fill) command, Debug, 408
F1 key, for changing commands, 21, 168
F2 key, for changing commands, 168
F3 key, for changing commands, 21
Far prefix, using in Debug, 402
Fastopen command
See also Fastopen program
disabling to free conventional memory, 284
syntax and explanation, 486 to 487
Fastopen program
described, 302 to 304
recommendations for using, 303
restrictions on using
with disk-compaction programs, 487
with floppy disks, 487
with networks, 487
running in the upper memory area, 303
starting, 303
Fc command
introduction to using, 93
syntax and explanation, 488 to 491
versus Diskcopy command, 443

Fcbs command
freeing conventional memory, 285
in the CONFIG.SYS file, 250
syntax and explanation, 492
Fdisk command
See also Fdisk program
syntax and explanation, 492
Fdisk program
creating a primary DOS partition, 152 to 154
creating an extended DOS partition, 154 to 155
creating logical drives, 155
deleting a partition or logical drive, 157 to 158
described, 149
drive letters for partitions, described, 156 to 160
formatting partitions with Format, 159 to 160
menus, described, 150
partitioning more than one hard disk, 159
partitions, described, 147 to 149
restriction with reassigned drives, 493
restriction with Unformat, 583
running
after setting up MS-DOS, 150
while running the Setup program, 149
viewing partition data, 151 to 152
File allocation table
checking for errors, 381 to 383
described, 127
File attributes
changing, 95, 368
described, 94
viewing, 94, 368
File control blocks, opening concurrently, 492
File Display Options command, MS-DOS Shell, 47 to 49
File list in the MS-DOS Shell window
described, 30
starting programs from, 55
File sharing
backing up files on networks, 372
installing, 565 to 566
File transfer memory, 251
Filename extensions
choosing for a file, 70
for batch programs, 70, 227
for program files, 70
for system files, 70
for text files, 71
order of precedence for running commands, 540
sorting a directory listing by extensions, 438
substituting files for replaceable variables, 498
using to
back up files, 139
combine files, 392
Filename extensions (*continued*)
using to (*continued*)
delete files, 88
rename files, 83
restore files to a directory, 143
view filenames, 109
using with wildcards, 73
Filenames
changing, 83, 551 to 552
conventions for, described, 69 to 70
in command syntax, defined, 361
number sign (#) in deleted filenames, 91
specifying with wildcards, 73 to 75
viewing by using wildcards, 109
viewing directory contents, 108
Files
See also Batch programs, Hidden files, System files
adding to a directory using Replace, 123, 552 to 554
alphabetizing data in, 568 to 570
associating with a program, 58 to 60
backing up, 136 to 140, 370 to 374
backup log file, 371 to 372
combining, 80, 389 to 393
comparing
using the Comp command, 387 to 389
using the Diskcomp command, 441 to 442
using the Fc command, 93, 488 to 491
converting .EXE files to binary format, 483 to 484
copying, 77 to 82, 389 to 393
copying from keyboard to printer, 81
copying to a printer port, 81
copying with directories, 118 to 120, 587 to 590
creating. *See* MS-DOS Editor or Edlin
decreasing time needed to open, 486
deleting, 86 to 89, 432
deleting to increase system speed, 292
editing. *See* MS-DOS Editor or Edlin
executable. *See* Executable files
expanding compressed files, 485
listing by common part of filenames, 109
loading into memory with Debug, 413 to 415
locking, 565 to 566
moving, 92
naming conventions for, 69 to 70
organizing, 16
path for, described, 103 to 104
printing, 83 to 85, 542 to 545
printing from MS-DOS Editor, 220
redirecting command output to, 162
renaming, 83, 551 to 552
renaming when copying, 79
replacing with updated versions, 122, 552 to 554

Files (*continued*)
restoring
after backing up, 141 to 144, 555 to 557
after deleting, 90 to 91, 579 to 581
from a defective disk, 145, 548 to 549
searching for, using MS-DOS Shell, 98
setting the number of open files, 252, 493
sharing, 565 to 566
sorting data in, 568 to 570
testing executable files, 399
tracking deleted files, 86, 522 to 525
types of, described, 70 to 71
using a search path
for data files, 363 to 365
for program or batch files, 123
using as command input, 163
verifying while writing to disk, 586
viewing
contents of, 75, 578
directory contents, 108
information about, 72, 99 to 100
one screen at a time, 537
Files command
in the CONFIG.SYS file, 250
introduction to using, 252
syntax and explanation, 493
Filters
described, 163
Find command, 164
More command, 163
Sort command, 164
Find command
introduction to using, 97
MS-DOS Editor, 216
syntax and explanation, 494 to 495
using as a filter, 164
Finding files
using a search path, 123
using MS-DOS Shell, 98
Finding text
using Edlin, 478 to 479
using MS-DOS Editor, 216
using the Find command, 97, 494 to 495
Fixed disk. *See* Hard disk
Flags, viewing contents of, 421 to 424
Floppy disk drives
3.5-inch, support for, 456
adding, 258 to 261
Floppy disk drives (*continued*)
described, 10, 127
detection of disk capacity, 131
Floppy disk system
displaying complete error messages, 386
increasing speed with a RAM disk, 310, 615
replacing files on, 553
Floppy disks
See also Disks
backing up by using MS-DOS Shell, 141
comparing, 441 to 442
copying, 444 to 446
creating a system disk, 133 to 134
formatting, 129 to 132
inserting and removing, 12
labeling and storing, 11
protecting information on, 12
restoring, 132, 581 to 585
sizes and types of, 125 to 126
For command, 496 to 497
Foreground colors, escape sequences for, 271, 595
Format command
See also Unformat command
creating a system disk, 133 to 134
introduction to using, 129 to 131
MS-DOS Shell, 131
restriction with networks, 501
restriction with reassigned drives, 501
syntax and explanation, 498 to 502
Formatting disks
after using Fdisk, 159 to 160
creating a system disk, 133 to 134
described, 127 to 128
preparing for MS-DOS files, 129 to 130, 498 to 502
quick format, 129, 132, 501
reducing fragmentation, 298
restoring after reformatting, 132, 581 to 585
restriction with reassigned drives, 501
safe format, 129, 501
specifying disk capacity, 130
unconditional format, 129, 499
using MS-DOS Shell, 131 to 132
when backing up files, 370
when copying a floppy disk, 444 to 446
Fragmentation
defined, 296
reformatting your hard disk, 298
transferring with Diskcopy, 446
Function keys, described, 8

G

G (go) command, Debug, 409 to 411
Goto command
  introduction to using, 236 to 237
  restriction on using in macros, 175
  syntax and explanation, 503
Graftabl command
  See also Chcp command, Mode command
  syntax and explanation, 504 to 505
Graphics adapters
  configuring, 536
  modes supported by Graphics command, 505
Graphics command
  See also Print command
  syntax and explanation, 505 to 508
Graphics mode
  ANSI escape sequences for, 595 to 596
  changing screen modes in MS-DOS Shell, 184
  displaying extended characters, 504
  for program items, 200
  printing the screen contents, 505 to 508
  starting MS-DOS Shell in, 454
Greater-than sign, redirection character, 161
Groups. *See* Program groups
GW-BASIC programs, converting to QBasic, 548

H

H (hex) command, Debug, 411
Handles
  creating, 428
  deallocating to expanded memory, 429
  extended memory block, 610
  mapping expanded-memory pages, 430
  used by EMM386.EXE, 607
Hard disk drives, described, 10
Hard disks
  See also Disks
  adjusting the interleave, 300
  backing up by using MS-DOS Shell, 141
  formatting, 501
  formatting after partitioning, 159 to 160
  partitioning, 147 to 160
  reorganizing
    by reformatting, 298
    by using disk-compaction programs, 297 to 298
  restoring, 132, 581 to 585
  restoring corrupted partition tables, 581, 584
  restriction with DRIVER.SYS, 604
Hard disks (*continued*)
  safe format, 129
  saving partition-table information, 524, 584
  speeding up. *See* Speeding up your system
  unconditional format, 129
Hardware code page, defined, 334
Hardware interrupt handling, 570
Hardware, described, 3 to 9
Head number, specifying
  for a logical drive, 603
  for a physical drive, 456
Help command, 508
Help menu in MS-DOS Shell, described, 65
Help Path command, MS-DOS Editor, 211
Help, online
  creating
    for program groups, 185
    for program items, 198
  for MS-DOS commands, 24, 362, 508
  for MS-DOS Editor, 209 to 211
  for MS-DOS Shell, 62 to 65
  printing an MS-DOS Editor Help topic, 220
.HEX files, loading into memory with Debug, 414
Hexadecimal arithmetic, in Debug, 411
Hidden file attribute, 94, 368
Hidden files
  changing or viewing attributes of, 368
  removing before using Rmdir, 558
  restriction on updating, 553
  viewing
    using MS-DOS Shell, 48
    using the Dir command, 437
High memory area (HMA)
  disk buffers in, 376
  freeing for use by programs, 288
  loading MS-DOS into, 447
  managing with HIMEM.SYS, 610
  running MS-DOS in, 282
HIMEM.SYS device driver
  Device command syntax for, 610 to 613
  installing, 281
  managing extended memory, 279 to 281
  required
    for EMM386.EXE, 317, 608
    for the Devicehigh command, 436
    for the Loadhigh command, 519
  running MS-DOS in extended memory, 283
HMA. *See* High memory area

I

I (input) command, Debug, 412
I (insert) command, Edlin, 467
If command
  introduction to using, 235 to 237
  processing exit codes with errorlevel, 373, 509
  syntax and explanation, 509 to 510
Input and output devices
  changing with Command, 385
  changing with Ctty, 396
INS key, for changing commands, 168
Insert mode, 174
Inserting text with Edlin, 467
Install command
  in the CONFIG.SYS file, 250
  restrictions with certain programs, 510
  syntax and explanation, 510
Installation disks, expanding files, 485
Interleave, adjusting, 300
Internal clock, setting, 575 to 576
Internal commands, defined, 24, 358
International. *See* Customizing for international use
Interrupt 15h interface, 522, 610
Interrupt 21h functions, 365
IO.SYS file
  and the primary DOS partition, 147
  copying when formatting a disk, 133, 500
  copying with the Sys command, 574
  restriction on backing up, 372

J

Join command
  See also Assign command, Subst command
  restriction with deletion tracking, 523
  restrictions with other commands, 512
  syntax and explanation, 511 to 513
Jumps
  in a batch program, 236 to 237, 242 to 244
  overriding with Debug, 402

K

K. *See* Kilobytes
Key combinations, conventions for, xiii
Keyb command
  See also Chcp command
  introduction to using, 337 to 338
  syntax and explanation, 513 to 516
  viewing code-page information, 349
KEYB.COM file, 339
KEYBOARD.SYS file
  default keyboard program, 513
  specifying the path for, 338
Keyboards
  codes for redefining keys, 263, 596
  described, 5 to 9
  diagrams for international layouts, 623
  enhanced, 513, 573
  introduction to redefining keys, 265 to 266
  loading a code page, 346
  preparing for code pages, 341 to 343
  producing extended characters, 631
  rearranging and adding characters, 337 to 339
  remapping extended keys, 593
  setting typematic rate, 536
  starting the Keyb program, 337 to 339
  switching between configurations, 337, 515
  switching to conventional functions, 573
  table of codes for, 514
  viewing code-page information, 349
Keys
  arrow keys, 7
  assigning commands to, 266
  conventions used in this guide, xiii
  described, 5 to 9
  for changing commands, 21, 166 to 168
  for scrolling in the MS-DOS Shell window, 40
  for selecting directories in MS-DOS Shell, 44
  for selecting files in MS-DOS Shell, 49
  for stopping a batch program, 227
  for stopping a command, 23
  for the Survival Guide in MS-DOS Editor, 210
  for working with Doskey, 170 to 174
  for working with MS-DOS Editor, 212 to 213
  function keys, 8
  redefining, 265 to 266
  shortcuts for MS-DOS Shell commands, 33
Kilobytes, defined, 4, 126

L

L (list) command, Edlin, 469
L (load) command, Debug, 413 to 415
Label command
  See also Dir command, Vol command
  introduction to using, 135
  restriction with reassigned drives, 517
  syntax and explanation, 516 to 517
Labels
  for disks. *See* Volume label
  in batch programs. *See* Goto command
Language. *See* Country

Lastdrive command
freeing conventional memory, 285
in the CONFIG.SYS file, 250
introduction to using, 253
syntax and explanation, 517
LCD monitors, 341, 600
Less-than sign, redirection character, 162
Lh command. *See* Loadhigh command
Line command, Edlin, 462
Line wrap, escape sequence for, 596
List boxes, in MS-DOS Shell, 35, 37
Listing
file contents, 75, 578
filenames in a directory, 107 to 111, 437 to 441
lines, with Edlin, 469
macros, 449
memory contents, with Debug, 405
previous commands, 170, 449
Loadhigh (lh) command
See also Dos command
freeing conventional memory, 325 to 326
syntax and explanation, 518 to 519
Locking files, 565 to 566
Log files, creating with Backup, 371 to 372
Logical drives
assigning two letters to a physical drive, 261
changing the number of, 253
creating or changing, 155
defining parameters for, 602 to 604
deleting, 157 to 158
described, 148, 602
drive letters in extended DOS partitions, 156
formatting after creating or changing, 159 to 160
versus physical drives, 258
Lost allocation units
converting to files, 382
reported by Chkdsk, 382
Lowercase letters for typing commands, 20
LPTn ports
configuring printers for, 255, 527
support for code-page switching, 613

M

M (move) command
Debug, 415
Edlin, 470
Macros
creating, 176
deleting, 178, 453
described, 175 to 176
Macros (*continued*)
editing, 178
limit on length of, 175
naming with MS-DOS command names, 177, 453
restriction on using Echo and Goto commands, 175
running, 177, 453
running from batch programs, 453
saving, 178
stopping or skipping commands in, 175
using redirection characters and pipes, 180, 452
using replaceable parameters in, 179 to 180
versus batch programs, 175 to 176
viewing a list of, 449
Main group, MS-DOS Shell, 30, 54
Mapping expanded-memory pages, 430
MB. *See* Megabytes
Md command. *See* Mkdir command
Megabytes, defined, 4, 126
Mem command
See also Chkdsk command
determining memory needed for device drivers, 436
getting upper-memory-area information, 320 to 321
syntax and explanation, 519 to 522
Memory
allocating for disk buffers, 375 to 376
and Edlin, 463, 481
and Graftabl, 505
comparing portions of, 403
conventional. *See* Conventional memory
copying, 415
creating a RAM disk, 309 to 313
creating a SMARTDrive cache, 304 to 308
described, 4, 274 to 277
entering data into, 406 to 408
expanded. *See* Expanded memory
extended. *See* Extended memory
filling addresses with values, 408
for drives specified by Lastdrive, 518
for file transfers, 251
for the Fastopen program, 487
freeing for use by programs
conventional, 281 to 286
expanded, 288
extended, 287
HMA. *See* High memory area
loading a file or sectors into, 413 to 415
locations, distinguishing in Debug, 402
managers, described, 277
managing. *See* Optimizing your system
searching, 423
specifying amount for a device driver, 436

Memory (*continued*)
specifying in MS-DOS Shell for programs, 199
transient versus resident, 385
upper. *See* Upper memory area
viewing amount of used and free, 519 to 522
viewing contents of, 405
Memory managers
for expanded memory, 276
for extended memory, 275
included with MS-DOS, described, 277
using EMM386.EXE
emulating expanded memory, 289 to 290
managing the upper memory area, 318
using HIMEM.SYS, 279 to 281
Memory-resident programs
loading, 510
running in the upper memory area, 325 to 326
Menu bar, MS-DOS Shell window, 29
Menus
creating with batch programs, 238 to 244
in MS-DOS Shell
described, 33
selecting, 32
selecting in MS-DOS Editor, 206
Merging files, with Edlin, 480
Messages
displaying on a floppy disk system, 386
displaying when a batch program runs, 229 to 230, 457
suppressing in MS-DOS Shell, 60
Microsoft Windows version 3.0
386 enhanced mode versus EMM386.EXE, 290
and the SMARTDrive cache size, 306
disabling mouse device drivers, 286
getting upper-memory-area information, 321
using EMM386.EXE from MS-DOS, 289
using HIMEM.SYS from MS-DOS, 279
using SMARTDRV.SYS from MS-DOS, 304
using the ram switch with EMM386.EXE, 319
using with extended memory, 276
using with the Stacks command, 285
Mirror command
See also Unformat and Undelete commands
in the AUTOEXEC.BAT file, 88, 128
introduction to using, 86 to 87
syntax and explanation, 522 to 525
Mirror program
See also Mirror command
described, 128
recovering lost information, 132, 582
removing deletion tracking, 523 to 524
restriction with reassigned drives, 523
Mirror program (*continued*)
saving information
about a disk drive, 128
about disk partitions, 128, 524, 584
about the current disk, 128
setting up deletion tracking, 86, 522 to 525
starting, 522
viewing mirror-file information, 583
Mkdir (md) command
See also Rmdir command
introduction to using, 113 to 114
syntax and explanation, 525
Mnemonics used in Debug, 402
Mode command
configuring printers, 255, 527
configuring serial ports, 257, 529
in the AUTOEXEC.BAT file, 246, 248
introduction to using, 255
redirecting printing, 532
setting device code pages, 346 to 348, 533
setting display mode, 535
setting typematic rate, 536
summary of functions, 526
viewing code-page information, 349
viewing device status, 531
Modem, described, 9
Monitors
See also Display adapters, Screen
described, 4
viewing code-page information, 349
Monochrome
display adapter
configuring, 536
using with DISPLAY.SYS, 601
starting MS-DOS Editor, 459
starting MS-DOS Shell, 455
More command
See also Dir command, Type command
introduction to using, 163
syntax and explanation, 537
Mouse
described, 9
installing a device driver for, 434
pointer in the MS-DOS Shell window, 29
MOUSE.SYS device driver, installing, 251
Move command, MS-DOS Shell, 92
Moving
files, 92
lines, with Edlin, 470
memory, with Debug, 415
text, with MS-DOS Editor, 214

MS-DOS
configuring for your system, 249 to 255
customizing. *See* Customizing your system
described, 13 to 14
linking to the upper memory area, 447
loading in the high memory area, 447
running Fdisk during MS-DOS setup, 149
running in extended memory, 282
version 5.0, new features, x
MS-DOS command interpreter, 384, 574
MS-DOS command line
cursor, defined, 20
described, 14, 19
MS-DOS commands
See also Commands
defined, 357
listed, 358
requesting online Help for, 24, 362, 508
using parameters with, 20
using switches with, 20
MS-DOS Editor
buffer, 214
buttons, 208
canceling a selected menu, 206
changing colors, 221
check boxes, 208
choosing commands, 206
copying text, 215
creating a file, 211, 217
customizing, 221
deleting text, 215
described, 203
dialog boxes, described, 207 to 208
finding text, 216
Help system, 209 to 211
hiding scroll bars, 221
moving text, 214
moving the cursor, 212 to 213
online introduction to, 205
opening a file, 218
printing a file, 220
printing a Help topic, 220
quitting, 211
replacing text, 216
saving a file, 219
selecting items in dialog boxes, 208
selecting menus, 206
selecting text, 213
setting tab stops, 221
specifying the Help path, 211
MS-DOS Editor (*continued*)
starting
from MS-DOS Shell, 204
with the Edit command, 204
with the QBasic command, 547
status bar, 209
text boxes, 207
using your keyboard, 212 to 213
MS-DOS Interrupt 21h functions, 365
MS-DOS partitions. *See* Fdisk program
MS-DOS prompt. *See* Command prompt
MS-DOS QBasic program, online help for, 547
MS-DOS Shell
See also MS-DOS Shell window
adding program groups, 184 to 186
adding programs to the Active Task List, 57
associating files with programs, 58 to 59
backing up a floppy disk, 141
backing up files and directories, 140
backing up your hard disk, 141
changing
file attributes, 96
group contents, 186 to 189
group properties, 201
program item properties, 190 to 201
copying files, 82, 118
creating a directory, 114
customizing. *See* Customizing MS-DOS Shell
deleting a directory, 117
deleting a program group, 186
deleting files, 89
described, 27
formatting floppy disks, 131 to 132
Help
creating Help messages, 185, 198
requesting, 62
using the Help menu, 64 to 65
viewing related procedures, 63
leaving temporarily, 65
moving files, 92
printing files, 85
quitting, 66
quitting programs, 58
removing file associations, 60
renaming a directory, 121
renaming files, 83
restoring files, 144
running multiple programs, 56
searching for files, 98

MS-DOS Shell (*continued*)
starting programs
from a program group, 54
from the file list, 55
using the Run command, 55
starting
from the AUTOEXEC.BAT file, 247
from the command prompt, 27
switching between programs, 56 to 57
switching to the command prompt, 30
Task Swapper, 31, 56 to 57
viewing
directories and filenames, 41
file attributes, 95
file contents, 76
file information, 42, 99 to 100
hidden and system files, 48
program groups and items, 42
MS-DOS Shell window
See also Program groups
canceling
a dialog box, 39
a selected menu, 33
an option, 38
selections, 53
changing
color schemes, 183 to 184
screen modes, 184
the current directory, 43 to 44
views, 41
choosing command buttons, 36
choosing commands, 33 to 34
collapsing directory levels, 46
customizing. *See* Customizing MS-DOS Shell
described, 28 to 29
elements of
Active Task List, 31, 56 to 57
check boxes, 35, 39
dialog boxes, 34 to 40
directory tree, 29, 41
drive icons, 29
file list, 30
list boxes, 35
menu bar, 29
menus, 33
mouse pointer, 29
option buttons, 35
program group, defined, 41
program item, defined, 41
program list, 30
root directory, 41
scroll bars, 29, 39 to 40
selection cursor, 29
MS-DOS Shell window (*continued*)
elements of (*continued*)
status bar, 29
text boxes, 35 to 36
title bar, 29
expanding directory levels, 45 to 46
extending a selection, 50 to 52
moving within dialog boxes, 35
opening program groups, 54
rearranging items in program groups, 189
repainting the screen, 61
selecting
a disk drive, 43
a file, 49
a menu, 32
all files, 52
an area, 31
file display options, 47 to 49
files across directories, 52
files in sequence, 50
files not in sequence, 51
items in list boxes, 37
multiple groups of files, 52
options, 38
sorting filenames, 48 to 49
suppressing confirmation messages, 60
updating, 61
updating disk information, 43
MS-DOS version number
including in the command prompt, 546
setting for programs, 561 to 564
viewing, 585
MS-DOS version table
for executable files, 564
for programs, 561 to 563
loading into memory, 617
MSDOS.SYS file
and the primary DOS partition, 147
copying
using the Sys command, 574
when formatting a disk, 133, 500
restriction on backing up, 372

N

N (name) command, Debug, 416 to 418
Near prefix, using in Debug, 402
Network commands, defined, 358
Networks
appending directories on network drives, 365
backing up shared files, 372
defining logical drives, 253
installing file sharing and locking, 565 to 566

Networks (*continued*)
reassigning network drives, 367
restrictions on using
with Chkdsk, 382
with Diskcomp, 443
with Fastopen, 487
with Format, 501
with port retry options, 531
with printer retry options, 528
with Recover, 549
with Sys, 574
with Unformat, 581
New command
MS-DOS Editor, 217
MS-DOS Shell, 185, 187
Nlsfunc command
See also Chcp command, Mode command
introduction to using, 345
syntax and explanation, 538
Non-DOS partitions, defined, 148
Numbers, typing, 6
Numeric keypad, described, 6

O

O (output) command, Debug, 418
Online Help
creating Help messages
for program groups, 185
for program items, 198
for MS-DOS commands, 24, 362, 508
for MS-DOS Editor, 209 to 211
for MS-DOS Shell, 62 to 65
printing an MS-DOS Editor Help topic, 220
Open command, MS-DOS Editor, 218
Open files accessed concurrently, 493
Opening files in MS-DOS Editor, 218
Opening files quickly, 486
Opening program groups, 54
Operands in Debug, 402
Operating system, defined, 13
Operation codes (opcodes), 401, 410
Optimizing your system
by freeing conventional memory
described, 281
running MS-DOS in extended memory, 282
streamlining your AUTOEXEC.BAT file, 286
streamlining your CONFIG.SYS file, 284
by freeing expanded memory, 288
by freeing extended memory, 287
Optimizing your system (*continued*)
by increasing system speed
adjusting the hard-disk interleave, 300
compacting your hard disk, 297 to 298
deleting unnecessary files, 292
described, 291
reducing file search time, 295
reformatting your hard disk, 298
using a RAM disk, 309 to 313
using a SMARTDrive cache, 304 to 308
using the Buffers command, 300
using the Chkdsk command, 294 to 295
using the Fastopen program, 302 to 304
by managing extended memory, 279 to 281
by managing the upper memory area
getting upper-memory-area information, 320 to 321
installing EMM386.EXE, 318
loading programs by size, 327
troubleshooting, 327 to 330
by running programs in the upper memory area
described, 313 to 316
device drivers, 323 to 325
memory-resident programs, 325 to 326
moving programs into UMBs, 322 to 323
setting up the CONFIG.SYS file, 317
described, 273 to 274
summary of methods, 331 to 332
Option buttons in MS-DOS Shell, 38
Options
selecting in MS-DOS Editor, 208
selecting in MS-DOS Shell, 37 to 38
Organizing files and directories, 15 to 17
Overlapping copy operations, in Debug, 415

P

P (page) command, Edlin, 472
P (proceed) command, Debug, 419
Page frames, described, 276
Pages in expanded memory, described, 276
Parallel ports
configuring printers connected to, 255, 527
redirecting output to a serial port, 532
support for code-page switching, 613 to 614
Parameters
in command syntax, defined, 361
in startup commands for program items, 192 to 195
using with a command, 20
Parent directory, defined, 102
Parity for serial ports, 257, 530

Partition table for a hard disk
restoring, 581, 584
saving information about, 524, 584
Partitioning a hard disk, 147 to 160
Partitions
See also Fdisk program
described, 147 to 149
saving partition-table information, 524, 584
PARTNSAV.FIL file, 524, 582, 584
Passwords
for program groups, 186
for program items, 188
Paste command, MS-DOS Editor, 214 to 215
Path
See also Search path
described, 103 to 104
in command syntax, defined, 361
joining to a disk drive, 511 to 513
limit on length of, 104
specifying multiple search paths, 540
substituting for a drive letter, 146, 571
viewing the directory structure of, 577
Path command
See also Append command
in the AUTOEXEC.BAT file, 246, 248
introduction to using, 123 to 124
syntax and explanation, 539 to 541
using with appended directories, 364
PATH environment variable, 560
Pause command
introduction to using, 231
syntax and explanation, 541
using to create customized menus, 239
PAUSE key
stopping a batch program temporarily, 227
stopping a command temporarily, 23
PCTRACKR.DEL file, 86, 523
Period (.) and double period (..) directories, 103
Pipe
See also Redirecting command input or output
defined, 361
sending macro output to another command, 452
with the Dir command, 439
with the Find command, 495
with the For command, 498
with the More command, 164
with the Sort command, 165
with the Tree command, 578
with the Type command, 579
Ports
AUX, using for input and output, 397
configuring, 257, 529
configuring printers for, 527
described, 9
device names for, 543
sending a byte value to, with Debug, 418
support for code-page switching, 613 to 614
viewing a byte at a port address, 412
Prefixes in Debug, 402
Prepared code page, defined, 334
Prevent Program Switch option, MS-DOS Shell, 201
Primary DOS partition
and drive letters, described, 156 to 160
creating, 152 to 154
described, 147 to 148
Print command
See also Graphics command, Mode command
introduction to using, 83 to 85
MS-DOS Editor, 220
MS-DOS Shell, 85
syntax and explanation, 542 to 545
Print queues
adding files to, 544
deleting files from, 543
effect on system efficiency, 85
emptying by canceling printing, 85
limit on length of an entry, 544
viewing, 85
PRINT SCREEN key, 507
PRINTER.SYS device driver
Device command syntax for, 613 to 614
preparing your printer for code pages, 343 to 345
Printers
breaking a time-out loop, 529
configuring, 255, 527
described, 9
loading a code page, 346
options for the Graphics command, 505 to 508
preparing for code pages, 343 to 345
redirecting command output to, 162
redirecting output to a serial port, 255, 532
setting parallel-printer modes, 529
specifying a parallel port for, 527
viewing code-page information, 349
Printing
a directory listing, 440
a display screen, 505 to 508
a Tree listing, 578
canceling, 85

Printing (*continued*)
files from MS-DOS Editor, 220
files generated by programs, 544
Help topics in MS-DOS Editor, 220
increasing speed of the Print command, 543
text files, 83 to 85, 542 to 545
text files, by using MS-DOS Shell, 85
using a print queue, 85
without using a print queue, 81
PRN parallel port, 527, 613
Processor, described, 4
Program files
See also Executable files
described, 71
Program groups
adding, 184 to 186
adding passwords for, 186
adding program items to, 186 to 188
changing properties of, 201
copying program items, 188
creating Help messages for, 185
defined, 41
deleting, 186
deleting program items from, 189
opening, 54
rearranging program items, 189
starting programs from, 54
viewing, 42
Program items
adding passwords for, 188, 198
adding to program groups, 186 to 188
changing properties of, 190
copying among program groups, 188
creating Help messages for, 198
creating startup commands for, 187, 191 to 195
defined, 41
defining shortcut keys for, 188, 196
deleting from program groups, 189
preventing switching, 201
rearranging, 189
reserving shortcut keys for, 200
setting video mode for, 200
specifying memory requirements for, 199
using a startup directory for, 188, 195
Program List command, MS-DOS Shell, 42
Program list in the MS-DOS Shell window, 30
Program/File Lists command, MS-DOS Shell, 42
Programs
See also Program items
adding to the Active Task List, 57
and SMARTDrive cache size, 618
associating files with, 58 to 60
Programs (*continued*)
loading
using the Install command, 510
using the Loadhigh command, 518 to 519
quitting from MS-DOS Shell, 58
redirecting disk operations with Assign, 366
running
from a RAM disk, 311
from MS-DOS Shell, 54 to 56
setting MS-DOS version for, 561 to 564
using Fcbs for older programs, 492
viewing status when loaded in memory, 520
Prompt command
See also Date command, Time command
introduction to using, 106 to 107
repositioning the command prompt, 268
repositioning the cursor, 267
running ANSI escape sequences, 263
syntax and explanation, 545 to 547
using in customized menus, 239, 244
PROMPT environment variable, 560
Prompts
Debug, 399
Edlin, 461
MS-DOS. *See* Command prompt
Prompts for more information, described, 22
Properties command, MS-DOS Shell, 190, 201
Properties in MS-DOS Shell
advanced, 198
application shortcut keys, 196
Call command in startup commands, 192
for groups, changing, 201
for program items, changing, 190
Help messages for program items, 198
memory requirements for programs, 199
password for program items, 198
pause after program exit, 197
program switching, 201
replaceable parameters in startup commands, 192 to 195
reserved shortcut keys, 200
startup commands, 191
startup directory for program items, 195
video mode, 200
Pseudoinstructions in Debug, 402

Q

Q (quit) command
Debug, 420
Edlin, 473
QBasic command, 547 to 548
Question mark (?) wildcard, 73 to 74
Quick access to files, 486

Quick format, 129, 132, 501
Quick Format command, MS-DOS Shell, 132
Quitting
  COMMAND.COM program, 485
  Edlin
    and saving the current file, 467
    without saving changes, 473
  MS-DOS Editor, 211
  MS-DOS Shell, 66
  programs from MS-DOS Shell, 58
  the Debug program, 420

R

R (register) command, Debug, 421 to 423
R (replace) command, Edlin, 474 to 476
RAM. *See* Random-access memory
RAM disk
  avoiding conventional memory, 284
  caution against storing data files, 310
  described, 309
  in emulated expanded memory, 310
  running programs from, 311
  setting the TEMP environment variable to, 312
  setting up, 614 to 616
RAMDRIVE.SYS device driver
  Device command syntax for, 614 to 616
  freeing expanded memory, 288
  freeing extended memory, 287
  installing, 310
  loading in the upper memory area, 310
  using to speed up your system, 309 to 313
Random-access memory, described, 4
Rd command. *See* Rmdir command
Read-only file attribute, 94, 368
Read-only files
  changing or viewing attributes of, 368
  replacing, 553
  restoring, 555
Read-only memory, tracing instructions in, 425
Reassigned drives
  formed by the Assign command, 366
  formed by the Join command, 511
  restrictions on using
    with Append, 365
    with Backup, 372
    with Chkdsk, 382
    with Diskcomp, 443
    with Diskcopy, 366
    with Fdisk, 493
    with Format, 501
    with Label, 517
    with Mirror, 523
Reassigned drives (*continued*)
  restrictions on using (*continued*)
    with Recover, 549
    with Restore, 556
    with Subst, 572
    with Sys, 574
  virtual, 571
.REC files, 145, 549
Recalling commands, 449
Recover command
  See also Chkdsk command
  exceeding space in the root directory, 548
  introduction to using, 145 to 146
  restoring a disk after using, 581
  restriction with networks, 549
  restriction with reassigned drives, 549
  syntax and explanation, 548 to 549
  versus Restore command, 549
Recovering
  allocation units, 294 to 295
  deleted files
    with deletion tracking, 90, 579 to 581
    without deletion tracking, 91, 580
  files from a defective disk, 145, 548 to 549
Recursive calls, 377
Redirecting command input or output
  described, 161 to 165
  in Doskey macros, 180, 452
  listing Doskey macros, 449
  printing a Tree listing, 578
  saving Chkdsk reports in a file, 383
  saving stored commands, 174, 449
  sending Dir command output to a file, 439
  setting TEMP before using a pipe, 163
  sorting data by using Sort, 164
  viewing data one screen at a time, 163
  with the Find command, 97, 164
  with the For command, 498
Redirecting parallel printer output, 532
Redirection characters
  combining commands with, 165
  defined, 161
  passing information through filters, 163
  redirecting command input, 163
  redirecting command output, 162
Refresh command, MS-DOS Shell, 43, 61
Register sets, allocating to EMM386.EXE, 607
Registers, viewing contents of, 421 to 424
Rem command
  See also Echo command
  in the CONFIG.SYS file, 250
  introduction to using, 232
  syntax and explanation, 550

Remarks, including in batch programs, 232, 550
Removing
a directory, 116, 558 to 559
a virtual drive, 146, 572
files, 86 to 89, 432
logical drives, 157 to 158
macros, 178
Ren command. *See* Rename command
Rename (ren) command
See also Label, Copy, Xcopy commands
introduction to using, 83
syntax and explanation, 551 to 552
Rename command, MS-DOS Shell
renaming directories, 121
renaming files, 83
Renaming
a directory, 120 to 121
a disk, 134, 516
files
using MS-DOS Shell, 83
using the Rename command, 83, 551 to 552
when copying with the Copy command, 79
Reorder command, MS-DOS Shell, 189
Repaint Screen command, MS-DOS Shell, 61
Repeating commands without retyping
using Doskey, 170 to 172
using editing keys, 167
Replace command
See also Attrib command
introduction to using, 122 to 123
restriction on hidden or system files, 553
syntax and explanation, 552 to 554
Replace mode, 174
Replaceable parameters
calling Set variables from batch programs, 560
changing the position of, 567
in startup commands for program items, 192 to 195
replaceable variables in the For command, 496
using in batch programs, 234 to 235
using in macros, 179 to 180, 452
using to create customized menus, 242
Replacing text
using Edlin, 474 to 476
using MS-DOS Editor, 216
Reserve Shortcut Keys option, MS-DOS Shell, 200
Restore command
See also Backup command
affected by country code, 394
introduction to using, 141 to 144
restriction with reassigned drives, 556
restriction with system files, 556
syntax and explanation, 555 to 557
using an old version of, 372
Restore command (*continued*)
using with an old version of Backup, 556
versus Recover command, 549
with files backed up from reassigned drives, 372
Restore Fixed Disk command, MS-DOS Shell, 144
Restore View command, MS-DOS Shell, 76
Restoring
disks, 132, 581 to 585
files and directories
after backing up, 141 to 144, 555 to 557
after deleting, 90 to 91, 579 to 581
viewing filenames on a backup disk, 144
Retry settings for serial ports, 256 to 257
Reverse sorting (Z to A), 569
Reverse video, for the command prompt, 546
Rmdir (rd) command
See also Attrib command, Dir command
introduction to using, 116
syntax and explanation, 558 to 559
using to rename a directory, 120
ROM. *See* Read-only memory
ROM BIOS, defined, 258
Root directory
changing to, 115
defined, 101
in MS-DOS Shell, 41
placing recovered files in, 548
Run command, MS-DOS Shell, 55, 59
Running
ANSI escape sequences, 263 to 264
batch programs, 227
macros, 177, 453
programs
from MS-DOS Shell, 54 to 56
from the AUTOEXEC.BAT file, 247
from within Debug, 409 to 411

S

S (search) command
Debug, 423
Edlin, 478 to 479
Safe format, 127, 129, 501
Save As command, MS-DOS Editor, 218 to 219
Save command, MS-DOS Editor, 218 to 219
Saving
disk information, 522 to 525, 582
files
in appended directories, 364
in Debug, 427
in Edlin, 467, 481
in MS-DOS Editor, 219
macros, 178

Scanner, installing a device driver for, 434
Screen
  changing attributes of, 269 to 272
  changing modes in MS-DOS Shell, 184
  clearing with Cls, 383
  creating custom screen displays, 268
  escape sequences for display mode, 596
  loading a code page, 346
  preparing for code pages, 341 to 343
  printing, 505 to 508
  repainting in MS-DOS Shell, 61
  saving the display, 605
  updating in MS-DOS Shell, 61
Scroll bars
  in the MS-DOS Editor window, 213, 221
  in the MS-DOS Shell window, 29, 39 to 40
Scrolling
  a directory display, 437
  in the MS-DOS Editor window, 213
  in the MS-DOS Shell window, 39 to 40
Search command, MS-DOS Shell, 98
Search path
  described, 123
  for data files, 363 to 365
  for executable files, 123, 539, 541
Searching for byte values, 423
Searching for files
  reducing file search time, 295
  using a search path, 123
  using MS-DOS Shell, 98
Searching for text
  using Edlin, 478 to 479
  using MS-DOS Editor, 216
  using the Find command, 97, 494 to 495
Secondary buffer cache
  specifying the size of, 376
  using to speed up programs, 301
  versus disk-caching programs, 301
Sectors
  caution against writing in Debug, 427
  comparing using Diskcomp, 441
  defective, reported by the Chkdsk command, 383
  defined, 127
  loading into memory with Debug, 413 to 415
  logical versus physical disk errors, 383
  recovering data from a defective disk, 145, 548 to 549
  specifying
    for a logical drive, 603
    for a physical drive, 456
    when formatting a disk, 500
Select Across Directories command, MS-DOS Shell, 52
Select All command, MS-DOS Shell, 52, 118
Selecting
  a disk drive in the MS-DOS Shell window, 43
  an area in the MS-DOS Shell window, 31
  check boxes
    in MS-DOS Editor, 208
    in MS-DOS Shell, 39
  files in MS-DOS Shell, 49 to 52
  menus
    in MS-DOS Editor, 206
    in MS-DOS Shell, 32
  options
    in MS-DOS Editor, 208
    in MS-DOS Shell, 37 to 38
  text, in MS-DOS Editor, 213
Selection cursor, MS-DOS Shell window, 29
Serial ports
  configuring, 257, 529
  redirecting parallel ports to, 255, 532
Set command
  See also Dir, Path, Prompt, Shell commands
  in the AUTOEXEC.BAT file, 247 to 248
  presetting Dir command options, 439 to 440
  syntax and explanation, 559 to 560
Setver command, 561 to 564
SETVER.EXE device driver, 617
Shadow RAM, 612
Share command
  See also File sharing, Networks
  syntax and explanation, 565 to 566
Shell. *See* MS-DOS Shell
Shell command
  in the CONFIG.SYS file, 250
  increasing the environment size, 386, 567
  syntax and explanation, 566 to 567
Shift command, 567 to 568
Shortcut keys
  and programs loaded with Install, 510
  for menu commands in MS-DOS Shell, 33
  for starting program items, 188, 196
  reserving for program items, 200
Show Information command, MS-DOS Shell, 95, 99
Single File List command, MS-DOS Shell, 41
Single period (.) for current directory, 103
Size of files, viewing, 72
SMARTDrive cache
  creating in expanded memory, 308
  creating in extended memory, 308
  described, 304
  effect of programs on, 618
  in emulated expanded memory, 308
  specifying the size of, 306 to 307
  versus secondary buffer cache, 301

SMARTDRV.SYS device driver
See also SMARTDrive cache
Device command syntax for, 617 to 619
disabling to free conventional memory, 284
freeing expanded memory, 288
freeing extended memory, 287
installing, 306
restriction with disk-compaction programs, 618
using to speed up your system, 304 to 308
Software, described, 13
Sort command
introduction to using, 164
redirecting output to the More command, 537
syntax and explanation, 568 to 570
Sorting
data in files, 568 to 570
filenames
in a Dir command listing, 110, 439
in the MS-DOS Shell window, 48 to 49
keyboard input, 570
Source, defined, 362
Speed of the Print command, increasing, 543
Speeding up your system
by adjusting the hard-disk interleave, 300
by compacting your hard disk, 297 to 298
by deleting unnecessary files, 292
by reducing file search time, 295
by reformatting your hard disk, 298
by using a RAM disk, 309 to 313
by using a SMARTDrive cache, 304 to 308
by using the Buffers command, 300
by using the Chkdsk command, 294 to 295
by using the Fastopen program, 302 to 304
described, 291
summary of methods, 332
Stacks command
freeing conventional memory, 284
in the CONFIG.SYS file, 250
syntax and explanation, 570
Starting
batch programs, 227
Edlin, 460
macros, 177, 453
MS-DOS Editor, 204
MS-DOS QBasic, 547
MS-DOS Shell, 27, 454
programs
from MS-DOS Shell, 54 to 56
from the AUTOEXEC.BAT file, 247
the Debug program, 399
the Doskey program, 21, 169, 448
the Fastopen program, 303, 486
the Fdisk program, 150, 492
Starting (*continued*)
the Keyb program, 337 to 339, 513
the Mirror program, 522
the Nlsfunc program, 538
the Share program, 565
Startup commands for program items
creating, 191
running batch programs in, 192
using replaceable parameters, 192 to 195
Startup directory for program items, 188, 195
Startup procedures
commands for, 246 to 249
defined, 246
samples of, 247
Status bar
in the MS-DOS Editor window, 209
in the MS-DOS Shell window, 29
Stop bits for serial ports, 257, 530
Stopping
a batch program, 227, 231
a command, 23
a program or activity, 374
String, defined, 362
Subdirectories
backing up, 138
copying, 119
creating, 113 to 114
defined, 16, 102
deleting, 116 to 117
organizing files with, 101
renaming, 121
restoring, 143, 555
viewing, 111
viewing in MS-DOS Shell, 46
Subst command
See also Join command, Lastdrive command
introduction to using, 146
restriction with deletion tracking, 523
restrictions with other commands, 147, 572
syntax and explanation, 571 to 573
using instead of Assign, 367
Survival Guide, MS-DOS Editor, 205, 209 to 210
Switches
in command syntax, defined, 361
using with a command, 20
Switches command
in the CONFIG.SYS file, 250
syntax and explanation, 573
Syntax conventions used in this guide, 360 to 362
Sys command
See also Copy command, Xcopy command
introduction to using, 133 to 134
restriction with networks, 574

Sys command (*continued*)
restriction with reassigned drives, 574
syntax and explanation, 574 to 575
.SYS files, described, 71
System configuration. *See* Customizing your system
System file attribute, 94, 368
System files
See also Device drivers
changing or viewing attributes of, 368
copying when formatting a disk, 133, 500
copying with the Sys command, 574 to 575
described, 71
removing before using Rmdir, 558
restrictions
on backing up, 372
on restoring, 556
on updating with Replace, 553
viewing
using MS-DOS Shell, 48
using the Dir command, 437
System prompt. *See* Command prompt
System resources, described, 274 to 278

T

T (trace) command, Debug, 424
T (transfer) command, Edlin, 480
Tabs, setting in MS-DOS Editor, 221
Tape drives, reconfiguring with Drivparm, 457
Task Swapper, MS-DOS Shell
described, 31
EGA.SYS requirement for, 605
enabling, 56
quitting programs, 58
switching between programs, 57
TEMP directory, cleaning out, 293
TEMP environment variable
required for using a pipe, 163
setting to a RAM disk, 312, 616
Temporary files
deleting, 293
storing on a RAM disk, 312
Testing
batch programs, 228
executable files, 399
Text boxes
in MS-DOS Editor, 207
in MS-DOS Shell, 35 to 36
Text color, ANSI escape sequences for, 271
Text files
See also ASCII files
printing, 83 to 85
viewing, 75
Text mode
ANSI escape sequences for, 595
changing screen modes in MS-DOS Shell, 184
for program items, 200
starting MS-DOS Shell in, 454
Text, copying from keyboard to a file, 80
Time
displaying in the command prompt, 546
format
changing by using the Country command, 335
table of country codes for, 395
of file creation, viewing, 72
setting or viewing, 575 to 576
Time command
See also Date command
affected by country code, 394
in the AUTOEXEC.BAT file, 246 to 247
introduction to using, 22
syntax and explanation, 575 to 576
Time-out loop for printers, breaking, 529
Title bar, MS-DOS Shell window, 29
TMP environment variable, 312
Tracking deleted files, 86, 522 to 525
Tracks
comparing using Diskcomp, 442
defined, 125
specifying
for a logical drive, 603
for a physical drive, 456
when formatting a disk, 500
Transferring text, using Edlin, 480
Tree command
See also Dir command
introduction to using, 111 to 112
syntax and explanation, 577 to 578
.TXT files, described, 71
Type command
introduction to using, 75
stopping a long display, 76
syntax and explanation, 578
Typematic rate, setting, 536
Typing numbers, 6

U

U (unassemble) command, Debug, 425 to 426
UMB. *See* Upper memory blocks
Unconditional format, 127, 129, 499
Undelete command
  See also Del, Mirror, Unformat
  described, 86
  introduction to using, 90 to 91
  syntax and explanation, 579 to 581
  using Unformat after Undelete, 583
Undeleting files, 90 to 91, 579 to 581
Unformat command
  See also Format command, Mirror command
  introduction to using, 132
  restriction with Fdisk, 583
  restriction with Format /u switch, 582
  restriction with networks, 581
  syntax and explanation, 581 to 585
Update disks, expanding files from, 485
Updating
  directories, 122, 552 to 554
  system files, 574 to 575
  the MS-DOS Shell window, 61
Upper memory area
  described, 277
  getting information about, 320 to 321
  loading RAMDRIVE.SYS into, 310
  maintaining a link to, 318, 447
  managing with EMM386.EXE, 318
  moving programs into, 322 to 323
  optimizing the use of, 327
  preparing to use, 317
  running device drivers in, 323 to 325, 435
  running memory-resident programs in, 325 to 326, 518
  running the Fastopen program in, 303
  setting up the CONFIG.SYS file, 317
  troubleshooting, 327 to 330
  using to free conventional memory, 313 to 316
Upper memory blocks
  described, 277, 313
  providing for the Devicehigh command, 436
Uppercase letters for typing commands, 20

V

Ver command, 585
Verify command
  See also Chkdsk command
  syntax and explanation, 586
Verifying files
  while copying, 390, 588
  while writing to disk, 586
Version number. *See* MS-DOS version number
Version table. *See* MS-DOS version table
VGA. *See* Video graphics adapters
Video adapters
  configuring, 535
  supported by DISPLAY.SYS, 600
Video graphics adapters, 341, 600
Video Mode options, MS-DOS Shell, 200
View File Contents command, MS-DOS Shell, 76
View, in the MS-DOS Shell window, 41
Viewing
  a group of related filenames, 109
  amount of used and free memory, 519 to 522
  binary files, 578
  code-page information, 349 to 350
  command output one screen at a time, 537
  commands
    using Doskey, 170 to 172, 449
    using editing keys, 167
  current time, 575 to 576
  device status, 531
  directories, 107 to 111, 437 to 441
  directories, using MS-DOS Shell, 41, 43 to 46
  directory structure of a path or disk, 111, 577
  disk volume label and serial number, 135
  environment variables, 559 to 560
  expanded memory status, 431
  file attributes, 94
  file contents, 75, 578
  file contents, using MS-DOS Shell, 76
  file information using MS-DOS Shell, 42
  file size and creation time, 72
  filenames on a backup disk, 144
  files one screen at a time, 537

Viewing (*continued*)
flags and registers, 421 to 424
hidden and system files, 48
lines in a file, using Edlin, 469
list of Doskey macros, 449
memory contents with Debug, 405
program groups, 42, 54
Virtual drives
defined, 571
deleting, 146, 572
using instead of a drive letter, 146, 571
Vol command
See also Format command, Label command
introduction to using, 135
syntax and explanation, 587
Volume label
creating, changing, or deleting, 134 to 135, 516
defined, 134
specifying when formatting a disk, 129, 499 to 500
viewing
using the Dir command, 135, 437
using the Vol command, 135, 587
Volume serial number
assigned when copying a disk, 445
assigned when formatting a disk, 130
defined, 134
viewing
using the Dir command, 135, 437
using the Vol command, 135, 587

W

W (write) command
Debug, 427 to 428
Edlin, 481
Weitek coprocessor, enabling or disabling support, 482
Wildcards
backing up selected files, 139
clearing a directory, 89
copying a group of files, 78
defined, 73
deleting a group of files, 88
for groups of files, 73 to 75
for single letters, 74
renaming files
as they are copied, 79
using the Rename command, 83
restoring selected files to a directory, 143
viewing a group of filenames, 109
Write-protect notch, 12

X

XA (allocate expanded memory) command, 428
Xcopy command
See also Copy, Diskcopy, Sys commands
copying files with the archive attribute, 369
introduction to using, 118 to 120
syntax and explanation, 587 to 590
using Diskcopy instead of, 589
XD (deallocate expanded memory) command, 429
XM (map expanded-memory pages) command, 430
XMS memory. *See* Extended memory
XMS Memory options, MS-DOS Shell, 199
XS (display expanded-memory status) command, 431

Microsoft Corporation
One Microsoft Way
Redmond, WA 98052-6399

Microsoft®
Making it all make sense™



0291 Part No. 20885