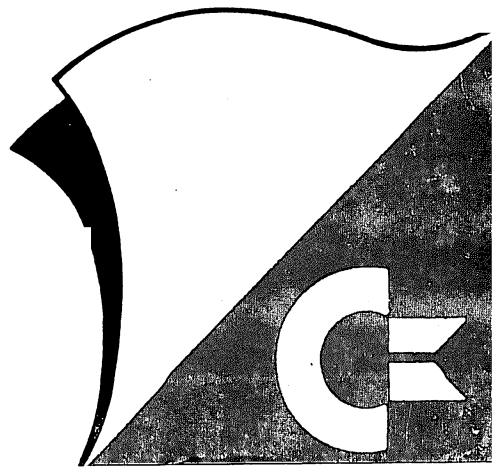




PERSONAL COMPUTER

- **MS-DOS® 3.2
User's Guide**
- **MS-DOS® 3.2
User's Reference**





COMMODORE PC PERSONAL COMPUTER

- **MS-DOS User's Guide**
- **MS-DOS User's Reference**

Information in this document is subject to change without notice and does not represent a commitment on the part of Commodore Business Machines or the Microsoft Corporation. The software described in this document is furnished under a license agreement or non-disclosure agreement. The software may be used or copied only in accordance with the terms of that agreement. It is against the law to copy the MS-DOS Disk Operating System on magnetic tape, disk, or any other medium for any purpose other than the purchaser's personal use.

Copyright © 1987 Commodore Electronics Limited.
All rights reserved.

Copyright © Microsoft Corporation, 1984, 1982, 1983, 1984, 1985, 1986, 1987.
All rights reserved.

Commodore is a registered trademark of Commodore Electronics Limited.

Microsoft®, MS-DOS®, GW-BASIC® and the Microsoft logo are registered trademarks of Microsoft Corporation.

USER'S MANUAL STATEMENT

WARNING:

This equipment has been certified to comply with the limits for a Class B computing device, pursuant to subpart J of Part 15 of the Federal Communications Commission's rules, which are designed to provide reasonable protection against radio and television interference in a residential installation. If not installed properly, in strict accordance with the manufacturer's instructions, it may cause such interference. If you suspect interference, you can test this equipment by turning it off and on. If this equipment does cause interference, correct it by doing any of the following:

- Reorient the receiving antenna or AC plug.
- Change the relative positions of the computer and the receiver.
- Plug the computer into a different outlet so the computer and receiver are on different circuits.

CAUTION: Only peripherals with shield-grounded cables (computer input-output devices, terminals, printers, etc.), certified to comply with Class B limits, can be attached to this computer. Operation with non-certified peripherals is likely to result in communications interference.

Your house AC wall receptacle must be a three-pronged type (AC ground). If not, contact an electrician to install the proper receptacle. If a multi-connector box is used to connect the computer and peripherals to AC, the ground must be common to all units.

If necessary, consult your Commodore dealer or an experienced radio-television technician for additional suggestions. You may find the following FCC booklet helpful: "How to Identify and Resolve Radio-TV Interference Problems." The booklet is available from the U.S. Government Printing Office, Washington, D.C. 20402, stock no. 004-000-00345-4.

This manual contains copyrighted and proprietary information. No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, photocopying, recording or otherwise, without the prior written permission of Commodore Electronics Limited.



User's Guide



Contents

Welcome v

- What is MS-DOS? vi
- Before You Begin vi
- How to Use This Guide viii

1 Learning About MS-DOS 1

- Terms You Should Know 2
 - Program 2
 - File 2
 - Filename 3
 - Directory 3
 - Volume Label 3
 - Disk drive 3
 - Drive name 4
 - Commands 5
- Keys You Use with MS-DOS 5
 - Special Keys 6
 - Control Key Combinations 6
 - The Return Key 7

2 Learning About Disks, Files, and Directories 9

- Floppy Disks 10
 - Disk Protection 11
- Hard Disks 12
- The Format Command 13
- How to Name Your Files 13
- Invalid Filenames 14
- Directories 14

3 Getting Started 17

- How to Start MS-DOS 17
- How to Quit 19
- How to Make a Backup Copy of Your MS-DOS Disk 19
- If You Have a Hard Disk 21
- If You Have Only One Floppy Disk Drive 23

4 Using Commands 25

File Commands 25

The Dir Command 25

The Copy Command 27

The Del Command 28

The Rename Command 29

The Type Command 29

The Print Command 30

Disk Commands 31

The Format Command 31

The Diskcopy Command 33

5 Using Applications with MS-DOS 35

How to Run Application Programs 35

How to Create a File with Edlin 36

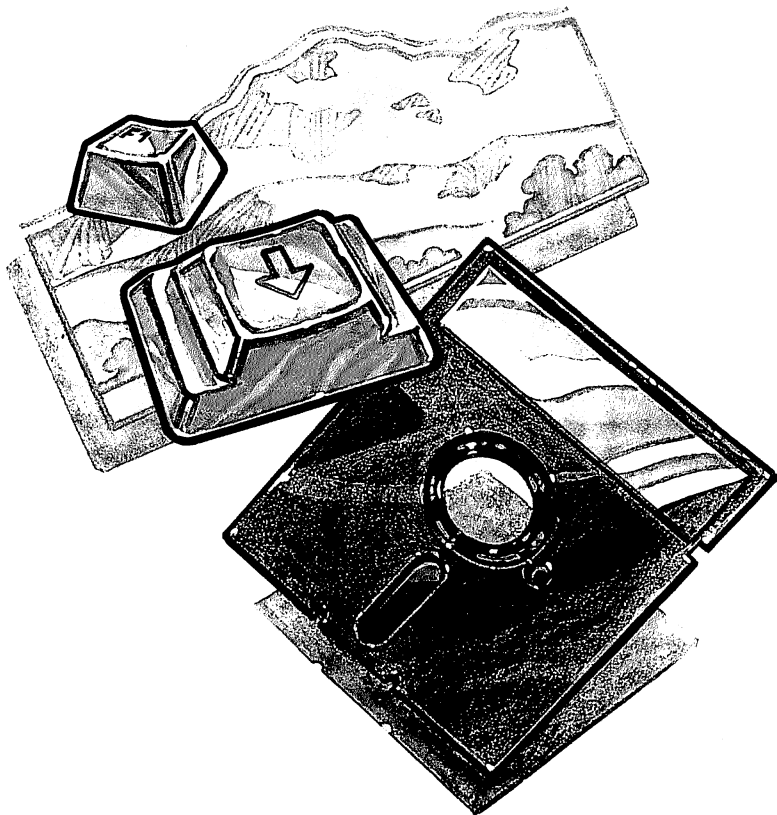
Summary 38

Terms 39

Index 43

Welcome

Welcome to the MS-DOS® operating system, version 3.2. What you have before you are two manuals: the *MS-DOS User's Guide* and *MS-DOS User's Reference*. If you are learning MS-DOS for the first time, or if you have used it before, these two manuals will introduce you to and explain both basic and advanced features of the MS-DOS operating system. Go at your own pace, though; you're not expected to read every word, or know every feature. But the more you learn about MS-DOS the easier you'll find it to use.



Introducing MS-DOS

What is MS-DOS?

The Microsoft® MS-DOS *operating system* is like a translator between you and your computer. Basically, this translator is a series of programs, making up what is called the MS-DOS operating system. The programs in this operating system let you communicate with your computer, your disk drives, and your printer, letting you manage these resources to your advantage.

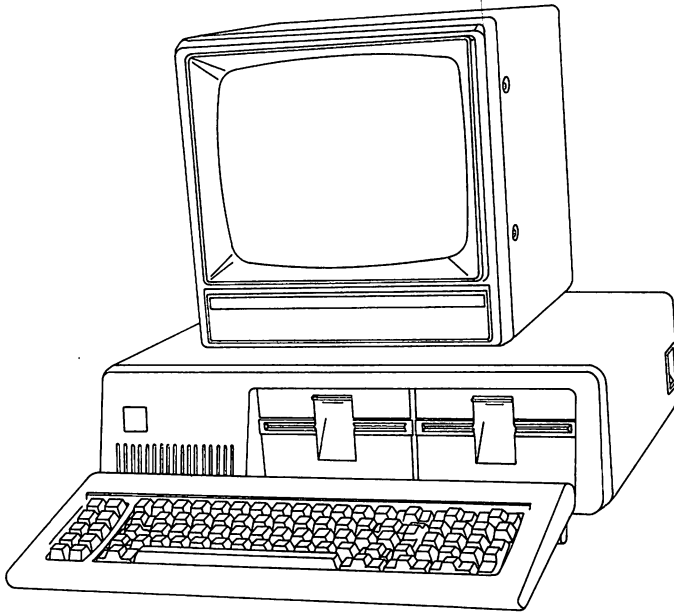
MS-DOS is more than just a translator, though. It is also a *disk* operating system. This means that you can use MS-DOS programs on computers with floppy disks or hard disks. Once you have loaded MS-DOS into your computer's memory, you can compose letters and reports, run programs and languages such as Microsoft GW-BASIC®, and use devices such as printers and disk drives.

System requirements

Before You Begin

Before you begin using this book you should have:

- A 16-bit personal computer that runs MS-DOS and has at least 256K bytes of memory.
- Two disk drives (either two floppy disk drives, or one hard disk drive and one floppy disk drive).



- An MS-DOS master disk.
- The following manuals:

MS-DOS User's Guide
MS-DOS User's
Reference

MS-DOS
Programmer's
Reference

Introduces you to the basics of MS-DOS.
Presents advanced features of MS-DOS
with a detailed description of MS-DOS
commands.

Provides detailed information necessary
for programming with MS-DOS.

How to Use This Guide

This manual introduces you to the MS-DOS operating system and teaches you how to use several MS-DOS features. The manual is organized so that you can easily find what you need to know, as in the following list, which gives you a quick overview of the topics covered.

Turn to	To learn
Chapter 1	About your keyboard
Chapter 2	About disks and files
Chapter 3	How to start MS-DOS How to quit MS-DOS
Chapter 4	How to use MS-DOS commands How to print a file
Chapter 5	How to run a program How to create a file
Terms	About MS-DOS terminology

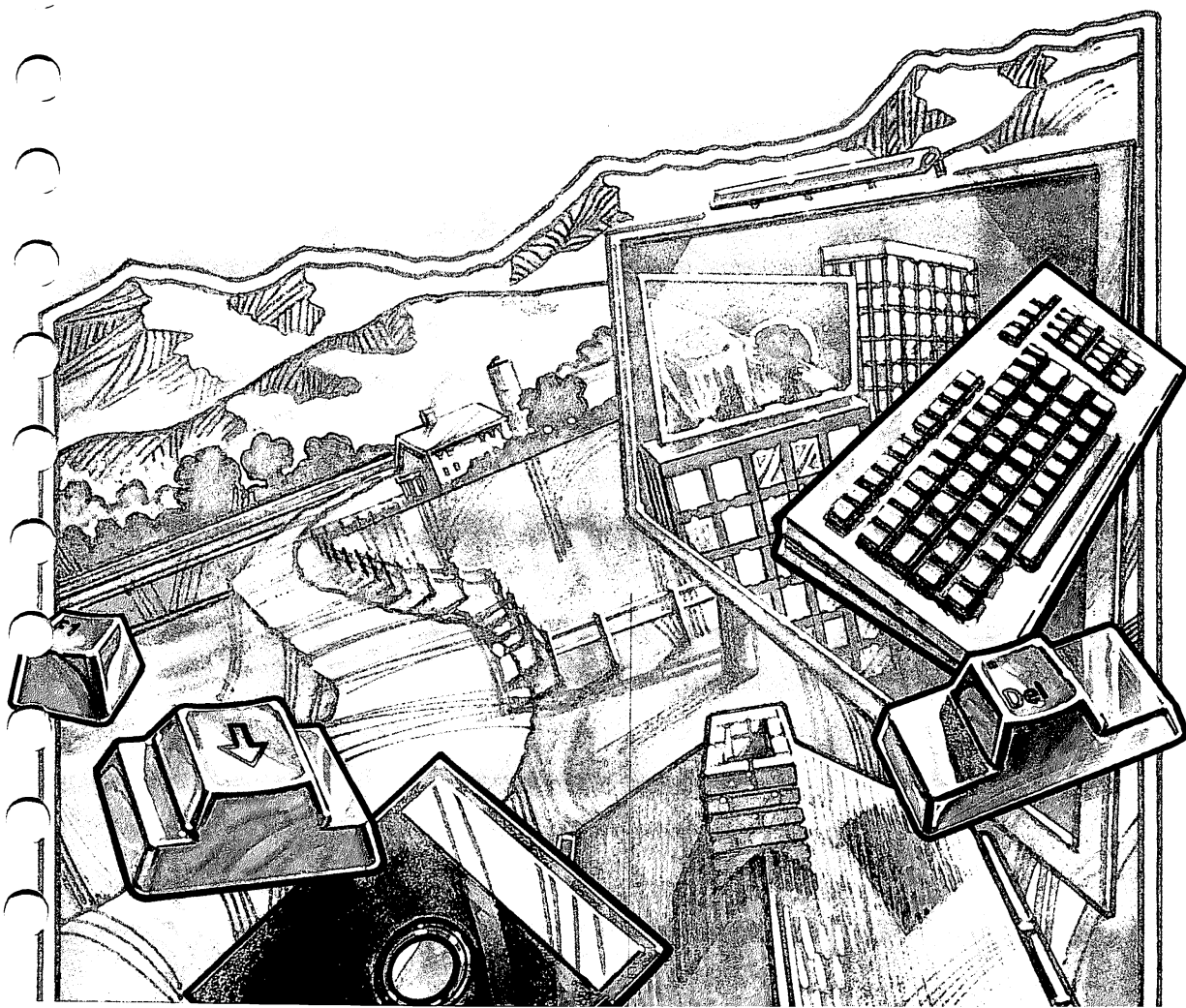


To learn more about MS-DOS refer to the *MS-DOS User's Reference*.

1 Learning About MS-DOS

In this chapter you will learn about:

- Important MS-DOS terms
- The MS-DOS keyboard



Introducing MS-DOS terms

Terms You Should Know

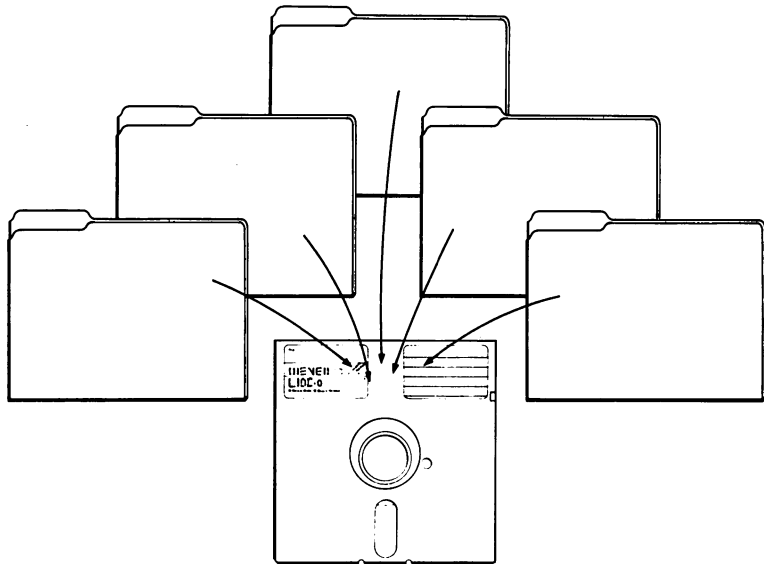
When you are introduced to a new or different idea, you must often learn a new set of words to understand the idea. The MS-DOS operating system is no exception. The following pages explain some terms you will need to know so that you can read and use this manual.

Program

Programs, often called application programs, applications, or software, are series of instructions written in computer languages. These instructions are stored in files and tell your computer to perform a task. For example, a program might tell your computer to alphabetically sort a list of names.

File

A file is a collection of related information, like the contents of a file folder in a desk drawer. File folders, for instance, might contain business letters, office memos, or monthly sales data. Files on your disks could also contain letters, memos, or data.



Filename

Just as each folder in a file cabinet has a label, each file on a disk has a name. This name has two parts: a *filename* and an *extension*. A filename can be from 1 to 8 characters in length, and can be typed in uppercase or lowercase letters. MS-DOS automatically converts filenames to uppercase letters.

Filename extensions consist of a period followed by one, two, or three characters. Extensions are optional, but it's a good idea to use them, since they are useful for describing the contents of a file to you and to MS-DOS. For instance, if you want to be able to quickly identify your report files, you can add the filename extension *.rpt* to each one. Here's an example of a filename with this extension:

```
progress.rpt
|           |
|filename   |filename extension
```

Directory

A directory is a table of contents for a disk. It contains the names of your files, their sizes, and the dates they were last modified.

When you look at the directory on your MS-DOS master disk, you will see many files with the extension *.exe* or *.com*. The extension *.exe* means *executable*, and *.com* means *command*. These extensions tell MS-DOS that the files are programs that can be run. Many files will have other kinds of extensions, such as *.doc* and *.txt*, which might contain text. Another common program file extension is *.bas* for BASIC programs.

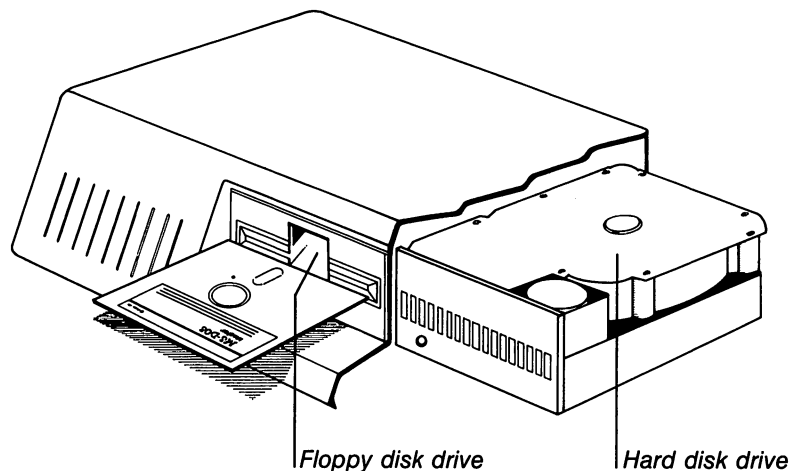
Volume Label

When you use a new disk, you can put a label on the outside of it to help you identify its contents. You can also give each of your disks an internal name, called a *volume label*.

You can look at the volume label on a disk by displaying its directory. Some programs may look at the volume label to see if you are using the correct disk. So make sure that you label your disks.

Disk Drive

Floppy disk drives are commonly referred to as the A drive and the B drive; a hard disk is usually referred to as the C drive. Check your computer manual to see which drive is A and which is B (or C).



Drive Name

A complete *drive name* consists of a *drive letter* and a *colon*. When using a command, you may need to type a drive name before your filename to tell MS-DOS where to find the disk that contains your file. For example, suppose you have a file named *finances.doc* on the disk in drive B. To tell MS-DOS where to find this file you would type the drive name before the filename:

```
b:finances.doc
```

drive name filename with extension

If you don't specify a drive name when you type a filename, MS-DOS automatically searches for the file on the disk in the *default drive*. To let you know that it is ready to receive a command, MS-DOS displays a prompt that contains the default drive letter followed by a greater-than sign. Following the prompt on your screen is the cursor, the small box or flashing underline that shows where the next character you type will appear.

Example

```
A>_
```

cursor

MS-DOS prompt

So when your prompt is A>, MS-DOS searches only the disk in drive A for files and programs — unless you tell it to search in another drive.

If you will be working primarily with files on drive B, it is easier to change the default drive to B, so that you won't have to type the letter b, followed by a colon, with *every* command and filename.

Command

Just as you will run programs to create and update files containing your data, you will also need to run some special programs, called MS-DOS commands, that let you manipulate entire files.

When you give MS-DOS certain commands, you are asking the computer to perform tasks. For example, when you use the **diskcopy** command to copy your MS-DOS master disk, you are running a program named **diskcopy** on the MS-DOS disk.

Other MS-DOS commands:

- Compare, copy, display, delete, and rename files
- Copy, format, and label disks
- Run your programs, as well as those supplied with MS-DOS such as **edlin**
- List directories
- Enter the date and time
- Set printer and screen options

Chapter 4, "Using Commands," contains more information on MS-DOS commands. In addition, the *MS-DOS User's Reference* contains detailed descriptions for each MS-DOS command.



Keys You Use with MS-DOS

The MS-DOS keyboard

Now that you've learned about MS-DOS commands, files, drives, etc., next you'll learn about the keys you will be using.

In addition to the keys you'd find on a typewriter, your computer keyboard has some keys that have special meanings to MS-DOS.

First, note that there are two important differences between a typewriter keyboard and a computer keyboard:

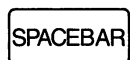


A computer understands the difference between a *one* and a lowercase *L*. Be sure you don't type a lowercase *L* when you mean a *one*.

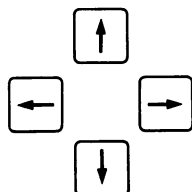


Capital O and zero may *look* alike, but they have different meanings to a computer. Many computers display a zero with a diagonal line (0) through it. Make sure you type the correct letter or number when you give commands to MS-DOS.

Special Keys



The SPACEBAR moves the cursor to the right. But to move the cursor along a line *without deleting any characters*, use the direction keys.



Direction keys move the cursor right, left, up, and down. They do not affect the characters that are displayed. Some programs ignore these keys or do not use them. In these manuals, the direction keys are also referred to as the RIGHT, LEFT, UP, and DOWN arrow keys.

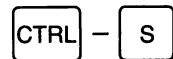


To correct typing mistakes on the current line, you use the BACKSPACE key (abbreviated as BKSP). This key deletes characters as it moves the cursor to the left.

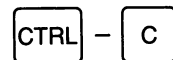
Control Key Combinations



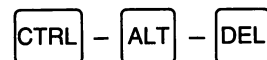
The CONTROL key (abbreviated as CTRL) has a special task. It lets you give complex commands to your computer by pressing only two or three keys. You must hold down the CONTROL key while you press another key. That is, you use the CONTROL key as you would the SHIFT key.



When you press the CONTROL key and the s key at the same time, you can stop the scrolling of the screen display. Then to continue scrolling, press CONTROL-S again.



When you press the CONTROL key and the c key at the same time, you can stop a command.



If you want to start (or restart) MS-DOS, press the CONTROL, ALT, and DEL keys at the same time.

The Return Key

RETURN

Press the RETURN key after you type commands. When you press the RETURN key, MS-DOS performs the command.

Now that you've seen what the keys on your keyboard can do and learned some of the MS-DOS vocabulary, you're ready to go on to Chapter 2, "Learning About Disks, Files, and Directories." There you'll learn about floppy disks and hard disks, about formatting your disks, and about naming your files.

What's next?



2 Learning About Disks, Files, and Directories

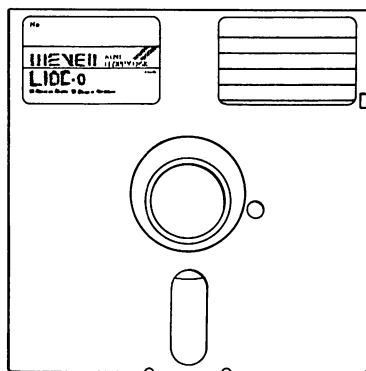
In this chapter you will learn about:

- Floppy disks
- Hard disks
- Filenames
- Directories

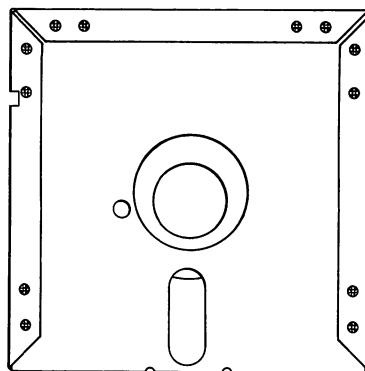
About floppy disks

Floppy Disks

A floppy disk is a magnetized disk that can store up to 400 single-spaced pages of text. Every floppy disk is enclosed in its own protective cover. The front of this cover is smooth, while the back has visible seams. You should always place labels on the front of the cover, at the top, so that the label doesn't touch the magnetic surface of the disk. It's also a good idea to use a felt-tip pen when writing on labels—a pencil or ballpoint pen can damage the disk if you press too hard.

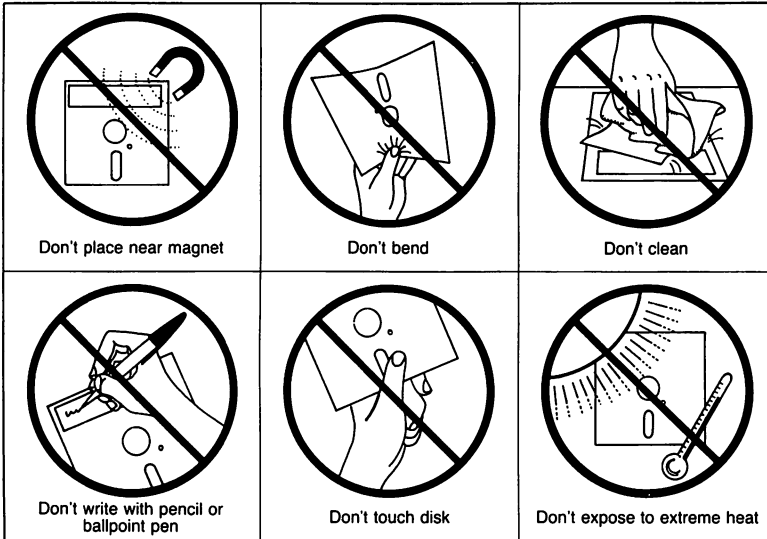
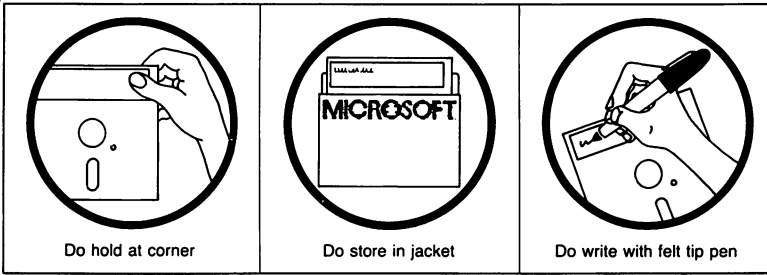


Front



Back

You should store floppy disks in a safe place, away from dust, moisture, magnetism, and extreme temperatures. Be sure to label each disk you use, since labels help you identify what files are on the disk and remind you that the disk has information stored on it.



Disk Protection

Labels do help you keep track of your disks, but you may also need to protect your disks. Some floppy disks are protected, letting you *examine* information on them without letting you *change* anything. These are called *write-protected* disks. They usually have a small piece of tape, called a *tab*, covering a notch on the right side of the disk. You can copy information onto a write-protected disk by first removing the write-protect tab; however, you should consider why the disk was protected — *before* you change its contents. After you have copied or changed a write-protected disk it's always a good idea to replace the write-protect tab.

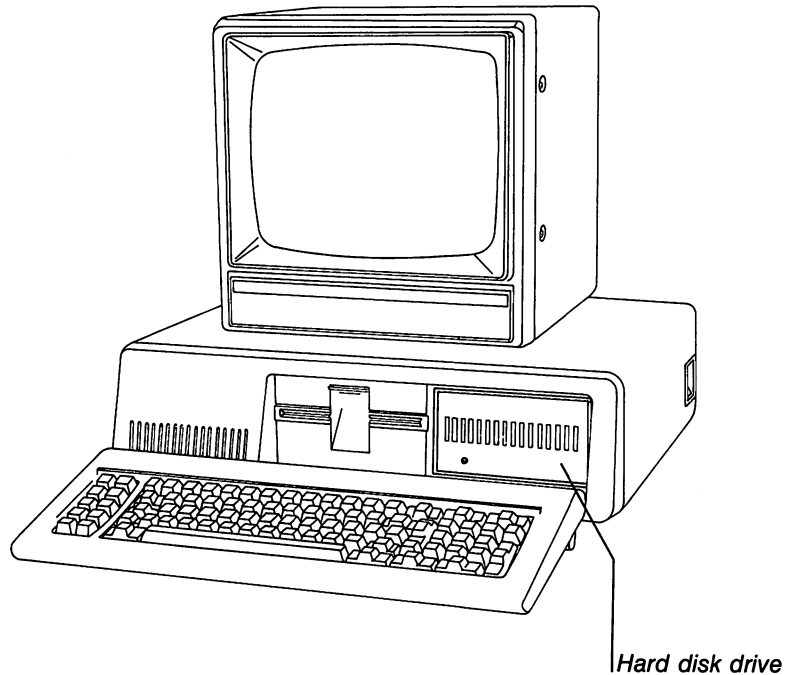
Protecting your disks

If a disk does not have a write-protect notch, it is always write-protected. Many application programs, including this version of MS-DOS, come on write-protected disks that protect the files from being destroyed accidentally.

About hard disks

Hard Disks

In addition to floppy disks, some computers use a hard disk, which can store much more information than a floppy disk. Computers also take less time to find information stored on a hard disk than on a floppy disk. A hard disk is usually built into the computer. Such a system might look like this:



When you store application programs, including MS-DOS, on your hard disk, you should keep a backup copy of the programs on a floppy disk in case the information on the hard disk is accidentally damaged or destroyed. (For more information about making a backup copy of your MS-DOS disk see Chapter 3, "Getting Started.")

The Format Command

Formatting your disks

Before you can use your new disks for storing information, you must *format* them. You do this with the **format** command, a special program that structures a disk so that MS-DOS can find information on it. The **format** command also checks the disk for defective spots.

You can format both floppy and hard disks. But remember that if a disk is not blank, formatting it destroys any data already on the disk.

You will learn more about the **format** command in Chapter 4, "Using Commands."

How to Name Your Files

Naming your files

When naming a file you may have trouble finding a name that uniquely identifies the file's contents. Dates, for example, are often used in filenames; however, they take up several characters, leaving you with little flexibility. Other common names for files are words like *budget*, *finances*, *analysis*, *report*, etc. These kinds of filenames identify the contents, but leave little room for dates. So the secret is to find a compromise—a point where you can combine a date with a word, creating a unique filename.

The name of a typical MS-DOS file (see Chapter 1) looks like this:

```
customer.lst
|          |
|filename  |filename extension
```

Notice that the filename was typed in lowercase letters. You can type filenames in uppercase or lowercase letters, even though MS-DOS converts them into uppercase letters. Some more examples of filenames are:

```
budget.86
takeover.bid
june86
finances.doc
schedule.may
```

Many of your filenames will contain only letters and numbers. But you may also use any of the following symbols (and letters) in your filenames and extensions:

A-Z a-z 0-9 \$ % ' - @ { } ~ ! #

Valid filename characters

Warning Some applications may not let you use all of these symbols. If in doubt use only letters and numbers.

Avoiding invalid filenames

Invalid Filenames

Although you do have some freedom when naming your files, there are certain names that you may not use, because MS-DOS reserves them for specific devices that your computer uses. These invalid names are: *aux*, *clock\$*, *com*, *con*, *keybd\$*, *lpt*, *lst*, *nul*, *prn*, and *scrn\$*. You may use these names as extensions, but remember not to use them to name your files.

The MS-DOS directory

Directories

The names of your files are kept in a directory on each disk. The directory also contains information on the sizes of the files, and the dates they were created and updated.

If you want to know what files are on your disk, you can use the **dir** command. This command tells MS-DOS to display all the files in a specific directory on a disk. For example, if your MS-DOS disk is in drive A and you use the **dir** command, the directory display would look similar to this:

```
- Volume in drive A is DOS 3-2
  Directory of A:\
```

COMMAND	COM	23612	3-21-86	12:00p
ANSI	SYS	1651	3-21-86	12:00p
ATTRIB	EXE	8234	3-21-86	12:00p
BACKUP	EXE	22906	3-21-86	12:00p
CHKDSK	EXE	9680	3-21-86	12:00p
FC	EXE	14446	3-21-86	12:00p
DISKCOPY	EXE	3936	3-21-86	12:00p
DRIVER	SYS	1102	3-21-86	12:00p
EDLIN	EXE	7356	3-21-86	12:00p
FDISK	EXE	16444	3-21-86	12:00p
FIND	EXE	6403	3-21-86	12:00p
GRAFTABL	EXE	8210	3-21-86	12:00p
GRAPHICS	EXE	13170	3-21-86	12:00p
JOIN	EXE	8942	3-21-86	12:00p
KEYBDV	EXE	2850	3-21-86	12:00p
KEYBFR	EXE	2912	3-21-86	12:00p
KEYBGR	EXE	2904	3-21-86	12:00p
KEYBIT	EXE	2856	3-21-86	12:00p
KEYBSP	EXE	2947	3-21-86	12:00p

KEYBUK	EXE	2850	3-21-86	12:00p
LABEL	EXE	2750	3-21-86	12:00p
MODE	EXE	13652	3-21-86	12:00p
RECOVER	EXE	4145	3-21-86	12:00p
REPLACE	EXE	4852	3-21-86	12:00p
RESTORE	EXE	21360	3-21-86	12:00p
SHARE	EXE	8544	3-21-86	12:00p
SUBST	EXE	9898	3-21-86	12:00p
TREE	EXE	8556	3-21-86	12:00p
RAMDRIVE	SYS	6454	3-21-86	12:00p
XCOPY	EXE	5396	3-21-86	12:00p
DISKCOMP	EXE	3808	3-21-86	12:00p
ASSIGN	COM	1523	3-21-86	12:00p
MDRE	COM	282	3-21-86	12:00p
PRINT	EXE	8824	3-21-86	12:00p
SORT	EXE	1898	3-21-86	12:00p
FORMAT	EXE	10973	3-21-86	12:00p
SYS	COM	4607	3-21-86	12:00p
37 File(s) 17408 bytes free				

Note The file sizes and dates you see on your screen may differ from the ones shown here, depending on your version of MS-DOS. Don't worry, though. Such variations do not affect the way you use MS-DOS or the way MS-DOS responds to your commands.

You can also get information about any file on your disk by typing the **dir** command followed by a filename. For example, to display directory information for a file named *schedule*, you could use the following command:

```
dir schedule
```

MS-DOS would respond by displaying the filename *schedule* followed by the file's size in bytes and the date and time it was last changed.

So far you have learned the basic background information that you need in order to use the MS-DOS operating system. In the final three chapters of this guide you'll learn to make your computer work for you, while you build a working knowledge of MS-DOS.

What's next



3 Getting Started

In this chapter you will learn:

- How to start MS-DOS
- How to quit MS-DOS
- How to make a backup copy of your MS-DOS disk
- What to do if you have a hard disk
- What to do if you have only one floppy disk drive

How to Start MS-DOS

The first two chapters in this manual introduced you to the fundamentals of MS-DOS. Now it's time to put your new knowledge to the test. You'll start by loading MS-DOS into your computer's memory.

To start MS-DOS, just follow these steps (these steps work for computers that have either hard disks or floppy disks):

- 1** First, make sure your computer is turned off.
- 2** Take the MS-DOS master disk out of the protective jacket.
- 3** Insert this disk into drive A. (Refer to your computer manual for the correct drive.)
- 4** Close the disk drive door.
- 5** Turn on the power for your monitor and your computer.

The light on the disk drive should glow, and you should hear some whirring noises as your computer "reads" the disk. You should then see the following on your screen:

```
Current date is Tue 1-01-1980
Enter new date (mm-dd-yy):
```

MS-DOS asks you to provide the date.

Starting MS-DOS

Setting the date and time

- 1 Type the date. For example, if the date is July 6, 1986, you simply type the following command, then press the RETURN key:

07-06-86

If the date is already correct, or you do not want to answer this prompt, press the RETURN key to move to the next step.

- 2 Type the time according to a 24-hour clock. For example, if it is 1:30 P.M., type the following, then press the RETURN key:

13:30

If the time is already correct, or you do not want to answer this prompt, press the RETURN key.

MS-DOS does not accept your command until you press the RETURN key.

Note If you make a mistake when you are typing the date or time, simply backspace over the mistake and retype. As you use the BACKSPACE key, you will notice that the characters disappear. If you make a mistake and have already pressed the RETURN key, press the CONTROL-ALT-DEL keys simultaneously to restart MS-DOS and try again.

Your screen should look something like this (your time and date may be different, depending on what you typed in steps 1 and 2):

```
Current date is Tue 1-01-1980
Enter new date (mm-dd-yy): 07-06-86
Current time is 0:00:45:10
Enter new time: 13:30
```

```
Microsoft(R) MS-DOS(R) Version 3.20
      (c)Copyright Microsoft Corp 1981-1986
```

A>_

Though other letters are often used, the A> is the standard MS-DOS prompt. When you see the A> prompt, MS-DOS is ready for instructions from you.

Before you start giving these instructions, however, you might like to know how to quit MS-DOS.

How to Quit

There is no “quit” command in MS-DOS, but you can end your MS-DOS session easily by following these steps:

- 1** Make sure that your last command is finished. You should see the MS-DOS prompt (usually A>) on the screen.
- 2** Remove the floppy disks from the drives; put them back in their protective jackets; and store them in a safe place, away from dust, moisture, and magnetism.
- 3** Turn off your computer.
- 4** Turn off your monitor.

Ending your MS-DOS session

How to Make a Backup Copy of Your MS-DOS Disk

In this section you'll learn how to make a backup copy of your MS-DOS disk if you have two floppy disk drives. If you have a hard disk, read the section “If You Have a Hard Disk.” If you have only one floppy disk drive, read this section, then the section “If You Have Only One Floppy Disk Drive” at the end of this chapter.

MS-DOS comes with a program named *diskcopy.exe* that lets you copy the contents of disks. You need not format your blank disks before you use the **diskcopy** command.

Making a backup copy of your MS-DOS master disk is easy:

- 1** Start MS-DOS with the MS-DOS master disk in drive A.
- 2** Make sure that a blank disk is in drive B.
- 3** At the MS-DOS prompt, type the following:

```
diskcopy a: b:
```

- 4** Press the RETURN key.

Your screen should look like this:

```
A> diskcopy a: b:
```

```
Insert SOURCE diskette in drive A:
```

```
Insert TARGET diskette in drive B:
```

```
Press any key when ready . . .
```

Making a backup copy

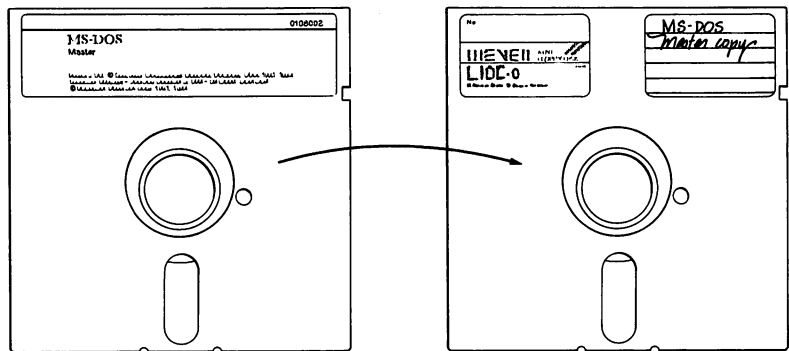
Note If you have only one disk drive, MS-DOS prompts you to insert the drive A disk. Refer to the section “If You Have Only One Floppy Disk Drive” for more information.

- 5 Press the SPACEBAR to start the **diskcopy** program.
When the **diskcopy** program is complete, MS-DOS asks:

Copy another? (Y/N)

- 6 Press N (for No) to end the **diskcopy** program.

You now have two MS-DOS disks: the MS-DOS master disk and the copy you have just made.



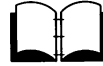
Label the new disk, and put your MS-DOS master disk in a safe place, away from dust, moisture, and magnetism. If anything should happen to the copy you have just made, you'll have to use the master disk to make another copy.

Note Always use your backup copy of the MS-DOS master disk. Keep the master disk in a safe place.

If You Have a Hard Disk

If your computer has a hard disk, you should copy all the files from the MS-DOS master disk onto the hard disk. Then each time you start MS-DOS you won't need to use a floppy disk; instead, you'll be able to start MS-DOS directly from the hard disk.

But before you can copy the MS-DOS files onto your hard disk, you may need to configure it first. To find out whether you need to do this, see Appendix F, "Configuring Your Hard Disk," in the *MS-DOS User's Reference*. When you have copied the MS-DOS files onto your hard disk, the original floppy disk will then be your backup copy.



Warning Whenever you format a disk, you destroy its contents, if there were any. It's a good idea to copy any files from your hard disk onto floppy disks before you format the hard disk. (To learn how to copy files, see Chapter 4, "Using Commands.") Once you have formatted your hard disk you should *never* have to format it again.

The following example assumes that your hard disk is named drive C. Refer to the documentation that came with your computer to find out the name of your hard disk; then follow these steps to format it:

Formatting your hard disk

- 1 Start MS-DOS and type the date and time.
- 2 Be sure that your MS-DOS master disk is in drive A.
- 3 At the MS-DOS prompt, type the following command:

```
format c: /v /s
```

- 4 Press RETURN. MS-DOS formats the disk in drive C.
When the format process is complete, MS-DOS displays the following prompt:

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

- 5 Type the name that you want to use to identify the hard disk (for example, *HARD DISK*), and press the RETURN key.

```
MS-DOS asks:
```

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

- 6 Press N (for No) to end the **format** program.

Copying files onto a hard disk

To copy files onto your newly formatted hard disk, you must use the **copy** command. This command is automatically loaded into your computer's memory when you start MS-DOS.

To copy your MS-DOS master disk onto a hard disk (drive C), follow these steps:

- 1 Make sure that the MS-DOS master disk is in drive A.
- 2 At the MS-DOS prompt, type the following command:

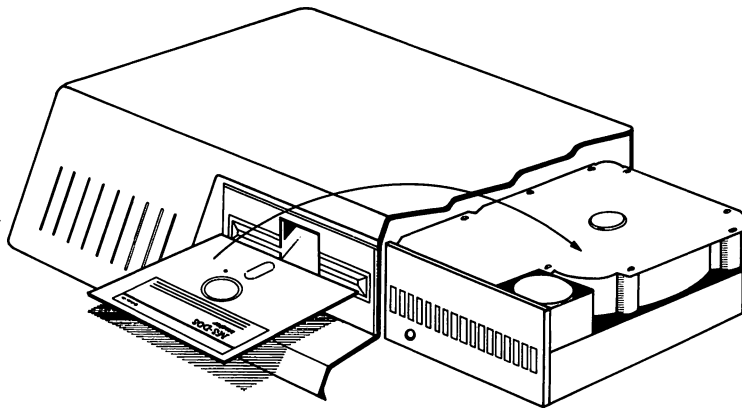
```
copy a:*. * c:
```

This command tells MS-DOS to copy all files on drive A to drive C.

- 3 Press the RETURN key.

The **copy** program then lists each file on the screen as it is copied onto the new disk. When the process is complete, MS-DOS shows you how many files it has copied.

You now have two MS-DOS disks: the MS-DOS master disk and the copy you have just made on your hard disk.



Master disk copied to hard disk

Now, put the master disk in a safe place, away from dust, moisture, and magnetism. You can use this disk to make another copy if anything happens to your hard disk.

If You Have Only One Floppy Disk Drive

Single drive computers

If your computer has only one floppy disk drive, you can still use MS-DOS commands as you would on a system with more than one drive, but you must specify a drive name as well. By specifying the drive letter you tell your computer to perform the command on that drive. MS-DOS then prompts you to insert the proper disk, as in the following example:

```
A> format a: /v
Insert new diskette for drive A:
and strike ENTER when ready_
```

If you specify drive B in a command when you have only one drive, MS-DOS prompts you to insert the disk for drive B. In this case A and B represent the disks that you put into the single drive.

For example, to make a copy of your MS-DOS disk, type the following at the MS-DOS prompt:

```
diskcopy a: b:
```

MS-DOS responds with the following message:

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

Remove the MS-DOS disk, put the blank disk into the drive, and press any key. You may need to reinsert the disks for drives A and B several times to complete the copy process.

Note The letter in the system prompt represents the default drive; it does *not* represent the last disk used.

You now know how to start and quit the MS-DOS operating system, and you've used some simple commands to copy and format your disks. In the next chapter you will learn to use more MS-DOS commands. As you read about these commands and start to use them, you'll begin to understand how the MS-DOS operating system works, and you'll begin to see what this system can do for you.

What's next?



4 Using Commands

In this chapter you will learn:

- How to use file commands
- How to print files
- How to use disk commands

File Commands

Using file commands

You can use several MS-DOS commands to manage your files. Some of the more common commands are **dir**, **copy**, **del**, **rename**, and **print**.

Note The following examples assume that drive A is the default drive.

The Dir Command

If you want to find out what files are on a disk, you can list its directory by using the MS-DOS **dir** command. For example, to display the directory of the disk in drive B, you would use the following command:

`dir b:`
*Show me the ...of the disk in drive B.
directory...*

If you use the **dir** command without a drive letter, MS-DOS lists the directory of the disk in the default drive.

Example

Suppose you want to see how many files are in the directory of the MS-DOS disk in drive A. To display this directory you would simply follow these steps:

Listing the MS-DOS directory

- 1 Make sure the MS-DOS disk is in drive A.
- 2 At the MS-DOS prompt, type the following command:

```
dir
```

- 3 Press the RETURN key. If necessary, you can press CONTROL-S to stop the list from scrolling. To view the rest of the display, you simply press CONTROL-S again.

Your screen should look similar to this:

```
Volume in drive A is DOS 3-2
Directory of A:\
```

COMMAND	COM	23612	3-21-86	12:00p
ANSI	SYS	1651	3-21-86	12:00p
ATTRIB	EXE	8234	3-21-86	12:00p
BACKUP	EXE	22906	3-21-86	12:00p
CHKDSK	EXE	9680	3-21-86	12:00p
FC	EXE	14446	3-21-86	12:00p
DISKCOPY	EXE	3936	3-21-86	12:00p
DRIVER	SYS	1102	3-21-86	12:00p
EDLIN	EXE	7356	3-21-86	12:00p
FDISK	EXE	16444	3-21-86	12:00p
FIND	EXE	6403	3-21-86	12:00p
GRAFTABL	EXE	8210	3-21-86	12:00p
GRAPHICS	EXE	13170	3-21-86	12:00p
JOIN	EXE	8942	3-21-86	12:00p
KEYBDV	EXE	2850	3-21-86	12:00p
KEYBFR	EXE	2912	3-21-86	12:00p
KEYBGR	EXE	2904	3-21-86	12:00p
KEYBIT	EXE	2856	3-21-86	12:00p
KEYBSP	EXE	2947	3-21-86	12:00p
KEYBUK	EXE	2850	3-21-86	12:00p
LABEL	EXE	2750	3-21-86	12:00p
MODE	EXE	13652	3-21-86	12:00p
RECOVER	EXE	4145	3-21-86	12:00p
REPLACE	EXE	4852	3-21-86	12:00p
RESTORE	EXE	21360	3-21-86	12:00p
SHARE	EXE	8544	3-21-86	12:00p
SUBST	EXE	9898	3-21-86	12:00p
TREE	EXE	8556	3-21-86	12:00p
RAMDRIVE	SYS	6454	3-21-86	12:00p
XCOPY	EXE	5396	3-21-86	12:00p
DISKCOMP	EXE	3808	3-21-86	12:00p
ASSIGN	COM	1523	3-21-86	12:00p
MORE	COM	282	3-21-86	12:00p
PRINT	EXE	8824	3-21-86	12:00p
SDRT	EXE	1898	3-21-86	12:00p
FORMAT	EXE	10973	3-21-86	12:00p
SYS	COM	4607	3-21-86	12:00p

37 File(s) 17408 bytes free

Note The file sizes and dates you see on your screen may differ from the ones shown here, depending on your version of MS-DOS. Don't worry, though. Such variations do not affect the way you use MS-DOS or the way MS-DOS responds to your commands.

Note The filenames used in the following examples are for illustrative purposes only — to use these commands, you would substitute the name of a file on the default disk.

The Copy Command

If you need to copy files, you can use the **copy** command to copy one or more files, either on the same disk or from one disk to another. For instance, suppose you need a copy of a file named *sales.doc* that you have on a disk in drive A, and suppose you want to call this new copy *monthly.rpt*:

Example

To copy the *sales.doc* file and call the new copy *monthly.rpt* you would just follow these steps:

- ❶ Make sure that the disk with the *sales.doc* file is in drive A and that A is the default drive.
- ❷ At the MS-DOS prompt, type the following command:


```
copy sales.doc monthly.rpt
```
- ❸ Press the RETURN key.

When you make a copy of a file, you cannot give the new file the same name as the original file. You can, however, copy a file from one disk to another and keep the same filename. For example, to copy a file from the disk in drive A to the disk in drive B, use the following command:

```
copy a:sales.doc b:sales.doc
```

Make a copy
of a file...

...named "sales.doc"
on drive A.

Name the copy of the
file "sales.doc" as well,...

...and put it on the disk in drive B.

Note In the above example, if A is the default drive (that is, if the prompt is A>), you needn't type the letter a, followed by a colon, before the first filename. Also, by default, the copy will have the name of the original file if you do not specify a new name. For example, the following commands all produce the same result:

Copying a file

```
copy a:sales.doc b:sales.doc
copy sales.doc b:sales.doc
copy sales.doc b:
```

The Del Command

Just as you may need to make copies of files, you may also need to remove old or unnecessary files to clean up your file system. So when you want to erase a file from a disk, you can use the MS-DOS **del** command. Remember, though, that the **del** command *permanently* erases the file. To delete an old *sales.doc* file from the disk in drive B, at the MS-DOS prompt you would use the following command:

```
del b:sales.doc
```

Delete a file... |...named "sales.doc" from the disk in drive B.

Example

Deleting a file

Suppose you have an old copy of the *sales.doc* file that you no longer need. To delete this file from the disk in the default drive, you would just follow these steps:

- 1 Make sure that the disk with the *sales.doc* file is in the default drive.
- 2 At the MS-DOS prompt, type the following command:

```
del sales.doc
```

- 3 Press the RETURN key. MS-DOS then deletes the *sales.doc* file from the disk.

Note

- To MS-DOS, the wildcard name *.* means *all files in a directory*, so be careful when you use this abbreviation in your commands. For example, if you type the **del** command, followed by *.* , MS-DOS asks if you want to delete *all* the files in the directory; if you then press Y, (for Yes) MS-DOS *permanently* deletes the files in the directory.
- The **del** command does not work if you type the word "delete." You can, however, type "erase" instead of "del."

The Rename Command

Occasionally, you may want to change the name of a file. For example, suppose you have a file named *monthly.rpt* on a disk. When you add other monthly reports to your disk, you may want to change the name to something more specific. To change the name to *annual.rpt*, for instance, you would use the following command:

```
rename monthly.rpt annual.rpt
```

Change the name of a file...	...from "monthly.rpt"...	...to "annual.rpt."
---------------------------------	-----------------------------	---------------------

You can only rename files on the same disk, so you *cannot* change *a:monthly.rpt* to *b:monthly.rpt*.

Example

Suppose you want to rename a file named *payroll.doc*, on the disk in the default drive, to *salary.doc*. You would simply follow these steps:

- 1 Make sure that the disk with the *payroll.doc* file is on the disk in the default drive (A).
- 2 At the MS-DOS prompt, type the following command:

```
rename payroll.doc salary.doc
```

- 3 Press the RETURN key.

Note The **rename** command can be abbreviated to **ren**.

The Type Command

If you want MS-DOS to display a file that contains text (often called a text file) on the screen, use the **type** command. For example, say you have created a file named *phone.lst* on the disk in drive A, and you want to check one of the phone numbers. To display the file on the screen, you would use the following command:

```
type a:phone.lst
```

Display on the screen...	...the file named "phone.lst" that is on the disk in drive A.
-----------------------------	--

Renaming a file

Displaying a file

Example

Suppose you want to check your employees' salary figures. So you decide to look at a file named *salary.doc* that is on the disk in the default drive. To display the *salary.doc* file you would just follow these steps:

- 1 Make sure that the disk with the *salary.doc* file is in the default drive (A).
- 2 At the MS-DOS prompt, type the following command:

 type salary.doc
- 3 Press the RETURN key.

MS-DOS then displays the *salary.doc* file on the screen.

Helpful Hints If the file is too long to fit on the screen, remember that you can press CONTROL-S to prevent it from scrolling off the screen. When you press CONTROL-S again, the file will resume scrolling.

MS-DOS displays only text files on the screen. So if you try to display a program file (one with an extension of *.com* or *.exe*), you will see only strange symbols on the screen.

If you have an application program that creates files, you may need to run the application to view them.

The Print Command

If you have a printer attached to your computer, you can print files with the MS-DOS **print** command. Assume, for example, that you have a file named *invest.mnt* and want to print it on your printer. You could use the following command:

```
print invest.mnt
```

Print a file (MS-DOS assumes this file is on the disk in the default drive.)...
 ...named "invest.mnt."

Example

Say you have a file that contains a list of investors and their phone numbers, and suppose you want to print this file and keep it near your phone. The file is named *invest.mnt* and is on the disk in drive B. Drive A is the default drive (A> is the prompt). To print the *invest.mnt* file, you would just follow these steps:

Printing a file

- 1 Make sure that the MS-DOS disk is in drive A.
- 2 Make sure that the disk with the *invest.mnt* file is in drive B.
- 3 Check to see that your printer is on and has paper.
- 4 At the MS-DOS prompt, type the following command:

```
print b:invest.mnt
```

- 5 Press the RETURN key.
- 6 MS-DOS prompts you for the name of the printing device connected to your computer. (This name is usually the communications port that the printer cable connects to.) Just type the name, or press the RETURN key to print to the default printer.

If the master disk is not in drive A, MS-DOS prompts you to insert it in the drive.

Hint While a file is being printed, you can type other commands to MS-DOS. You can even run other programs or create and modify files. But since printing a file takes a lot of your computer's resources, your tasks may take longer if you try to do them while you are printing a file. So if you have a long file to print, you might schedule the printing for when you plan to be away from your computer.

Disk Commands

Using disk commands

This section presents two commands that you use for disks: **format** and **diskcopy**.

The Format Command

When you purchase new disks they are blank and unformatted, so you must format them before MS-DOS can use them. Formatting structures a disk so that MS-DOS can find and store information on it. It also checks the disk for defective spots. You can format a disk by using the **format** command.

To format a blank disk in drive B, you would use the following command:

```
format b: /v
|         |         |
|         |         |...and ask for a label.
|         |         |...on drive B,...
|         |         |Format a disk...
```

Note If you have only one disk drive, MS-DOS prompts you to insert the disk that you want to format. Refer to the section "If You Have Only One Floppy Disk Drive" in Chapter 3.

You can also format a blank disk so that some special MS-DOS files are copied onto it during formatting. These files are necessary if you want to use the disk to start MS-DOS. So to format a blank disk in drive B and include the special MS-DOS files, you would use the following command:

```
format b: /v /s
Format a      ...and copy the special MS-DOS files.
disk...      ...ask for a label,...
              ...on drive B,...
```

It is important to copy these files with the **format** command when you make a copy of your MS-DOS master disk.

If you don't want to use the disk to start MS-DOS you don't need to specify the /s option. But if you have a disk and don't know whether you can use it to start MS-DOS, just put the disk into drive A and press CONTROL-ALT-DEL. If the disk doesn't contain the system files, MS-DOS displays an error message.

Example

Formatting a disk

Suppose you need to create a new data disk to hold some tax records, but you don't want to copy the special MS-DOS files when formatting the disk. So to format and label a blank disk (in drive B) *without including the special MS-DOS files*, you simply follow these steps:

- 1 Make sure that the MS-DOS disk is in drive A.
- 2 At the MS-DOS prompt, type the following command:

```
format b: /v
```

- 3 Press the RETURN key.
Your screen should look like this:

```
A>format b: /V
Insert new diskette for drive B:
and strike ENTER when ready_
```

- 4 Insert a blank disk in drive B.

- 5 Press the RETURN key to start the format process.

When formatting is complete, MS-DOS displays the following prompt:

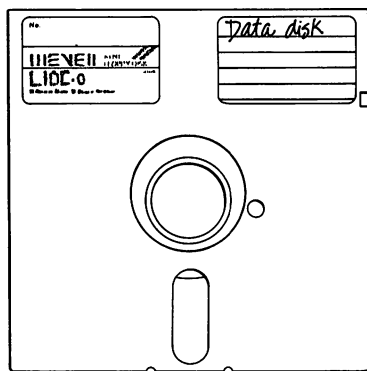
Volume label (11 characters, ENTER for none)?

- 6 Type a label that identifies the contents of this disk (for example, *DATA DISK*), and press RETURN. MS-DOS then asks:

Format another? (Y/N)

- 7 Press N (for No) to exit the **format** program.

Now your disk is formatted and ready to use. Be sure to label it on the outside cover, and remember to include the volume label that you used in step 6. The label will remind you that you have formatted the disk, and will help you identify its contents.



Warning The **format** program destroys any information that is on a disk. It's a good idea to list the directory of a disk that contains files *before* you format it; then you will be sure that you aren't destroying important files.

The Diskcopy Command

You may often need to make copies of entire disks instead of individual files. You can do this easily with the MS-DOS **diskcopy** command. To use the **diskcopy** command, you must have:

- an MS-DOS disk
- a disk you want to copy
- a blank disk to put the copy on

To copy the contents of a disk in drive A to a disk in drive B, you would use the following command:

```
diskcopy a: b:
Copy a disk...    ...to drive B.
                  ...from drive A...
```

Example

Copying a disk

Suppose you want to bring a data disk with you on a business trip, but you don't want to take your original disk because it might get damaged. All you have to do is use the **diskcopy** command to make a copy of the disk. For example, to copy the contents of a disk in drive A to a disk in drive B, you simply follow these steps:

- 1 Put the disk you want to copy in drive A.
- 2 Put a blank disk in drive B.
- 3 At the MS-DOS prompt, type the following command:

```
diskcopy a: b:
```

- 4 Press the RETURN key.
Your screen should look like this:

```
A>diskcopy a: b:
```

```
Insert SOURCE diskette in drive A:
```

```
Insert TARGET diskette in drive B:
```

```
Press any key when ready . . .
```

- 5 Press the SPACEBAR to start the **diskcopy** process.
When the disk has been copied, MS-DOS asks:

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

- 6 Press N (for No) to exit the **diskcopy** program.

What's next?

If you'd like more detailed information about the commands in this chapter, see your *MS-DOS User's Reference*. But if you are satisfied that you know enough about these commands, you can go on to the next chapter, where you'll learn how to run some applications with MS-DOS.

5 Using Applications with MS-DOS

In this chapter you will learn some common uses of MS-DOS, such as:

- Running application programs
- Using **edlin**

How to Run Application Programs

MS-DOS lets you run many different application programs, including spreadsheets, word processing programs, and graphics packages. These application programs can help you in a number of ways. For instance, they can help you balance a budget, figure income taxes, or manage information, such as inventories, stocks, and address lists.

Starting an application

Once you have started MS-DOS you can run an application program, as follows:

- ❶ Place the application disk in an empty disk drive (usually drive B if you have two floppy disk drives or a hard disk).
- ❷ Change the default drive to the one that contains the application disk.
- ❸ Type the name of the application program you want to run.

For example, if you had a word processor called Phrase, you might want to use it to write a monthly status report. Suppose also that to start Phrase you must type the word "phrase."

So, to create this report, you would follow these steps:

- ❶ Put your Phrase program disk into drive B.
- ❷ Change your default disk drive to B by typing the following:

b :
- ❸ Start Phrase by typing "phrase."

Then you could use Phrase to create, edit, format, or print your status report.

How to Create a File with Edlin

MS-DOS includes a program called *edlin.exe*, a *line editor* that lets you create and edit files. **Edlin** is called a line editor because it lets you edit files line by line.

To help you learn how to use **edlin**, the following section takes you through a sample editing session in which you'll use **edlin** to create a small file.

Suppose a client asks you to write a catchy advertisement for an electric pencil sharpener, so you decide to create a file named *pencil.ad* on the disk in the default drive. You want the file to contain the following lines:

Introducing...

The X-1000 Automatic Pencil Sharpener

From Sharpe Office Supplies

The World Leader in Office Sharpeware

Creating a file with edlin

The following example shows you how to start **edlin**, create the *pencil.ad* file, and exit **edlin**. All you have to do is follow these steps:

- 1 Make sure the MS-DOS disk is in drive A.
- 2 At the MS-DOS prompt, type the following command, then press the RETURN key:

```
edlin pencil.ad
```

Since you are just creating the file, **edlin** responds with the following message:

```
New file
*_
```

- 3 When you see the asterisk (*), type the letter I (for Insert) and press the RETURN key. You will see line number 1. **Edlin** uses line numbers to help you edit, but they are not part of your file.
- 4 Type the following lines. Remember to press the RETURN key after *each* line, including the last line.

```
Introducing...
The X-1000 Automatic Pencil Sharpener
From Sharpe Office Supplies
The World Leader in Office Sharpeware
```

Note Use the BACKSPACE key to erase a mistake on a line before you press the RETURN key. If you do press the RETURN key before correcting the mistake, don't worry about it, this is just a practice session.

Your screen should look like this:

```
A>edlin pencil.ad
New file
*i
1:*Introducing...
2:*The X-1000 Automatic Pencil Sharpener
3:*From Sharpe Office Supplies
4:*The World Leader in Office Sharpeware
5:*_
```

- 5 When you see 5*, press CONTROL-C to return to the **edlin** prompt (the asterisk).

- 6 Then, at the asterisk (*), type the letter E (for End). You will then be returned to the MS-DOS prompt.

You now have a file named *pencil.ad* on the disk in your default drive. If you type the MS-DOS **dir** command, you should see an entry for *pencil.ad*. You can also view this file by using the **type** command as follows:

```
type pencil.ad
```

To learn more about how to use **edlin**, refer to the *MS-DOS User's Reference*.

Summary

So far you have learned the basics of MS-DOS. These include:

- MS-DOS terms
- The MS-DOS keyboard
- Using disks, files, and directories
- Starting and quitting MS-DOS
- Using file and disk commands
- Printing files
- Running applications
- Starting **edlin**

What's next?



If you don't remember everything about these topics, just refer to the appropriate sections to refresh your memory. It's also a good idea to look ahead at the "Terms" section. There you can browse through the definitions and familiarize yourself with MS-DOS terminology. Then once you're confident in your abilities, you can proceed to the more advanced *MS-DOS User's Reference*.

In the *User's Reference* you'll read and learn about multilevel directories, batch files, more about **edlin**, and much more. Also, if you need a more detailed explanation or example of how to use a command, you can refer to the commands section of the *MS-DOS User's Reference*. So if this *User's Guide* hasn't answered all your questions, you know where to look for help.

Terms

. This abbreviation means *all files in the directory*. The command "copy a: *.* b:" means *copy all files from the disk in drive A to the disk in drive B*.

Application software Another name for software, programs, or application programs. Software is written in a computer language and consists of a series of instructions that tell the computer to perform tasks.

Backup disk A copy of any disk you make with the **diskcopy**, **copy**, or **backup** command. (See the *MS-DOS User's Reference* for more information on the **backup** command.) You should always make a backup copy of the MS-DOS master disk before you begin using MS-DOS on a routine basis. Store the master disk in a safe place and use the copy for your work.



Byte A unit of information on a computer. You can use the **dir** command to see how many bytes, or characters, are in a file.

Character A letter, number, or symbol that you type at your keyboard or see on your screen.

Command A short program that tells MS-DOS how to do a specific task.

CONTROL key Used in combination with other keys to give MS-DOS special commands such as "stop the last command" and "stop the display from scrolling." Press the CONTROL key at the same time as you press another key.

CONTROL-C A control key sequence that stops a command while it is running. See also *CONTROL key*.

CONTROL-S A control key sequence that stops or restarts the scrolling of the screen display. See also *CONTROL key*.

Copy An MS-DOS command that copies one or more files on the same disk, or from one disk to another.

Cursor The lighted shape on the screen that shows where the next character you type will appear. The cursor is usually a blinking line or small box.

A

B

C

D

Default disk drive The drive where MS-DOS searches for any filenames that you may type. MS-DOS looks for files in the default drive unless you specify a different drive. The standard MS-DOS prompt contains the default drive letter. For example, if the prompt is A>, then "A" is the default drive.

Del An MS-DOS command that tells MS-DOS to delete one or more files. A synonym for **del** is **erase**.

Device errors Errors that MS-DOS displays while reading or writing to devices on your computer. A device can be, for example, a printer, a disk drive, or a monitor.

Dir An MS-DOS command that means "directory." When you type the **dir** command, MS-DOS displays the contents of the disk in the default drive. The command **dir b:** displays the contents of the disk in drive B.

Directory A table of contents for a disk. The directory contains the names of your files, the sizes of the files, and the dates they were created or last modified.

Disk See *Floppy disk; Hard disk*.

Disk drive A piece of hardware attached to your computer. A disk drive can be either a floppy or a hard disk drive. You insert floppy disks into floppy disk drives. Usually, hard disks are built into the computer.

Floppy disk drives are commonly referred to as the A drive and the B drive. Hard disks are usually the C drive. Your computer manual should tell you how your drives are labeled.

Disk operating system A group of programs that act as a translator between you and your computer. MS-DOS is a disk operating system.

Diskcopy An MS-DOS command that copies disks. **diskcopy** formats a disk before copying files onto it.

Drive name Consists of a drive letter and a colon. A drive name tells MS-DOS which drive to search for the file. For example, the command **type a:progress.rpt** contains a drive name (a:) that tells MS-DOS to look on the disk in drive A for the file called *progress.rpt*.

Editor A program that allows you to manipulate text and data on the computer. Editors allow you to move, add, and delete characters and lines, and to save files. The MS-DOS editor is called **edlin**.

Edlin A line-oriented editor that comes with MS-DOS. See also *Editor*.

E

ENTER key See *RETURN key*.

Erase A synonym for the MS-DOS **del** command. See also *Del*.

Error messages Messages that appear on the screen after MS-DOS detects a problem while trying to process a command or program. See the *MS-DOS User's Reference* for the appropriate response to each error message.



File A collection of related information. A file on a disk can be compared to a file folder in a desk drawer. For example, a file folder named *friends* might contain the names and addresses of your friends. A file on a disk could contain the same information, and could also be named *friends*. Programs are also stored in files.

Filename A filename can be from one to eight characters in length and can have an extension of up to three characters separated from the filename by a period (.). An example of a complete filename is *progress.rpt*. Certain filenames are reserved by MS-DOS and should not be used when naming your files. These filenames are: *aux*, *clock\$*, *com*, *con*, *kybd\$*, *lpt*, *lst*, *nul*, *prn*, and *scrn\$*.

Filename extension A filename extension contains from one to three characters. Most application programs supply their own extensions for files they create. For example, all GW-BASIC files use a filename extension of *.bas*. See also *Filename*.

Fixed disk See *Hard disk*.

Floppy disk Used for storing programs and files.

Format An MS-DOS command that structures blank disks so that MS-DOS can store data on them. You must format every blank disk before it can be used with MS-DOS. **Format** also checks the disk for defective spots.

GW-BASIC A general-purpose computer language. Often, BASIC (or GW-BASIC) is the first computer language that people learn.

Hard disk Sometimes called a fixed disk, one that is built into the computer. A hard disk can store much more information than a floppy disk, and the computer can retrieve information from it faster.

Memory Another name for *computer storage*.

Monitor The computer screen.

MS-DOS master disk MS-DOS is distributed on one or more floppy disks (called *master disks*) along with the user's manuals. Before you start using MS-DOS on a routine basis, you should always make a backup copy of the master disk or disks.

F

G

H

M

O

Operating system A series of programs that translate your commands to the computer, helping you perform such tasks as creating files, running programs, and printing documents. See also *Disk operating system*.

P

Print An MS-DOS command that prints files on your printer.

Printer A printing device that is attached to your computer. It lets you print files so that you have a paper copy or printout.

Program A set of instructions, written in computer language, that tells the computer how to perform some task.

Prompt A word or symbol that MS-DOS shows on the screen to tell you it is ready for you to type something. The standard MS-DOS prompt consists of the default drive letter (usually A, B, or C) and a greater-than sign. An example of the MS-DOS prompt is B>.

R

Rename An MS-DOS command that renames files. You can use the abbreviation **ren** in place of the full command name.

RETURN key The key you usually press after entering data or text, or after you type an MS-DOS command. On some computers the RETURN key is called the ENTER key.

S

Scrolling The movement of text on your screen as it rolls up and off the top of the screen.

Software The programs, routines, or instructions that allow the computer to perform tasks. Some examples of software include: operating systems, word processing programs, and spreadsheets.

T

Task Something your computer does when you give it a command. A program is an example of a task.

Type An MS-DOS command that displays files on the monitor.

V

Volume label An internal name on a disk. You should put a volume label on each of your disks to help you identify them.

W

Write-protect tab The small, removable tab that covers the write-protect notch on a disk. You can copy information onto the disk by removing the tab. See also *Write-protected disk*.

Write-protected disk A floppy disk that you can examine but not change. These disks are called *write-protected disks*. They usually have a small tab covering a notch on the right edge of the disk. If the disk does not have a write-protect notch, you cannot change the information on the disk.

Index

Abbreviation, wildcard 28, 39
Advanced topics, *MS-DOS User's Reference* 38
Application programs, running 35
Application software. *See* Software
Arrow keys. *See* Direction keys
Asterisk (*)
 Edlin prompt 37,
 wildcard character 28, 39

BACKSPACE key 6, 18, 37
Backup command 39
Backup copy 19
Backup disk, definition 39
.Bas extension 3, 41
Blank disk 33
Blank disk, formatting 31
Booting MS-DOS. *See* Starting MS-DOS
Byte, definition 39

Capital letters 3, 5
Capital letters, in filenames 13
Changing a filename 29
Characters, deleting 6
.Com extension 3
Command
 Backup 39
 Copy 22, 27, 39
 definition 5, 39
 Del 28, 40
 Dir 25, 40
 Diskcopy 19, 33, 39, 40
 Erase 41
 Format 13, 21, 31
 Print 30, 42
 Rename 29
 stopping 39
 Type 29, 42
Communications port 31
Computer language, GW-BASIC 41

Computer memory 41
Computer system 12
CONTROL key 6, 39
Control key sequence
 CONTROL-ALT-DEL 6, 18, 32
 CONTROL-C 6, 37, 39
 CONTROLS 6, 26, 30, 39
 use of 39
Copy command 22, 27, 39
Copying a disk 33
Copying files 22, 27
Correcting typing mistakes 18, 37
Creating a file, Edlin 36
Cursor
 moving 6
 MS-DOS 4

Date prompt 17
Default disk drive, definition 4
Default drive 23, 27, 30, 40
Del command 28, 40
Deleting
 characters 6
 files 28
Device error 40
Device names 14
Dir command 25, 40
Direction keys 6
Directory
 contents 14
 definition 3, 40
 displaying 14
 listing 25
 MS-DOS 14
 variations 15
Disk
 backup 39
 blank 33
 blank, formatting 31
 copying 33

Disk (*continued*)

- floppy 10
- formatting 13, 31
- hard 12
- label 33
- safe storage of 19
- volume label 3
- write-protected 11, 42

Disk commands 31

Disk drive

- definition 3, 40
- floppy 3
- hard 3

Disk operating system, MS-DOS 40, 42

Disk protection 42

Diskcopy command 19, 33, 39, 40

Diskcopy screen 19, 34

Diskcopy.exe program 19

Displaying a file 29

.Doc extension 3

DOWN arrow key 6

Drive letter 4

Drive name

- definition 4
- explanation 40
- floppy disk 40
- hard disk 40

Editing a file, Edlin 36

Editor, definition 40

Edlin

- creating a file 37
- definition 40
- editing a file 37
- exiting 37
- exiting a file 37
- line editor 40
- line numbers 37
- prompt, asterisk (*) 37
- screen 37

ENTER key. *See* RETURN key

Erase command. *See* Del command

Erasing a file 28

Error message 41

Error, device 40

.Exe extension 3

Executable file, .exe 3

Extension

- .bas 3, 41

Extension (*continued*)

- .com 3
- .doc 3
- .exe 3
- .rpt 3
- .txt 3
- command 3
- executable 3
- filename 3, 41

File

- copying 27
- definition 2, 41
- deleting 28
- displaying 29
- hidden 15
- naming 13
- printing 30
- renaming 27, 29

File commands 25

Filename

- allowed length 41
- definition 3, 13, 41
- extension 3, 13, 41
- illegal 14
- invalid characters in 13
- length 3
- parts 13
- reserved 41

Fixed disk. *See* Hard disk

Floppy disk

- copying 19
- description 10
- do's and don'ts 11
- master 17
- storage capacity 10
- storing 10, 19
- use of 41
- write-protected 11, 42

Floppy disk drive, drive name 40

Floppy disk drive, if you have only one 23

Format command 13

- adding a label 32
- copying a special files 32
- description 13
- formatting a disk 32
- formatting a hard disk 21
- options 32
- purpose 31, 41

Format command (*continued*)
 warning 21, 33
 Format screen 32
 Formatting a disk 13, 31
 Further reference
MS-DOS User's Reference 38
 Terms 38

Glossary 38
 GW-BASIC, definition 41

Hard disk
 backup copy of MS-DOS 21
 configuring 21
 copying master disk onto 22
 description 12, 41
 formatting 21
 storage capacity 41
 Hard disk drive, drive name 40
 Hidden files 15

Internal name. *See* Volume label
 Invalid filenames 14

Key
 BACKSPACE 6, 18, 37
 CONTROL 6, 39
 direction
 DOWN arrow 6
 LEFT arrow 6
 RIGHT arrow 6
 UP arrow 6
 RETURN 7, 41, 42
 SHIFT 6
 SPACEBAR 6
 Keyboard
 differences between keys 5
 keys 5
 special keys 6

Label 10
 Labeling a disk 33
 LEFT arrow key 6
 Letters, lowercase 3, 5, 13
 Letters, uppercase 3, 5, 13

Line editor, Edlin 36, 40
 Line numbers, Edlin 37
 Listing a directory 25
 Lowercase letters 3, 5, 13

Master disk, MS-DOS 17, 19, 41
 Memory, definition 41
 Message, error 41
 Mistakes in typing, correcting 37
 Monitor 41
 Moving the cursor 6
 MS-DOS
 a quick guide 38
 advanced topics 38
 cursor 4
 default drive letter 40
 directory 14
 glossary 39
 line editor, Edlin 36
 operating system, definition 40, 42
 prompt 4
 quitting 19
 starting and restarting 6, 17, 18
 terms 2, 39

MS-DOS command
 Backup 39
 Copy 22, 27, 39
 definition 5, 39
 Del 28, 40
 Dir 25, 40
 Diskcopy 19, 39, 40
 Erase 41
 Format 13, 21, 31
 Print 30, 42
 Rename 29
 Type 29, 42

Notch, write-protect 42

Operating system, MS-DOS 40, 42

Port, communications 31
 Print command 30, 42
 Printer, definition 42
 Printing a file 30
 Program, definition 2, 42

Programming language, GW-BASIC 41

Prompt

A> 18

date 17

default drive letter 42

definition 42

Edlin, asterisk (*) 37

MS-DOS 4, 42

time 17

Protecting your disks 11, 14

Quitting

Edlin 37

MS-DOS 19

Re-booting MS-DOS. *See* Restarting MS-DOS

References

MS-DOS User's Reference 38

Terms 38

Removing a file 28

Rename command 29, 42

Renaming a file 27, 29

Reserved filenames 41

RESET key 7

Restarting MS-DOS 6, 18

RETURN key 7, 41, 42

RIGHT arrow key 6

.Rpt extension 3

Running application programs 35

Screen

Diskcopy 19, 34

Edlin 37

Format 32

startup, MS-DOS 17

Scrolling, starting and stopping 26, 30, 39

Setting the date and time 18

SHIFT key 6

Shortcuts

Rename command 29

wildcard abbreviation 28, 39

Single drive computers 23

Software

application 42

operating system 42

program 42

spreadsheet 42

Software (*continued*)

word processing 42

Software, definition 39

See also Application programs

SPACEBAR key 6

Special characters 13

Special files, MS-DOS 32

Special keys 6

Starting

application programs 35

MS-DOS 6, 17, 18

Startup screen 17

Stopping

a command 6

scrolling 6

Storage, computer. *See* Memory

Synonym

Del 28

Erase 28

System prompt, MS-DOS 4

Tab, write-protect 42

Task 5, 42

Terms, MS-DOS 2, 39

Text editor, Edlin 36

Text file 29

Time prompt 17

.Txt extension 3

Type command 29, 42

Typing mistakes, correcting 6, 18, 37

UP key 6

Uppercase letters 3, 5, 13

Valid filename characters 13

Volume label 3, 21, 33, 42

Wildcard characters, asterisk (*) 28, 39

Write-protect notch 12, 42

Write-protect tab 11, 42

Write-protected disk 42

CAUTION

During shipment, the hard disk drive of your PC Commodore must be in a parking position. *On your hard disk* there is an additional SHIPDISK command to set your drive into parking position. Before the system is powered down for shipping or transportation, this command should be issued as the last DOS command. SHIPDISK is not described in the DOS manual. Since this command is not on the standard system diskette, you *should copy it there*.



User's Reference

Information in this document is subject to change without notice and does not represent a commitment on the part of Microsoft Corporation. The software described in this document is furnished under a license agreement or nondisclosure agreement. The software may be used or copied only in accordance with the terms of the agreement. It is against the law to copy this software on magnetic tape, disk, or any other medium for any purpose other than the purchaser's personal use.

© Copyright Microsoft Corporation, 1986

Microsoft®, MS-DOS®, and XENIX® are registered trademarks and The High Performance Software™ is a trademark of Microsoft Corporation.

IBM® is a registered trademark of International Business Machines Corporation.

Intel® is a registered trademark of the Intel Corporation.

Lotus® is a registered trademark of the Lotus Corporation.

Contents

Introduction ix

About This Manual ix

How to Use This Manual x

1 More About Files and Directories 1

Before You Start 1

File Protection 2

How MS-DOS Keeps Track of Your Files 2

Multilevel Directories 3

Paths 6

Pathnames 6

Wildcards 8

The ? Wildcard 8

The * Wildcard 8

Using Directories 9

How to Create a Directory 10

How to Change Your Working Directory 10

How to Display Your Working Directory 10

How to Delete a Directory 11

How to Rename a Directory 12

2 About Commands 13

Types of MS-DOS Commands 13

Redirecting Command Input and Output 16

How to Redirect Your Output 16

Filters and Pipes 17

Command Pipes 17

3 MS-DOS Commands 19

About This Chapter 19

Command Options 20

More About Options 21

MS-DOS Commands 22

4 Batch Processing 101

Why Use Batch Files? 101

How to Create Batch Files 101

About Batch Processing 102

What is an Autoexec.bat File? 103

How to Create an Autoexec.bat File 105

How to Create a Batch File with Replaceable Parameters	106
How to Run a Batch File	107
How to Use Temporary Files	108
Batch Processing Commands	108
Echo	109
For	110
Goto	112
If	113
Pause	114
Rem	115
Shift	116
5 MS-DOS Editing and Function Keys	117
Special MS-DOS Editing Keys	117
How MS-DOS Uses the Template	117
Special Editing Functions	119
Control Characters	122
6 The Line Editor (Edlin)	125
About Edlin	125
How Edlin Works	125
How to Start Edlin	126
How to Quit Edlin	127
Special Editing Keys	128
F1	129
F2	130
F3	131
DEL	132
F4	133
ESC	134
INS	135
F5	137
7 Edlin Commands	139
Some Tips for Using Edlin Commands	140
Edlin Command Options	142
The Line Option	142
The Question Mark Option	142
The Text Option	143
Append	144
Copy	145
Delete	147
Edit	150
End	152
Insert	153
List	156
Move	159
Page	161

- Quit 162
- Replace 163
- Search 166
- Transfer 169
- Write 170

8 File Comparison Utility (FC) 171

- Introduction 171
- Limitations on Comparisons 172
- How to Use FC 172
- FC Switches 173
- How FC Reports Differences 175
- Redirecting FC Output to a File 175
- Examples 176

9 Link: A Linker 181

- Introduction 181
- Starting and Using Link 182
 - Using Prompts to Specify Link Files 182
 - Using a Command Line to Specify Link Files 184
 - Using a Response File to Specify Link Files 187
 - Using Search Paths with Libraries 189
 - The Map File 190
 - The Temporary Disk File — Vm.tmp 191
- The Link Options 192
 - Viewing the Options List 193
 - Pausing to Change Disks 193
 - Packing Executable Files 194
 - Producing a Public-Symbol Map 194
 - Copying Line Numbers to the Map File 195
 - Preserving Lowercase 196
 - Ignoring Default Libraries 197
 - Setting the Stack Size 197
 - Setting the Maximum Allocation Space 198
 - Setting a High Start Address 199
 - Allocating a Data Group 199
 - Removing Groups from a Program 200
 - Setting the Overlay Interrupt 200
 - Setting the Maximum Number of Segments 201
 - Using DOS Segment Order 202
- How Link Works 203
 - Alignment of Segments 203
 - Frame Number 203
 - Order of Segments 204
 - Combined Segments 204
 - Groups 205
 - Fixups 205

10 Debug 207

Introduction	207
How to Start Debug	208
Method 1: Debug	208
Method 2: Command Line	209
Debug Command Information	209
Debug Command Parameters	210
Assemble	213
Compare	215
Dump	216
Enter	218
Fill	220
Go	221
Hex	223
Input	224
Load	225
Move	227
Name	228
Output	230
Quit	231
Register	232
Search	235
Trace	236
Unassemble	237
Write	239
Debug Error Messages	241

Appendix A Instructions for Users with Single Floppy Disk Drive Systems 243

Appendix B How to Configure Your System 245

What is a Configuration File?	245
Config.sys Commands	246
Break	247
Buffers	248
Country	249
Device	250
Drivparm	251
FCBS	253
Files	254
Shell	255
Sample Config.sys File	256

Appendix C Installable Device Drivers 257

Introduction 257

Ansi.sys 257

Notes: 257

Cursor Functions 258

Erase Functions 259

Modes of Operation 260

Driver.sys 263

Ramdrive.sys 264

Appendix D Disk and Device Errors 267

Type Messages 267

Action Messages 269

Device Messages 269

Appendix E MS-DOS Message Directory 271

MS-DOS Messages 271

Appendix F Configuring Your Hard Disk (Fdisk) 313

Introduction 313

How to Start Fdisk 314

Choice 1 — Creating a DOS Partition 315

Choice 2 — Changing the Active Partition 316

Choice 3 — Deleting a DOS Partition 317

Choice 4 — Displaying Partition Data 318

Choice 5 — Next Fixed Disk 319

Index 321

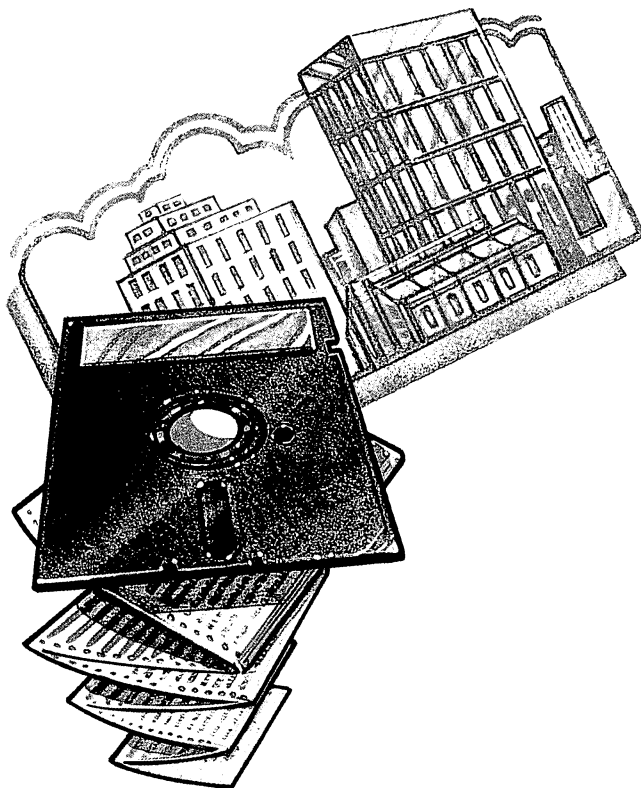


Introduction

About This Manual

This is a reference manual for the MS-DOS® operating system. By now you are probably familiar with the introductory information in the *MS-DOS User's Guide* and know how to start MS-DOS, how to copy, delete, and rename files, how to make copies of disks, and how to run applications.

Also, you should have already made a backup copy of your MS-DOS master disk and stored the original in a safe place. If you haven't done so, refer to the *MS-DOS User's Guide* to learn how to make a backup copy of your disk.



How to Use This Manual

The following table is a quick overview of the topics covered in this manual. For more specific topics, you may have to check the table of contents or the index.

Turn to	If you need to know
Chapter 1	About multilevel directories About paths
Chapter 2	About MS-DOS commands
Chapter 3	How to use MS-DOS commands
Chapter 4	How to make a batch file How to use commands
Chapter 5	About MS-DOS editing keys
Chapter 6	How to use edlin , the line editor
Chapter 7	How to use edlin commands
Chapter 8	How to use FC, the file comparison utility
Chapter 9	How to create executable files with link
Chapter 10	How to debug files with debug
Appendix A	What to do if you have only one disk drive
Appendix B	How to configure your system
Appendix C	About ANSI escape sequences About installing device drivers
Appendix D	What a disk error means
Appendix E	What an MS-DOS error message means
Appendix F	How to configure your hard disk

What's next

Now that you've seen a brief summary of what topics this manual covers, you're ready to start with Chapter 1, "More About Files and Directories." There you'll learn about some of the more advanced features of MS-DOS.

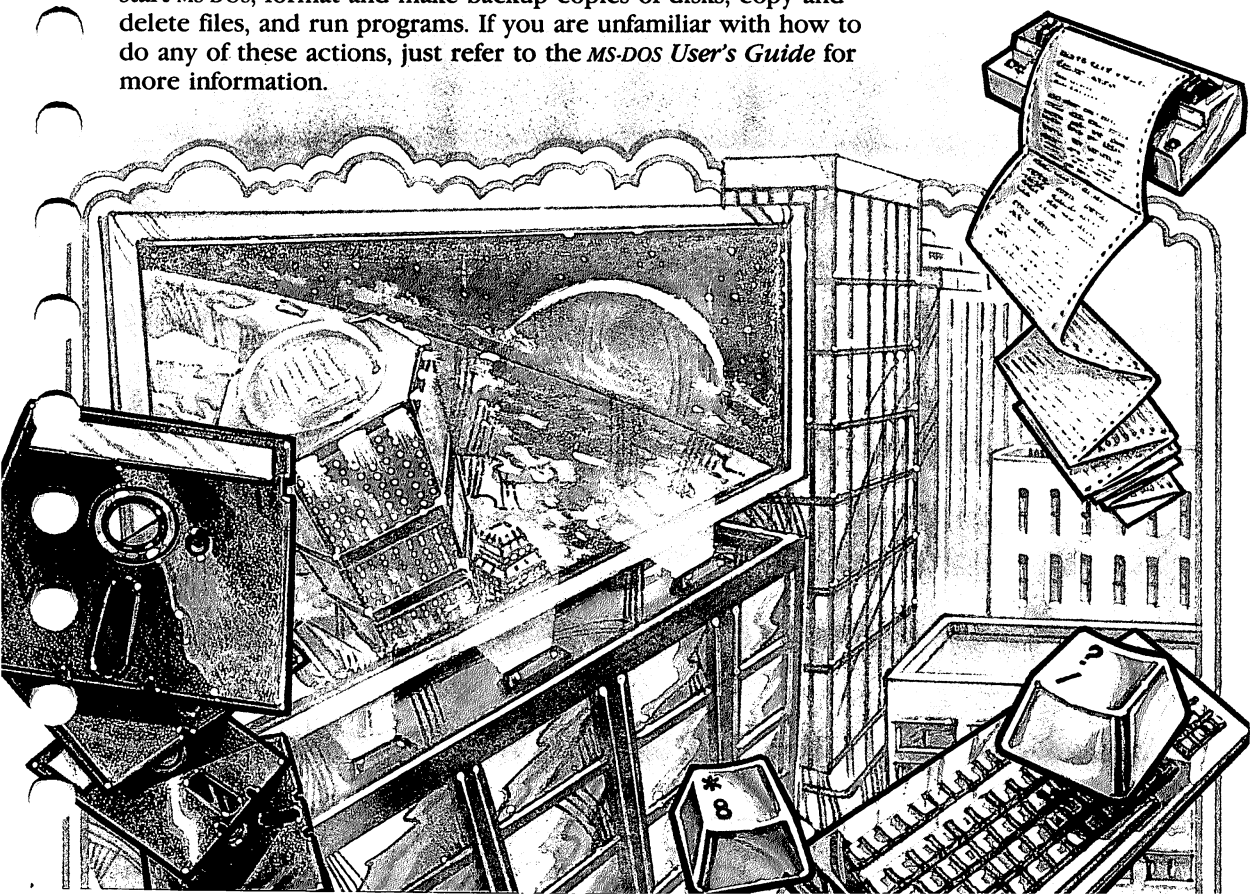
1 More About Files and Directories

In this chapter you will learn about:

- Protecting and keeping track of your files
- Multilevel directories
- Using wildcards

Before You Start

Before you read this chapter, you should already know how to start MS-DOS, format and make backup copies of disks, copy and delete files, and run programs. If you are unfamiliar with how to do any of these actions, just refer to the *MS-DOS User's Guide* for more information.



Protecting your files

File Protection

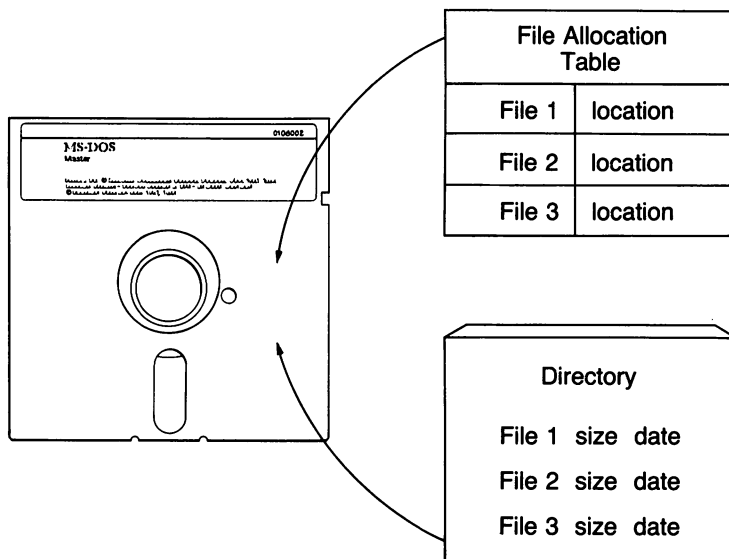
The MS-DOS operating system is a powerful and useful tool for processing personal and business information. As with any computer, though, errors may occur and information may be misused. So, if you are doing work that cannot be replaced or that requires a lot of security, you should protect your programs.

You can take simple but effective measures like putting your disks away when you're not using them, or covering the write-protect notch on your program disks. Also, if your disks contain valuable information, you should make backup copies of them on a regular basis. Another way to protect your programs is by installing your equipment in a secure office or work area.

File Allocation Table

How MS-DOS Keeps Track of Your Files

As you learned in the *MS-DOS User's Guide*, MS-DOS stores files in directories. In addition to directories, it uses an area on a disk called the File Allocation Table. When you format a disk with the **format** command, MS-DOS copies this table onto the disk and creates an empty directory, called the **root** directory. So, on each of your disks, the directories store the files, and the File Allocation Table keeps track of their locations. The table also allocates the free space on your disks so that you have enough room to create new files.



These two system areas, the directories and the File Allocation Table, enable MS-DOS to recognize and organize the files on your disks. To check these areas on a disk for consistency and errors you should use the MS-DOS **chkdsk** command.

For example, to check the disk in drive A, type the **chkdsk** command followed by *a:*

In response, MS-DOS displays a status report and any errors it has found, such as files that show a nonzero size in the directory but that really have no data in them.

For an example of such a display and for more information on **chkdsk**, see the description of the **chkdsk** command in Chapter 3, "MS-DOS Commands."

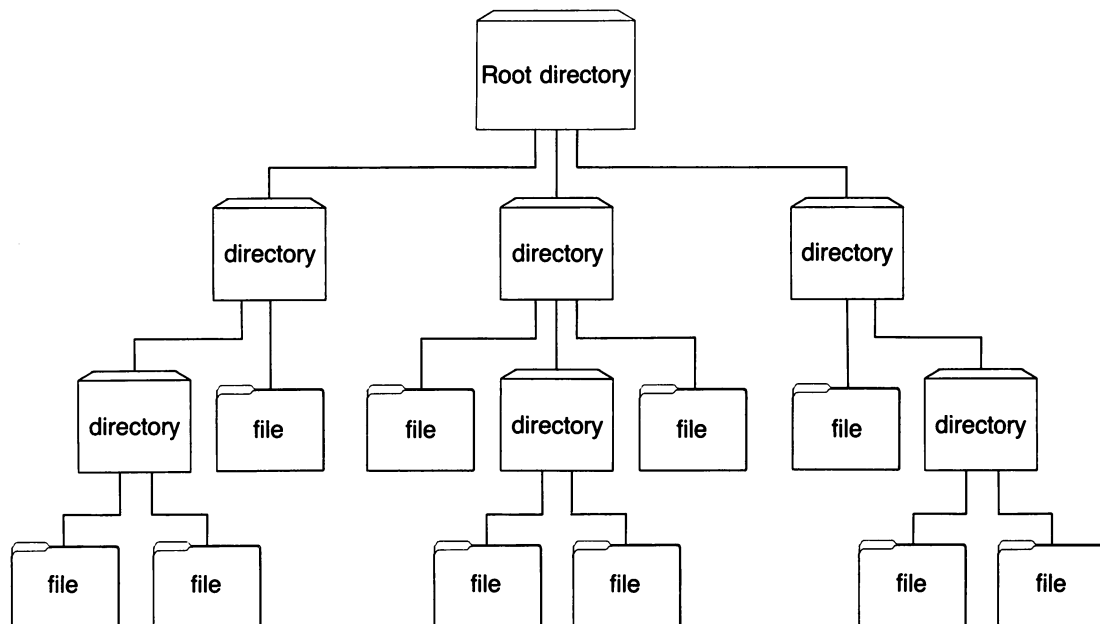
Checking a disk

Multilevel Directories

When there is more than one user on your computer, or when you are working on several different projects, the number of files in the directory can become large and unwieldy. You may want to keep your files separate from a coworker's, or you may want to organize your programs into convenient categories.

In an office, you can separate and organize files that belong to different people or that relate to specific projects by putting them in different file cabinets. For example, you might put your accounting programs in one file cabinet and your letters in another. You can do the same thing with MS-DOS by putting your files into different directories.

Using directories is one way that you can divide your files into convenient groups. Any one directory can contain a maximum of 255 files and directories. These directories may also contain other directories (referred to as *subdirectories*). This organized file structure is called a *multilevel* or *hierarchical directory* system.



The root directory

The first level in a multilevel directory is the root directory, which is created automatically when you format a disk and start putting files on it. You can create more directories and subdirectories within the root directory.

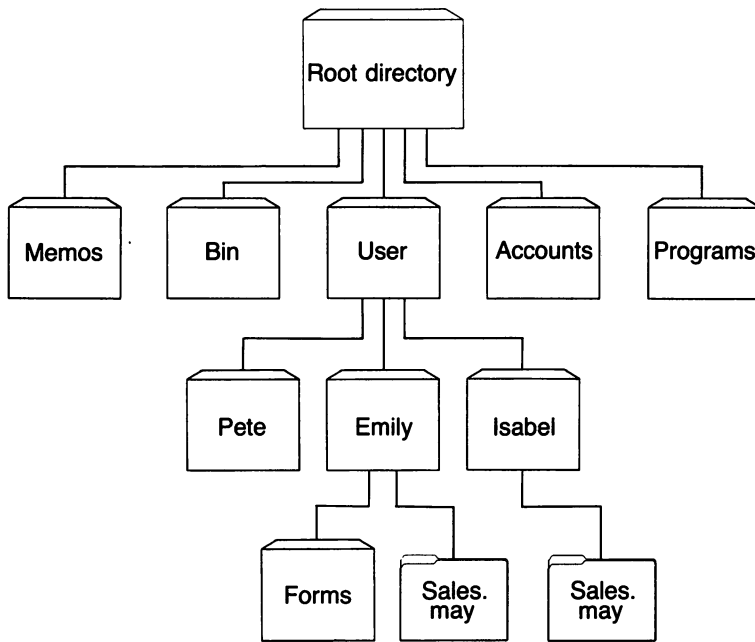
As you create new directories for groups of files, or for other people using the computer, the directory system grows. And within each new directory you can add new files or create new subdirectories.

You can move around in the multilevel system by starting at the root and traveling through intermediate subdirectories to find a specific file. Conversely, you can start anywhere within the file system and travel toward the root. Or you can go directly to any directory without traveling through intermediate levels.

Your working directory

The directory that you are in is called the *working directory*. The filenames and commands discussed in this chapter relate to your working directory and do not apply to any other directories in the structure. When you start your computer, you start out in the working directory. Similarly, when you create a file, you create it in the working directory.

Because you can put files in different directories, you and your coworkers can have files with the same names, but with unrelated content. The following figure illustrates a typical multilevel directory structure:



In this example, five subdirectories of the root directory have been created. These subdirectories are:

- A directory of external commands, named *bin*.
- A *user* directory containing separate subdirectories for all users of the system.
- A directory containing accounting information, named *accounts*.
- A directory of programs, named *programs*.
- A directory of text files, named *memos*.

As you can see, Pete, Emily, and Isabel each have their own directories, which are subdirectories of the *user* directory. Emily has a subdirectory named *forms*, and both Emily and Isabel have *sales.may* files in their directories, even though Isabel's *sales.may* file is unrelated to Emily's.

This organization of files and directories is not important if you work only with files in your own directory, but if you work with someone else, or on several projects at once, the multilevel directory system becomes handy. For example, you could get a list of the files in Emily's *forms* directory by typing the following command:

```
dir \user\emily\forms
```

Note that a backslash (\) separates directories from other directories and files. In the previous example the first backslash includes the root directory. The use of the backslash alone indicates the root directory. For example, the following command displays a list of the files in the root directory:

```
dir \
```

To find out what files Isabel has in her directory, you would type the following command:

```
dir \user\isabel
```

This command tells MS-DOS to travel from the root directory to the *user* directory to the *isabel* directory, and to then display all filenames in the *isabel* directory.

Paths

When you use multilevel directories, you must tell MS-DOS where the files are located in the directory system. Both Isabel and Emily, for example, have files named *sales.may*, so each would have to tell MS-DOS in which directory her file resides when she wants to use it. This is done by giving MS-DOS a *pathname* to the file.

Pathnames

A pathname is a sequence of directory names followed by a filename. Each directory name is separated from the previous one by a backslash (\).

Using pathnames

The general format of a pathname is as follows:

```
[ \directoryname ] [ \directoryname... ] \filename
```

A pathname may contain any number of directory names up to a total length of 63 characters. If a pathname begins with a backslash, MS-DOS searches for the file beginning at the root of the directory system. Otherwise, it begins at the working directory and searches along the path from there. Here are two examples:

The pathname of Emily's *sales.may* file is:

```
\user\emily\sales.may
```

The pathname of Isabel's *sales.may* file is:

```
\user\isabel\sales.may
```

When you are in your working directory, you may use a filename and its corresponding pathname interchangeably. Some sample names are:

<code>\</code>	The root directory.
<code>\programs</code>	A directory under the root directory that contains program files.
<code>\user\isabel\forms\1040</code>	A typical full pathname. This one is for a file named <i>1040</i> in the <i>forms</i> directory, which belongs to Isabel.
<code>sales.may</code>	A file in the working directory.

A *parent directory* is any directory that contains subdirectories. MS-DOS provides special shorthand notations for the working directory and the parent of the working directory, and automatically creates these two entries whenever you create a directory:

- MS-DOS uses the shorthand name “.” to indicate the name of the working directory in all multi-level directory listings.
 - .. These two dots are the shorthand name “..” for the working directory's parent directory (one level up). If you type the **dir** command followed by two dots, MS-DOS lists the files in the parent directory of your working directory.
- If you type the following command MS-DOS lists the files in the parent's *parent* directory:

```
dir ..\..
```

Parent directories

Wildcards

If you are using multilevel directories, you will find it easier to search for files on your disks if you use two special characters, called *wildcards*. The wildcard characters are the asterisk (*) and the question mark (?). They are useful in MS-DOS command lines because they give you flexibility when you are specifying paths and files.

Using the ? wildcard

The ? Wildcard

A question mark (?) in a filename or filename extension means that any character can occupy that position. The following command, for example, lists all filenames on the default drive that begin with the characters *memo*, that have any character in the next position, that end with the characters *aug*, and that have an extension of *.txt*:

```
dir memo?aug.txt
```

Here are some examples of files that might be listed by the above command:

```
MEMO2AUG.TXT  
MEMO9AUG.TXT  
MEMOBAUG.TXT
```

The * Wildcard

An asterisk (*) used in a filename or filename extension means that any character can occupy that position *or any of the remaining positions in the filename or extension*. For example, the following command lists all the directory entries on the default drive with filenames that begin with the characters *memo* and that have an extension of *.txt*:

```
dir memo*.txt
```

Here are some examples of files that might be listed by this **dir** command:

```
MEMO2AUG.TXT  
MEMO9AUG.TXT  
MEMOBAUG.TXT  
MEMOJULY.TXT  
MEMOJUNE.TXT
```

Important The wildcard abbreviation *.* refers to all files in the directory. This feature can be both powerful and destructive when used with MS-DOS commands. For example, the **del** command followed by the wildcard abbreviation *.* deletes all files on the default drive, regardless of filename or extension.

In general, you should not use more than one asterisk wildcard in a command line. For example, if you type the following command, all the files in the directory will be listed, not just those that contain the number "1":

```
dir *1*
```

Examples:

Suppose you want to find a certain accounting file but can't remember its exact name. What you can do is list the directory entries for all files named *accounts* in the default directory of drive A (regardless of their filename extensions). To do this quickly, you could just type the following command:

```
dir a:accounts.*
```

Similarly, to list the directory entries for all files with *.txt* extensions or in a directory called *reports* (regardless of their filenames) on the disk in drive B, type the following command:

```
dir b:\reports\*.txt
```

This command is useful if your text files have *.txt* extensions. For example, by using the **dir** command with wildcard characters, you could get a listing of all your text files—even if you don't remember their filenames. For more information on the **dir** command, refer to Chapter 3, "MS-DOS Commands."

Using the * wildcard

Using Directories

The following sections describe how to display, change, and delete any directory. You will also learn how to create directories and subdirectories.

How to Create a Directory

Creating a new directory

To create a subdirectory in your working directory, use the **mkdir** (make directory) command. For example, to create a new directory named *user* under your working directory, simply type the following command:

```
mkdir user
```

After MS-DOS runs this command, a new directory will exist under your working directory. You can also make directories anywhere in the directory structure by specifying **mkdir** followed by a path. MS-DOS automatically creates the "." and ".." entries in the new directory.

To put files in the new directory, you can use the MS-DOS line editor, **edlin**. Chapter 6, "The Line Editor (Edlin)," describes how to use **edlin** to create and save files. You can also create and save files if you have a word processing program such as Microsoft® Word.

How to Change Your Working Directory

Changing directories

With MS-DOS it is easy to change from your working directory to a different directory: you simply type the **chdir** (change directory) command followed by a path. For example, if you type **chdir \user** and then press the RETURN key, MS-DOS changes the working directory to *\user*. You can also specify any path after the command so that you can "travel" around the directory structure. The following command, for example, puts you in the parent directory of your working directory:

```
chdir ..
```

How to Display Your Working Directory

Displaying your working directory

All commands are executed while you are in your working directory. You can find out the name of the directory you are in by typing the MS-DOS **chdir** command with no path. For example, if your working directory is *\user\pete*, when you type **chdir** and press the RETURN key, you would see the following:

```
A:\USER\PETE
```

This is your *working drive*, A, plus the working directory, *\user\pete*.

Shortcut You can also type the letters **cd** for the **chdir** command to save time. For example, the following commands are the same:

```
cd \user\pete
chdir \user\pete
```

If you want to see the contents of the `\user\pete` directory, you can use the MS-DOS **dir** command. The subdirectory might look like this:

```
Volume in drive A has no ID
Directory of A:\USER\PETE
```

```
.                <Dir>          08-09-86      10:09a
..              <Dir>          08-09-86      10:09a
TEXT            <Dir>          08-09-86      10:09a
FILE1          TXT      5243    08-04-86      9:30a
    4 File(s)    836320 bytes free
```

Note that MS-DOS lists both files and directories in this output. As you can see from the display, Pete has a subdirectory named *text*; the “.” refers to the working directory `\user\pete`; the “..” is short for the parent directory `\user`, and *file1.txt* is a file in the `\user\pete` directory. All these directories and files are on the disk in drive A.

Note Because files and directories are listed together, you cannot give a subdirectory the same name as a file in that directory. For instance, if you already have a path `\user\pete`, where *pete* is a subdirectory, you cannot create a file named *pete* in the `\user` directory.

How to Delete a Directory

If you create a directory and decide later that you don't want it any more, you can delete it with the MS-DOS **rmdir** (remove directory) command.

The **rmdir** command lets you delete any directory by specifying its path, but the directory must be *empty* except for the “.” and “..” entries. This prevents you from accidentally deleting files and directories.

Removing a directory

To remove all the files in a directory (except for the "." and ".." entries), type **del** followed by the path of the directory. For example, to delete all files in the `\user\emily` directory, type the following command:

```
del \user\emily
```

MS-DOS prompts you with the following message:

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

If you really want to delete all the files in the directory, type *Y* (for Yes). If not, type *N* (for No) to stop the command.

Now you can use the **rmdir** command to delete the `\user\emily` directory by typing the following command:

```
rmdir \user\emily
```

Shortcut To save time you can also use the letters **rd** for the **rmdir** command.

How to Rename a Directory

Renaming a directory

There is no command to rename a directory in MS-DOS. You can, however, rename a directory that has no subdirectories. Suppose, for example, you want to rename the `\user\pete` directory and call it `\user\emily`. To do this you would follow these steps (remember to press the RETURN key after each step):

- 1 To create the new directory, type:

```
mkdir \user\emily
```

- 2 Then to copy the files from the old directory to the new directory, type:

```
copy \user\pete\*.* \user\emily
```

- 3 Now to delete the contents of the old directory, type:

```
del \user\pete\*.*
```

(Type *Y* in response to the prompt "Are you sure?")

- 4 Finally, to remove the old directory, type:

```
rmdir \user\pete
```


2 About Commands

In this chapter you will learn about:

- Internal and external MS-DOS commands
- Redirecting input and output
- Command grouping symbols

Types of MS-DOS Commands

There are two types of MS-DOS commands:

Internal commands

External commands

Internal commands are the simplest, most commonly used commands, but you cannot see them when you list the directory on your MS-DOS disk because they are part of a file named *command.com*. When you type internal commands, MS-DOS performs them immediately. This is because they were loaded into your computer's memory when you started MS-DOS. Following is a list of the MS-DOS internal commands:

break	exit	ren
chdir	for	rmdir
cls	goto	set
copy	if	shift
ctty	mkdir	time
date	path	type
del	pause	ver
dir	prompt	verify
echo	rem	vol

What are internal commands?

Using pathnames with internal commands

Some internal commands can use paths and pathnames. Specifically, four commands—**copy**, **dir**, **del**, and **type**—have greater flexibility when you specify a pathname after the command.

The formats of these commands are as follows:

copy *pathname* *pathname*

If the second *pathname* is a directory (a *path*), MS-DOS copies all the files you specify in the first *pathname* into that directory, as in the following example:

```
copy \user\pete\*.* sales
```

del *pathname*

If the *pathname* is a directory (a *path*), all the files in that directory are deleted. If you try to delete a path, the prompt “Are you sure (Y/N)?” is displayed. Type *Y* (for Yes) to complete the command, or *N* (for No) to stop the command. Example:

```
del \user\pete
```

dir *pathname*

The following command displays the directory for a specific *pathname*:

```
dir \user\pete
```

type *pathname*

You must specify a *pathname* (or *filename*) for this command. MS-DOS then displays this file on your monitor in response to the **type** command. Example:

```
type \user\emily\report.nov
```

What are external commands?

Any filename with an extension of *.com*, *.exe*, or *.bat* is considered an *external* command. For example, files such as *format.exe* and *diskcopy.exe* are external commands. And, because these commands are also files, you can create new commands and add them to MS-DOS. Programs that you create with most languages (including assembly language) will be *.exe* (executable) files. Note, however, that when you use an external command, you do not need to type its filename extension.

Note If you have more than one external command with the same name, MS-DOS will run only one of them, according to the following order of precedence: *.com*, *.exe*, *.bat*.

Suppose, for example, that your disk includes the files *format.exe* and *format.bat*. If you were to type the external command **format**, MS-DOS would always run the program *format.exe* first. To run the batch file *format.bat*, you would have to place it in a separate directory and give a path along with the external command.

The following external commands are described in Chapter 3, "MS-DOS Commands":

append	find	recover
assign	format	replace
attrib	grftabl	restore
backup	graphics	share
chkdsk	join	sort
command	keybxx	subst
diskcomp	label	sys
diskcopy	mode	tree
exe2bin	more	xcopy
fdisk	print	

Before MS-DOS can run external commands, it must read them into memory from the disk. When you give an external command, MS-DOS immediately checks your working directory to find that command. If it isn't there, you must tell MS-DOS which directory the external command is in. You do this with the **path** command.

When you are working with more than one directory, you may find it more convenient to put all the MS-DOS external commands in one directory. Then, when it needs them, MS-DOS can quickly find the external commands at one location.

Suppose, for example, that you are in a working directory named *\user\prog* and that the MS-DOS external commands are in *\bin*. To find the **format** command, you must tell MS-DOS to choose the *\bin* path, as in the following command, which tells MS-DOS to search in your working directory *and* in the *\bin* directory for all commands:

```
path \bin
```

You need only specify this path once during each computer session. Also, if you want to know what the current path is, you can simply type the **path** command by itself. In response, MS-DOS displays the working path on the screen.

Using paths with external commands

Using the path command

You can automatically set your path when you start MS-DOS by including the **path** command in a file called *autoexec.bat*. Refer to Chapter 4, "Batch Processing," for more information on the *autoexec.bat* file.

Redirecting Command Input and Output

Usually, MS-DOS receives input from the keyboard and sends its output to the screen. You can, however, redirect this flow of command input and output. For instance, you may want input to come from a file instead of from the keyboard, and you may want output from a command to go to a file or lineprinter instead of to the screen. With redirection you can also create *pipes* that let the output from one command become the input for another command.

How to Redirect Your Output

Redirecting output

By default, most commands send output to your monitor. If you want to change this and send the output to a file, you just use a greater-than sign (>) in your command. For example, the following command displays *on the screen* a directory listing of the disk in the default drive:

```
dir
```

The **dir** command can send this output to a file named *contents* if you type the following:

```
dir > contents
```

If the *contents* file doesn't exist, MS-DOS creates it and stores your directory listing there. If *contents* does exist, MS-DOS replaces what is in the file with the new data.

Appending output

If you want to append your directory or add one file to another (instead of replacing the entire file), you can use two greater-than signs (>>) to tell MS-DOS to append the output of the command (such as a directory listing) to the end of a specified file. For example, the following command appends your directory listing to an existing file named *contents*:

```
dir >> contents
```

If *contents* doesn't exist, MS-DOS creates it.

Often, it's useful to have input for a command come from a file instead of from the keyboard. This is possible in MS-DOS by using a less-than sign (<) in your command. For example, the following command sorts the file *names* and sends the sorted output to a file called *namelist*:

```
sort < names > namelist
```

Filters and Pipes

A *filter* is a command that reads your input, transforms it in some way, and then outputs it to your screen. In this manner the input is "filtered" by the program.

MS-DOS filters include: **find**, **more**, and **sort**. Their functions are as follows:

find	Searches for text in a file.
more	Displays the contents of a file one screenful at a time.
sort	Alphabetically sorts the contents of a file.

You can redirect the output from a filter into a file, or use it as input for another filter by using pipes. The following section explains how filters are piped together.

Command Pipes

If you want to use the output from one command as the input for another, you can *pipe* the commands to MS-DOS. Piping is done by separating commands with the pipe symbol, which is a vertical bar (|). The following command, for example, displays an alphabetically sorted listing of your directory on the screen:

```
dir | sort
```

The pipe sends all output generated by the **dir** command (on the left side of the bar) as input to the **sort** command (on the right side of the bar).

You can also use piping with redirection if you want to send the output to a file. For example, the following command creates a file named *direct.lst* on your default drive:

```
dir | sort > direct.lst
```

MS-DOS filter commands

Piping commands

The *direct.lst* file now contains a sorted listing of the directory on the default drive.

You can also specify a drive other than the default drive. Suppose, for example, you want to send the sorted data to a file named *direct.lst* on drive B. To do this you could simply type the following:

```
dir | sort > b:direct.lst
```

A *pipeline* may consist of more than two commands. The following command, for instance, sorts your directory, shows it to you one screenful at a time, and puts --More-- at the bottom of your screen when there is more output to be seen:

```
dir | sort | more
```

Since commands and filters can be piped together in many different ways, you will find many uses for them.

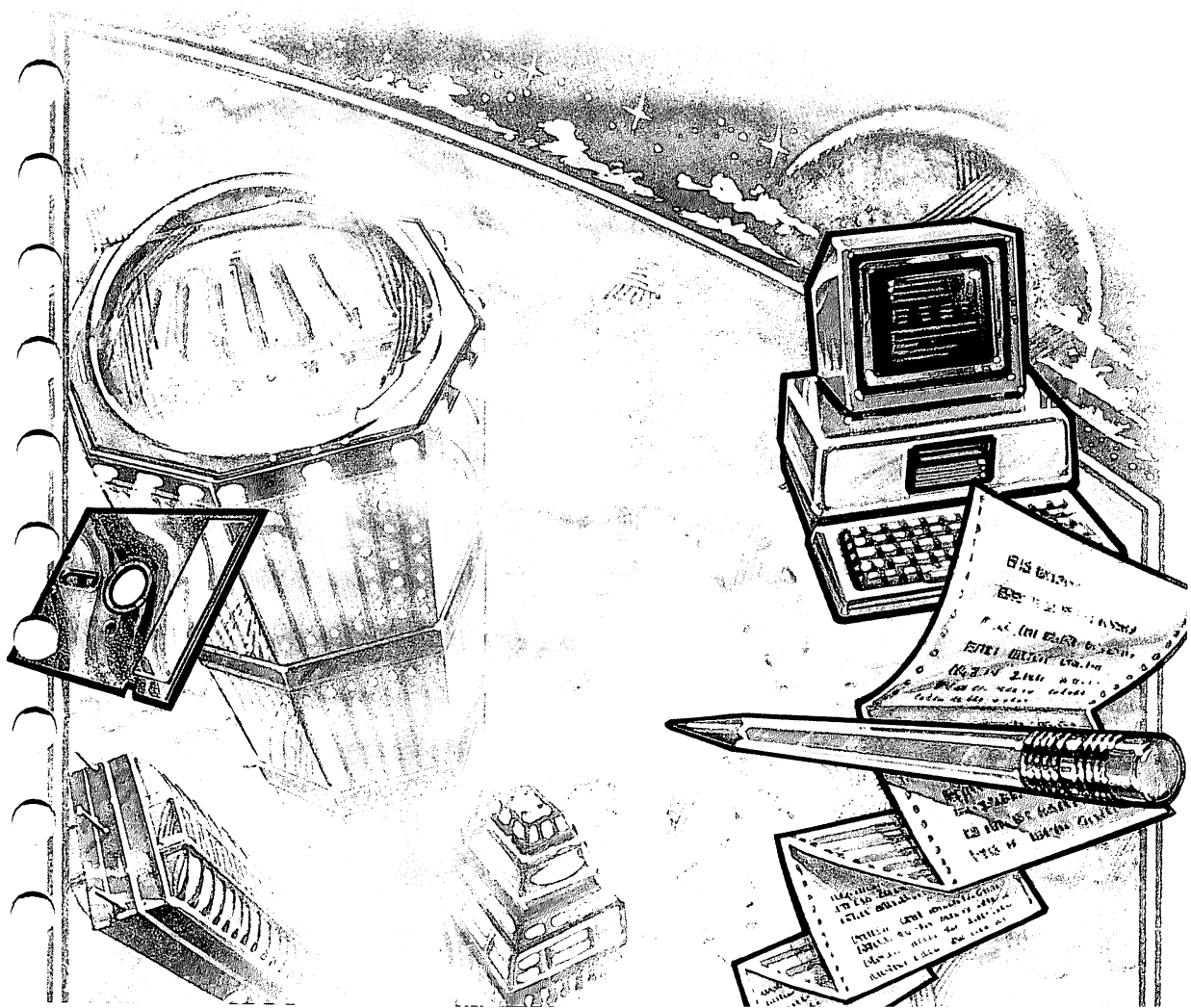
In this chapter you've learned about internal and external MS-DOS commands and how to redirect their input and output. The next chapter lists each of these commands in alphabetical order along with their syntax and comments about their use.

What's next

3 MS-DOS Commands

About This Chapter

This chapter lists the MS-DOS commands in alphabetical order. For each command the syntax is given along with detailed descriptions, comments, and examples.



External and internal commands are shown by the following symbols:



This shows that the command is internal.



This shows that the command is external.

Some of the MS-DOS commands do not work over a computer network. If you try to use these commands, MS-DOS displays the error message "Cannot *command* to a network device," where *command* is the name of the command you typed.

If the command does not work over a network (on a shared or remote device), you will see this symbol in the command description:



The commands that do not work over a network are:

chkdsk
diskcomp
diskcopy
fdisk
format

label
recover
subst
sys

What are command options?

Command Options

Command options give MS-DOS extra information about a command. If you do not include some options, MS-DOS provides values called *default values*. For the default values of particular commands, you should refer to individual command descriptions in this chapter.

MS-DOS commands use the following syntax:

command [*options*]

command is an MS-DOS command, and [*options*] is one or more of the following:

- drive:* Refers to a disk drive name. You need only specify a drive name if you are using a file that is *not* on the default drive.
- filename* Refers to any file and includes the filename extension (if there is one). The filename option does *not* refer to a device or drive name.
- pathname* Refers to a filename by the following syntax:
`[\directory][\directory...] \filename`
- path* Refers to a directory name by the following syntax:
`[\directory][\directory...] \directory`
- switches* Control MS-DOS commands. Switches begin with a slash; for example, /p.
- arguments* Provide more information to MS-DOS commands. You usually choose between arguments; for example: ON or OFF.

More About Options

The *MS-DOS User's Reference* uses the following conventions for command options:

- You must supply the text for any of the variable items shown in italics. For example, when *filename* is shown, you should type the name of your file.
- Items in brackets ([]) are optional. If you want to include optional information, you should only type the information within the brackets. Do not type the brackets.
- An ellipsis (...) means that you can repeat an item as many times as necessary.
- You must separate commands from their options by inserting certain characters or spaces. These characters are sometimes called separators. Generally, you should use spaces to separate commands from their options; for example:

```
rename dull.doc sharpe.doc
```

Conventions for MS-DOS command options

- You can also use a semicolon (;), an equal sign (=), or a tab to separate MS-DOS commands from their options.
In this manual, spaces separate commands from their options.
- You should use punctuation marks, such as backslashes, slashes, equal signs, quotation marks, or colons, where appropriate.
- Disk drives are referred to as *source* drives and *target* drives. A source drive is the drive *from* which you transfer information. A target drive is the drive *to* which you transfer information.
- Many commands manipulate *strings* of text. A string is a group of characters that can include letters, numbers, spaces, and any other characters. Searching for a particular word in a file is a common use of a string.

MS-DOS Commands

The following MS-DOS commands are described in this chapter. Note that synonyms for commands are in parentheses.

Note If you have only one floppy disk drive, refer to Appendix A, "Instructions for Users with Single Floppy Disk Drive Systems," before running any of the following commands.

append	Sets a search path for data files.
assign	Assigns a drive letter to a different drive.
attrib	Sets or displays file attributes.
backup	Backs up one or more files from one disk to another.
break	Sets CONTROL-C check.
chdir	Changes directories or prints the working directory (cd).
chkdsk	Scans the directory of the default or designated drive and checks for consistency.
cls	Clears the screen.
command	Processes internal MS-DOS commands.
copy	Copies the specified file(s).
date	Displays and sets the date.
del	Deletes the specified file(s) (erase).
dir	Lists the requested directory entries.
diskcomp	Compares disks.

diskcopy	Copies disks.
exe2bin	Converts executable (.exe) files to binary format.
exit	Exits the command processor and returns to the previous level.
fdisk	Configures hard disks for MS-DOS.
find	Searches for a constant string of text.
format	Formats a disk to receive MS-DOS files.
graftabl	Loads a table of graphics characters.
graphics	Prepares MS-DOS for printing graphics.
join	Joins a disk drive to a pathname.
keybxxx	Loads a keyboard program.
label	Labels disks.
mkdir	Makes a directory (md).
mode	Sets operation modes for devices.
more	Displays output one screen at a time.
path	Sets a command search path.
print	Prints files.
prompt	Allows you to change the prompt.
recover	Recovers a bad disk or file.
ren	Renames first file as second file (rename).
replace	Replaces previous versions of files.
restore	Restores backed up files.
rmdir	Removes a directory (rd).
set	Sets one string value to another in the environment, or displays the environment.
share	Installs file sharing and locking.
sort	Sorts data forward or backward.
subst	Substitutes a string for a pathname.
sys	Transfers MS-DOS system files from one drive to the drive specified.
time	Displays and sets the time.
tree	Displays directory and file names.
type	Displays the contents of a file.
ver	Prints the MS-DOS version number.
verify	Verifies all writes to a disk.
vol	Displays the volume label.
xcopy	Copies files and subdirectories.

These commands are described in detail in the following pages.

Append



Append**Purpose:**

Sets a search path for data files.

Syntax:

append [*drive*:][*path*][;[*drive*:][*path*]...]

or

append ;

Comments

The *path* parameter specifies the directory that MS-DOS searches for a data file. You can, however, specify more than one *path* to search by separating each with a semicolon (;).

If you don't use options with the **append** command, MS-DOS displays the current data path.

And if you use the following command, MS-DOS sets the NUL data path.

append ;

This means that MS-DOS searches only the working directory for data files.

Append searches the data path for all files, regardless of their file extensions, only with the following MS-DOS system calls:

Code	Function
0FH	Open File (FCB)
23H	Get File Size
3DH	Open Handle

You can use the **append** command across a network to locate remote data files.

Warning If you are using the MS-DOS **assign** command, you must use the **append** command before **assign**.

Example:

Suppose you want to access data files in a directory called *letters* on drive B, and in a directory called *reports* on drive A. To do this, use the following command:

```
append b:letters;a:reports
```

Assign



Assign

Purpose:

Assigns a drive letter to a different drive.

Syntax:

assign *x* = *y*

Comments:

The **assign** command lets you read and write files on drives other than A and B for applications that use only those two drives.

In the above syntax line, *x* is the drive that MS-DOS currently reads and writes to, and *y* is the new drive that you want MS-DOS to read and write to.

Note that you can use a space between the drive letters instead of the equal sign (=). Also, remember not to type a colon after the drive letters *x* and *y*.

Warning Because the **assign** command hides the true device type from commands that require actual drive information, you should not use this command with the **backup** or **print** commands or during normal use of MS-DOS. Also note that two other commands, **format** and **diskcopy**, ignore drive reassignments.

Examples:

To reset all drives back to their original assignments, type the **assign** command and press the RETURN key. Remember, though, you cannot assign a drive if it is being used by another program, and you cannot assign an undefined drive.

To ensure compatibility with future versions of MS-DOS, you should use the **subst** command instead of **assign**. The following commands, therefore, are equivalent:

```
assign a = c  
subst a: c:\
```

As a second example, the following command enables you to use drives other than A and B, such as a hard disk drive, C:

```
assign a=c b=c
```

All references to drives A and B would then go to drive C.

Setting drive assignments

Attrib

Purpose:

Sets or resets the read-only attribute of a file, or displays the attributes of a file.

Syntax:

attrib [**±r**] [**±a**] [*drive:*]*pathname*

Comments:

If an application opens a file with read *and* write permission, **attrib** forces read-only mode to allow file sharing over a network.

- +**r** sets the read-only attribute of a file.
- r** disables read-only mode.
- +**a** sets the archive attribute of a file.
- a** clears the archive attribute of a file.

[*drive:*]*pathname* specifies the name of the drive and the full pathname of the file you want to reference.

The **backup**, **restore**, and **xcopy** (**mcopy**) commands use the archive attribute to control a selective **backup/restore/xcopy** on files that have been modified. You can use the **+a** and **-a** options to select files that you want to back up with the **backup/m** switch, or copy with the **xcopy** (or **mcopy**) **/m** or **/a** switches.

To display the attribute of a specific file on the default drive, type the **attrib** command followed by the *pathname* of the file. To print the attributes of all files in a specific directory, use the wildcard abbreviation *.* in the pathname.

Examples:

The following example gives the file *report.txt* read-only permission:

```
attrib +r report.txt
```

Suppose you want to use the **xcopy** command to copy all the files in the default directory of drive A, except for those that have an extension of *.bak*, to drive B. You would type the following:

```
attrib +a a:*. *
attrib -a a:*.bak
```

Attrib



Changing read and write permission

and:

`xcopy a: b: /m`

or:

`copy a: b: /a`

If you use the **/m** switch, **xcopy** automatically turns off the archive bits of the files in drive A as it copies them.

Backup

Purpose:

Backs up one or more files from one disk to another.

Syntax:

backup [*drive:*][*pathname*] [*drive:*] [/s][/m][/a]
[/d:*date*] [/t:*time*][/L:*filename*]

Comments:

The **backup** command can back up files on disks of different media. For example, you can back up files from:

- Hard disk to floppy disk
- Floppy disk to floppy disk
- Floppy disk to hard disk
- Hard disk to hard disk

This command also backs up files from one floppy disk to another even if the disks have a different number of sides or sectors.

The first *drive:* option is the name of the disk drive that you want to back up. The second *drive:* option is the name of the backup disk drive. You should note that unless you specify the /a option, **backup** erases the old files on a backup disk before adding new files to it.

This **backup** program and the one supplied by IBM® are compatible. File and disk formats are the same as those that the IBM **backup** program uses unless you set the /L switch, which is described in the following list.

Backup



The Backup switches

You can specify the following switches with **backup**:

- /s** Backs up subdirectories also.
- /m** Backs up only those files that have changed since the last backup.
- /a** Adds the files to be backed up to those already on the backup disk. It does not erase old files on the backup disk.
- /d** Backs up only those files that you last modified on or after *date*.
- /t** Backs up only those files that you last modified at or after a certain *time*.
- /L** Makes a backup log entry in the specified file. If you do not specify a filename, **backup** places a file called *backup.log* in the root directory of the disk that contains the files being backed up. The first line of the entry in the file contains [*date time*] where *date* and *time* are the backup dates and times. Each subsequent line in the entry corresponds to a backed up file. These lines each consist of a filename and the number of the disk that contains the file. You can use this information when you need to restore a particular file from a floppy disk, but you must specify which disk to restore so that **backup** does not have to search for files. If the backup log file already exists, **backup** appends the current entry to the file.

Backup displays the name of each file as it is backed up. You should label and number each backup disk consecutively to help you restore the files properly with the **restore** command.

If you are sharing files, you can back up only the files that you have access to.

Note You should not use the **backup** command if the drive you are backing up has been assigned, joined, or substituted with the **assign**, **join**, or **subst** command. If you do, you may not be able to restore the files with the **restore** command.

The **backup** program returns the following exit codes:

- 0 Normal completion
- 1 No files were found to back up
- 2 Some files not backed up due to sharing conflicts
- 3 Terminated by user
- 4 Terminated due to error

You can use the batch processing **if** command for error processing that is based on the *errorlevel* returned by **backup**.

Example:

Suppose Emily wants to backup all of the files in the `\user\emily` directory on drive C to a blank, formatted disk in drive A. To do this, she would type the following:

```
backup c:\user\emily a:
```

Exit codes

Backing up files

Break

Break**Purpose:**

Sets CONTROL-C check.

Syntax:

break [ON]

or

break [OFF]

Comments:

Depending on the program you are running, you may use CONTROL-C to stop an activity (for example, to stop sorting a file). Normally, MS-DOS checks to see whether you press CONTROL-C while it is reading from the keyboard or writing to the screen or printer. If you set **break** to ON, you extend CONTROL-C checking to other functions such as disk reads and writes.

Examples:

To check for CONTROL-C only during screen, keyboard, and printer reads and writes, type the following:

```
break off
```

To find out how **break** is currently set, type the **break** command and press the RETURN key.

Note Some programs may set themselves to respond to CONTROL-C at any time. Setting **break** does not affect these programs.

CONTROL-C checking

Chdir

Synonym:

`cd`

Purpose:

Changes a directory to a different path; displays the working directory.

Syntax:

chdir [*path*]

Comments:

If your working directory is `\user\emily` and you want to change your path to another directory (such as `\user\pete`), type the following command and press the RETURN key:

```
chdir \user\pete
```

MS-DOS puts you in the new directory. There is also a shorthand notation for the **chdir** command:

```
cd ..
```

This command always puts you in the parent directory of your working directory.

Examples:

If you use **chdir** without a path, you can display the name of your working directory. For example, if your working directory is `\user\pete` on drive B, and you type the **chdir** command, then press the RETURN key, MS-DOS displays the following:

```
b:\user\pete
```

The following command displays the name of the working directory on drive B:

```
chdir b:
```

Chdir (change directory)



Displaying the name of your working directory

Chkdsk (check disk)

Chkdsk**Purpose:**

Scans the disk in the specified drive and checks it for errors.

Syntax:

chkdsk [*drive:*] [*pathname*] [/f] [/v]

Comments:

You should run **chkdsk** occasionally on each disk to check for errors. If you do run **chkdsk** on a disk and any errors are found, **chkdsk** displays the error messages, followed by a status report.

Chkdsk status report

A typical status report might look like this:

```
160256      bytes total disk space
8192        bytes in 2 hidden files
512         bytes in 2 directories
30720       bytes in 8 user files
121344      bytes available on disk
```

```
65536 bytes total memory
53152 bytes free
```

The Chkdsk switches

The /f switch fixes errors on the disk. If you do not specify this switch, **chkdsk** does not correct errors that it finds in your directory, though it does display messages about files that need to be fixed.

If you specify the /v switch, **chkdsk** displays messages while it is running.

If you type a filename after **chkdsk**, MS-DOS displays a status report for the disk and for the individual file.

Note **Chkdsk** does not correct errors on a disk unless you specify the /f switch. For more information on **chkdsk** errors, refer to the specific error message in Appendix E, "MS-DOS Message Directory."

Examples:

Saving a Chkdsk status report

If you want to save the **chkdsk** status report for future use, you can redirect the output from **chkdsk** to a file by typing the following:

```
chkdsk a:>filename
```

The errors are then sent to the specified file. Remember, though, not to use the /f switch when you redirect **chkdsk** output.

If **chkdsk** finds errors on the disk in drive A and you want to try to correct them, type the following command:

```
chkdsk a: /f
```

Chkdsk now tries to correct any errors it encounters on the disk in drive A.

Cls



Cls

Purpose:

Clears the terminal screen.

Syntax:

`cls`

Comment:

This command clears your terminal screen by sending the ANSI escape sequence (ESC[2J) to your console.

Example:

To clear your screen, type the following:

`cls`

Clearing your screen

Command

Purpose:

Starts the command processor.

Syntax:

command [*drive:*][*path*][*cttydev*] [/e:*nnnnn*][/*p*] [/c *string*]

Comments:

This command starts a new command processor (the MS-DOS program that contains all internal commands).

When you start a new command processor you also create a new command environment. This new environment is a copy of the old, parent environment. However, you can change the new environment without affecting the old one.

The command processor is loaded into memory in two parts: *transient* and *resident*. Some applications write over the transient memory part of *command.com* when they run. When this happens, the resident part of the command processor looks for the *command.com* file on disk so that it can reload the transient part.

The *drive:path* options tell the command processor where to look for the *command.com* file if it needs to reload the transient part into memory.

Cttydev allows you to specify a different device (such as AUX) for input and output. See the **ctty** command in this chapter for more information.

The /e switch specifies the environment size, where *nnnnn* is the size in bytes. The size may range between 128 and 32768 bytes. The default value is 128 bytes.

If *nnnnn* is less than 128 bytes, MS-DOS defaults to 128 bytes and displays the following message:

```
Invalid environment size specified
```

If *nnnnn* is greater than 32768 bytes, MS-DOS displays the same message, but defaults to 32768 bytes.

The /p switch tells *command.com* not to exit to any higher level shell.

The /c switch, if used, should be the last switch in the command. It tells the command processor to perform the command or commands specified by *string* and to then return.

Command



Starting a new command processor

Example:

The following command tells the MS-DOS command processor to do three things:

- Start a new command processor under the current program
- Run the command "chkdsk b:"
- Return to the first command processor

```
command /c chkdsk b:
```

To learn how to use a pathname and the **/p** switch with **command**, see the sample *config.sys* file in Appendix B, "How to Configure Your System."

Copy

Purpose:

Copies one or more files to another disk. This command also appends files and copies files on the same disk.

Syntax:

copy [*drive:*]*pathname* [*drive:*]*pathname* [/v] [/a] [/b]
(to copy files)

copy *pathname* + *pathname* ...
(to append files)

Comments:

If you do not specify the second *pathname*, the copy is created on the default drive and has the same name as the original file (first *pathname*). If the original file is on the default drive and you do not specify the second *pathname*, the **copy** command quits (you are not allowed to copy a file to itself), and MS-DOS displays the following error message:

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

Note If the source and target files are both in the working directory, you may use filenames instead of complete pathnames.

The second *drive:pathname* option may take one of three forms:

- If the second option is a drive name only, MS-DOS copies the original file to the designated drive, keeping the original filename. For example, the following command makes a copy on drive B named *memo.doc*:
- If the second option is a filename only, MS-DOS copies the original file to one on the default drive, and renames it with the specified filename. For example, the following command makes a copy of *memo.doc*, names it *letter.doc*, and places it on the default drive:

```
copy memo.doc b:
```

```
copy memo.doc letter.doc
```

Copy



Copying a file

- If the second option includes a drive name, MS-DOS copies the original file to one on the specified drive. For example, the following command makes a copy of *memo.doc* on the default drive, names the copy *letter.doc*, and places the copy on the disk in drive B:

```
copy memo.doc b:letter.doc
```

The **/v** switch causes MS-DOS to verify that the sectors written on the target disk are recorded properly. If MS-DOS cannot verify a write, it displays an error message. Although there are rarely recording errors when you run **copy**, the **/v** switch lets you verify that critical data has been correctly recorded; it also makes the **copy** command run more slowly because MS-DOS must check each entry recorded on the disk.

The **/a** or **/b** switch lets you copy either ASCII or binary files, respectively. Each switch applies to the filename preceding it, and to all remaining filenames in the command, until **copy** encounters another **/a** or **/b** switch.

Examples:

When used with a source filename:

- /a** Causes the file to be treated as an ASCII (text) file. Data in the file is copied up to but not including the first end-of-file mark (in **edlin** this is CONTROL-Z). The remainder of the file is not copied.
- /b** Causes the entire file to be copied, including any end-of-file marks.

When used with a target filename:

- /a** Causes an end-of-file character to be added as the last character of the file; for example:

```
copy memo.doc /a letter.doc
```
- /b** Does not add an end-of-file character; for example:

```
copy billing.asm /b billing2.asm
```

When you are appending files the default switch is always **/a**.

Appending files

The **copy** command also allows you to append files. To do this you simply list any number of files as options to **copy**, each separated by a plus sign (+), then specify a target file to send the combined files to; for example:

```
copy intro.rpt + body.rpt + b:sum.rpt report
```

This command combines files named *intro.rpt*, *body.rpt*, and *sum.rpt* (on drive B), and places them in a file called *report* on the default drive. If you leave out the target file, MS-DOS combines the files into the first specified file.

You can also combine several files into one by using wildcards; for example:

```
copy *.txt combin.doc
```

This command takes all files with an extension of *.txt* and combines them into one file named *combin.doc*.

In the following example, each file that matches **.txt* is combined with its corresponding *.ref* file. The result is a file with the same filename but with the extension *.doc*. Thus, *file1.txt* is combined with *file1.ref* to form *file1.doc*, *xyz.txt* with *xyz.ref* to form *xyz.doc*, and so on:

```
copy *.txt + *.ref *.doc
```

The following **copy** command combines all files matching **.txt* and all files matching **.ref* into one file named *combin.doc*:

```
copy *.txt + *.ref combin.doc
```

Warning Do not try to append files if one of the source filenames has the same extension as the target. For example, if the file *all.txt* already exists, the following command is an error:

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

MS-DOS would not detect the error until it tried to append *all.txt*. But at that point, **copy** might have already destroyed the *all.txt* file.

Copy compares the filename of the input file with the filename of the target. If they are the same, that one input file is skipped, and MS-DOS prints the error message "Content of destination lost before copy." Further joining proceeds normally. For example, the following command appends all *.txt files (except *all.txt*) to *all.txt*:

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

This command will not produce an error message.

If you want to copy files and subdirectories, you should use the **xcopy** command. Refer to the **xcopy** command in this chapter for more information on how to do this.

Copying files and subdirectories

Ctty

Ctty



Purpose:

Lets you change the device from which you issue commands.

Syntax:

ctty *device*

Comments:

The *device* parameter specifies the device from which you are giving commands to MS-DOS. **Ctty** is useful if you want to change the device on which you are working. In this command, the letters **tty** represent the console; that is, your keyboard.

Examples:

The following command moves all command I/O (input/output) from the current device (the console) to an AUX port such as another terminal:

```
ctty aux
```

The next command moves I/O back to the console:

```
ctty con
```

Note There are many programs that do not use MS-DOS for input, output, or either. 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.

Date**I**

Date**Purpose:**

Enters or changes the date known to the system.

Syntax:

date [*mm-dd-yy*]

Comments:

You can change the date from your terminal or from a batch file. (MS-DOS does not automatically display a prompt for the date if you use an *autoexec.bat* file, so you may want to include a **date** command in that file.) MS-DOS records this date in the directory when you create or change a file.

Remember to use only numbers when you type the date; allowed numbers are:

mm = 1–12

dd = 1–31

yy = 80–99 or 1980–2099

The date, month, and year entries may be separated by hyphens (-) or slashes (/). MS-DOS is programmed to change months and years correctly, whether the month has 31, 30, or 28 days—or 29 days, since MS-DOS handles leap years, too.

It is possible for you to change the *mm-dd-yy* format in which the date is displayed and entered. The **country** command in the *config.sys* file allows you to change the date format to the European standard *dd-mm-yy*. For more information on the *config.sys* file, see Appendix B, "How to Configure Your System."

Examples:

Displaying the current date

If you simply type the **date** command, MS-DOS displays the following message:

```
Current date is weekday mm-dd-yy
Enter new date (mm-dd-yy):_
```

If you do not want to change the date shown, press the RETURN key. Or you can type a particular date after the **date** command, as in the following example:

```
date 3-9-86
```

In this case the "Enter new date:" prompt does not appear after you press RETURN.

Del

Synonym:

erase

Purpose:

Deletes all files specified by the drive and pathname.

Syntax:

del [*drive:*]*pathname*

Comments:

You can use the * and ? wildcards to delete more than one file at a time.

If the *pathname* is *.* the prompt "Are you sure?" appears. If you then type Y as a response, all files on the disk are deleted.

Note Once you have deleted a file on your disk, you cannot recover it.

Examples:

The following command deletes a file named *vacation*:

```
del vacation
```

If you have two files named *vacation.feb* and *vacation.apr*, you can delete them both with the following command:

```
del vacation.*
```

Del (Delete)

I

Deleting a file

Dir (Directory)**I**

Dir**Purpose:**

Lists the files in a directory.

Syntax:

dir [*drive*:[*pathname*][*/p*][*/w*]

Comments:

If you just type the **dir** command by itself, all directory entries on the default drive are listed. If you include a drive name in the command, such as *b:*, all entries in the default directory of the disk in drive B are listed. If you include a filename without an extension (*invoices*, for example), MS-DOS lists all files named *invoices* in the default directory of the disk in the default drive.

When you use the **dir** command with a filename and drive letter (*b:invoices*, for example), MS-DOS displays all files on the disk in drive B with the filename *invoices*. In all cases, (except when using the */w* switch), MS-DOS lists files with their size in bytes and with the time and date of their last modification.

Note that the following **dir** commands are equivalent, since you can use the wildcards ? and * in the *pathname* option:

Command	Equivalent
dir	dir *.*
dir filename	dir filename.*
dir .ext	dir *.ext

The Dir switches

You may use the */p* switch and the */w* switch with **dir**.

The */p* switch used with **dir** selects page mode. It also makes the directory display stop scrolling when the screen has filled. To resume scrolling the display, press any key.

The */w* switch selects wide display and causes MS-DOS to display only filenames and not other file information. The wide display lists up to five files per line.

Note that if the **country** command in the *config.sys* file is set to a country other than the U.S., the directory date and time formats may differ. For more information on the *config.sys* file, See Appendix B, "How to Configure Your System."

Example:

If your directory contains more files than you can see on the screen at one time, type the following:

```
dir /p
```

This command displays the directory one screenful at a time.

Diskcomp

Diskcomp**Purpose:**

Compares the contents of the disk in the source drive to the disk in the target drive.

Syntax:

diskcomp [*drive:*] [*drive:*] [/1] [/8]

Comments:

The first *drive:* option specifies the drive that contains the source disk, and the second *drive:* option specifies the drive that contains the target disk.

If you specify only one drive, **diskcomp** uses the default drive as the target drive. If you specify the same drive as the source and target, **diskcomp** does a comparison using one drive, and prompts you to insert the disks as appropriate.

The /1 switch causes **diskcomp** to compare just the first side of the disk, even if the disks and drives that you are using are double-sided.

The /8 switch causes **diskcomp** to compare just the first 8 sectors per track, even if the disks contain 9 or 15 sectors per track.

Diskcomp performs a track-by-track comparison of the disks. It automatically determines the number of sides and sectors per track based on the format of the source disk. If the target disk is not the same type as the disk in the source drive, **diskcomp** displays the following message:

```
Drive types or diskette types
not compatible
```

If all the tracks are the same, **diskcomp** displays the message:

```
Compare OK
```

If the tracks are not the same, **diskcomp** displays a "Compare error" message that includes the track and side number where it found the mismatch.

When **diskcomp** completes the comparison, it prompts you with the following message:

```
Compare another diskette (Y/N)?_
```

The Diskcomp switches

If you type **Y**, **diskcomp** prompts you to insert the proper disks and does the next comparison. If you type **N**, **diskcomp** ends. If the disk in the default drive does not contain MS-DOS and you end **diskcomp**, you'll receive the following message:

Insert disk with COMMAND.COM in drive A
and strike any key when ready

Diskcomp does not work on network drives, and you cannot use it with assigned, joined, or substituted drives. If you attempt to use the **diskcomp** command with these types of drives, it displays an error message.

Note When comparing a disk with a backup disk that you made with the **copy** command, you may receive the "Compare error" message, even if the files on the disks are identical. This is because the **copy** command duplicates the information, but doesn't necessarily place it in the same location on the target disk. In this case, you should use the FC utility to compare individual files on the disk. See Chapter 8, "File Comparison Utility (FC)."

When correctly written programs exit back to MS-DOS, they return an exit code: 0 if no error occurred, or a value greater than zero if there was a problem. This exit code can be tested in batch files, and it allows batch programmers to "branch" to an error-handling routine in the batch file.

The **diskcomp** command returns the following exit codes:

- | | |
|---|---|
| 0 | Compared OK
The disks compared exactly. |
| 1 | Did not compare
The disks were not the same. |
| 2 | CONTROL-C error
The user terminated with CONTROL-C. |
| 3 | Hard error
An unrecoverable read or write error occurred — did not compare. |
| 4 | Initialization error
There is not enough memory — invalid drives or command line syntax. |

You can use the batch processing **if** command to perform error processing based on the *errorlevel* returned by **diskcomp**.

Diskcomp exit codes

Comparing two disks

Example:

If you want to compare two disks, but your computer has only one floppy disk drive, drive A, you can simply type the following command:

```
diskcomp a:
```

MS-DOS prompts you to insert both disks, as required.

Diskcopy

Purpose:

Copies the contents of the disk in the source drive to the disk in the target drive.

Syntax:

diskcopy [*drive:*] [*drive:*]

Comments:

The first *drive:* option is the source drive. The second *drive:* option is the target drive.

If you omit both options, MS-DOS performs a single-drive copy operation on the default drive. If you omit just the second option, MS-DOS uses the default drive as the target drive. In either case, though, **diskcopy** destroys the contents of the target disk.

Diskcopy prompts you to insert the source and target disks at appropriate times and waits for you to press any key before continuing.

After copying, **diskcopy** then prompts you with the following message:

Copy another diskette (Y/N)?_

If you press *Y*, MS-DOS prompts you to insert source and target disks, and performs the next copy on the drives that you originally specified.

To end the **diskcopy** process, press *N*.

Because disk space is not allocated sequentially, disks that have had a lot of files created and deleted on them become fragmented. So, the first free sector found by **diskcopy** becomes the next sector allocated, regardless of its location on the disk.

A fragmented disk can delay finding, reading, or writing a file. To prevent further fragmentation, you should use either the **copy** command or the **xcopy** command to copy your disk, instead of using the **diskcopy** command. Because the **copy** and **xcopy** commands copy files sequentially to a disk, the new disk will not be fragmented.

The following command, for example, copies all files from the disk in drive A to the disk in drive B:

```
xcopy a:*. * b:
```

Diskcopy



Diskcopy figures out the number of sides to copy, based on the source drive and disk.

Diskcopy exit codes

When correctly written programs exit back to MS-DOS, they return an exit code: 0 if no error occurred, or a value greater than zero if there was a problem. This exit code can be tested in batch files, and it allows batch programmers to "branch" to an error-handling routine in the batch file.

- | | |
|---|---|
| 0 | Copied successfully |
| 1 | Non-fatal read/write error
An unrecoverable but non-fatal read or write error occurred. |
| 2 | CONTROL-C error
The user entered CONTROL-C to terminate diskcopy . |
| 3 | Fatal hard error
Diskcopy was unable to read the source disk or format the target disk. |
| 4 | Initialization error
There is not enough memory — invalid drives or command line syntax. |

You can use the batch processing **if** command to perform error processing based on the *errorlevel* returned by **diskcopy**.

Example:

Copying a disk

To copy the disk in drive A to the disk in drive B, use the following command:

```
diskcopy a: b:
```

Diskcopy prompts you to insert both disks and press any key to begin copying.

Exe2bin

Exe2bin



Purpose:

Converts *.exe* (executable) files to binary format.

Syntax:

exe2bin [*drive:*]*pathname* [*drive:*]*pathname*

Comments:

This command converts *.exe* (executable) files to binary format. The first *pathname* is the input file; if you do not specify an extension, it defaults to *.exe*. The input file is converted to a *.bin* file format (a memory image of the program) and placed in the output file (the second *pathname*).

If you do not specify a drive name, **exe2bin** uses the drive of the input file. Similarly, if you do not specify an output filename, **exe2bin** uses the input filename. And finally, if you do not specify a filename extension in the output filename, **exe2bin** gives the new file the extension *.bin*.

Some restrictions do apply when you use the **exe2bin** command: the input file must be in valid *.exe* format produced by the linker; the resident, or actual code and data part of the file must be less than 64K; and there must be no STACK segment.

With **exe2bin**, two kinds of conversions are possible, depending on whether the initial CS:IP (Code Segment:Instruction Pointer) is specified in the *.exe* file:

- If the CS:IP is not specified in the *.exe* file, **exe2bin** assumes you want a pure binary conversion. If segment fixups are necessary (that is, if the program contains instructions requiring segment relocation), the command **E** prompts you for the fixup value. This value is the absolute segment at which the program is to be loaded. The resulting program will be usable only when loaded at the absolute memory address specified by your application. The command processor will not be able to load the program.



- If the **CS:IP** is 0000:100H, **exe2bin** assumes that the file will run as a *.com* file with the location pointer set at 100H by the assembler statement **ORG** (the first 100H bytes of the file are deleted). No segment fixups are allowed, since *.com* files must be segment relocatable; that is, they must assume the entry conditions explained in the Microsoft Macro Assembler manuals (*User's Guide* and *Reference Manual*). Once the conversion is complete, you may rename the output file with a *.com* extension. The command processor will then be able to load and execute the program in the same way as the *.com* programs supplied on your MS-DOS disk.

Exit

Purpose:

Exits the **command.com** program (the command processor) and returns to a previous level, if one exists.

Syntax:

exit

Comments:

If you use the MS-DOS **command** program to start a new command processor, you can use the **exit** command to return to the old command processor. Also, while running an application program, you can exit to the MS-DOS command processor, and then return to your program.

Refer to the **command** command in this chapter for more information.

Example:

If you start a new command processor by typing the following command, you can then return to the previous command processor by typing the **exit** command:

```
command c:\
```

Exit

I

Fdisk

Fdisk**Purpose:**

Configures hard disks for MS-DOS.

Syntax:

fdisk

Comments:

Fdisk configures a hard disk for use with MS-DOS. Before you can use your hard disk for the first time, you must use **fdisk** to configure it. (Your dealer may already have done this step.)

Fdisk displays a series of menus to help you partition your hard disk for MS-DOS.

For more information on how to use **fdisk**, see Appendix F, "Configuring Your Hard Disk (Fdisk)."

Find

Purpose:

Searches for a specific string of text in a file or files.

Syntax:

find [/v] [/c] [/n] "*string*" [drive:][*pathname*]

Comments:

Find is a filter that takes as options a *string* (a group of characters) and a series of filenames. After searching the files given on the command line, **find** displays any lines it has found that contain the specified string. In your command line, this *string* must be enclosed in double quotation marks.

If you do not specify a *pathname*, **find** takes input from its standard input (usually the keyboard) and displays any lines that contain the specified string.

The switches for **find** are as follows:

- /v Displays all lines *not* containing the specified string.
- /c Prints only the count of lines that contain a match in each of the files.
- /n Precedes each line by its relative line number in the file.

Note You must type double quotation marks around a string that already has double quotation marks.

Examples:

The following command displays all lines from a file named *pencil.ad* that contain the string "Automatic Pencil Sharpener":

```
find "Automatic Pencil Sharpener" pencil.ad
```

The next command causes MS-DOS to display all names of the files on the disk in drive B that do not contain the string "date":

```
dir b: | find /v "date"
```

Find



The Find switches

Format**Format****Purpose:**

Formats the disk in the specified drive to accept MS-DOS files.

Syntax:

format *drive*:[/1][/4][/8] [/n:xx][/t:yy][/v][/s]

or

format *drive*:[/1][/b][n:xx] [/t:yy]

Comments:

The **format** command initializes the directory and the file allocation tables on a disk. You must use this command to format all new disks before MS-DOS can use them.

When using the commands, you must specify the drive that you want to format. **Format** then uses the drive type to determine the default format for a disk.

When you format a hard disk, **format** prompts you to verify the volume label:

Enter current Volume Label for drive x:

If your hard disk does not have a volume label, press the RETURN key. (Note: If your hard disk has never been formatted before, or if it has a bad boot sector, **format** will not prompt you for a volume label.)

If the volume label that you enter does not match the label on the hard disk, **format** displays the following message:

Invalid Volume ID Format failure

Otherwise, it continues:

WARNING, ALL DATA ON NON-REMOVABLE DISK
DRIVE x WILL BE LOST!
Proceed with Format (Y/N)?_

If you want to format your hard disk, type *Y* and press the RETURN key. If not, type *N* and press the RETURN key.

The */1* switch formats a disk for single-sided use, even if the disk or drive is double-sided. If the drive is double-sided and you don't specify this switch, you won't be able to use the formatted disk in a single-sided drive.

Using the Format switches

The **/4** switch formats a double-sided disk in a high-capacity disk drive. Note, however, that if you are using a single- or double-sided drive, you may not be able to reliably read disks formatted with this switch.

The **/8** switch formats a disk for 8 sectors per track. If you do not specify this switch, **format** defaults to either 9 or 15 sectors per track (depending on the type of drive being used). Note that **format** *always* creates either 9 or 15 sectors per track; when you specify this switch, though, it tells MS-DOS to *use* only 8 sectors per track.

The **/b** switch formats a disk with 8 sectors per track and allocates space for the hidden system files. If you use this switch with the **format** command, you can place any version of MS-DOS on the disk by using that version's **sys** command. If you don't use this **/b** switch, you can place only MS-DOS 3.2 on the disk by using the **sys** command. You cannot use the **/s** or the **/v** switch with the **/b** switch.

The **/n:xx** option specifies the number of sectors per track that **format** uses to format a floppy disk.

The **/t:yy** option specifies the number of tracks that **format** places on a floppy disk.

The **/v** switch prompts for a volume label after the disk is formatted. A volume label identifies the disk and can be up to 11 characters in length. An example of a volume label is *PROGRAMS*.

If you use the **/s** switch, it must be the last switch that you type. This switch copies the operating system files from the disk in the default drive to the newly formatted disk. The files are copied in the following order:

io.sys
msdos.sys
command.com

If the operating system is not on the default drive, **format** prompts you to insert a system disk in the default drive (or in drive A if the default drive is non-removable).

When formatting is complete, **format** displays a message showing the total disk space, any space marked as defective, the total space used by the operating system (when you use the **/s** switch), and the space available for your files.

Copying the operating system

The following table shows which switches you can use for certain types of disks:

Disk type	Valid switches
160/180KB	/1 /4 /8 /b /n /t /v /s
320/360KB	/1 /4 /8 /b /n /t /v /s
720KB	/n /t /v /s
1.2KB	/n /t /v /s
hard disk	/v /s

Note Formatting destroys any previously existing data on a disk and ignores drive assignments created with the **assign** command. You should not format disks on drives used in the **assign**, **join**, or **subst** commands, and you cannot **format** disks over a network.

Format exit codes

Format returns the following exit codes:

- 0 Successful completion
- 3 Terminated by user (CONTROL-C)
- 4 Fatal error (any error other than 0, 3, or 5)
- 5 N response to hard disk prompt, "Proceed with format (Y/N)?"

You can check these exit codes by using the *errorlevel* condition with the **if** batch processing command.

Examples:

Formatting a floppy disk

To format a floppy disk in drive A and put the operating system on it, type the following:

```
format a: /s
```

And to format a floppy disk in drive A for use with data, type the following:

```
format a: /v
```

The **/v** switch causes **format** to prompt you for a volume label. You should type a label since it will help you identify the data that the disk contains.

Graftabl

Purpose:

Loads additional character data into a table in memory for use with a color or graphics adapter.

Syntax:

graftabl

Comments:

Graftabl loads a table of the ASCII characters, 128 through 255, into memory. If you have a color or graphics adapter, this table lets you display foreign language characters when you are in graphics mode.

After **graftabl** loads the character table, it displays the following message:

GRAPHICS CHARACTERS LOADED

You can load the graphics table only once each time you start MS-DOS. If you try to load the table a second time, **graftabl** displays the following message:

GRAPHICS CHARACTERS ALREADY LOADED

Note This command increases the size of MS-DOS resident in memory.

Example:

To load the graphics table into memory, type the following:

graftabl

Graftabl



**Loading the graphics
table into memory**

Graphics



The Graphics printer options

The Graphics switches

Graphics

Purpose:

Lets you print a graphics display screen on a printer when you are using a color or graphics monitor adapter.

Syntax:

graphics [*printer*] [/r] [/b]

Comments:

The *printer* option may be one of the following:

COLOR1	Prints on an IBM Personal Computer Color Printer with black ribbon.
COLOR4	Prints on an IBM Personal Computer Color Printer with RGB (red, green, blue, and black) ribbon.
COLOR8	Prints on an IBM Personal Computer Color Printer with CMY (cyan, magenta, yellow, and black) ribbon.
COMPACT	Prints on an IBM Personal Computer Compact Printer.
GRAPHICS	Prints on an IBM Personal Graphics Printer.

If you do not specify the *printer* option, **graphics** defaults to the GRAPHICS printer type.

The /r switch prints black and white (as seen on the monitor) on the printer. The default is to print black as white and white as black.

The /b switch prints the background in color. This option is valid for COLOR4 and COLOR8 printers.

To print the screen, press the SHIFT and PRINTSCREEN keys at the same time. If the computer is in 320 × 200 color graphics mode, and if the printer type is COLOR1 or GRAPHICS, **graphics** prints the screen contents with up to four shades of gray. If the computer is in 640 × 200 color graphics mode, **graphics** prints the screen contents sideways on the paper.

Note This command increases the size of MS-DOS resident in memory.

Example:

To print a graphics screen on your printer, type the following command:

`graphics`

Then, when the screen displays the information you want to print, press the SHIFT and PRINTSCREEN keys at the same time.

Printing a graphics screen

Join

Join**Purpose:**

Joins a disk drive to a specific path.

Syntax:

join [*drive: drive:path*]

or

join drive: /d

Comments:

With the **join** command you don't need to name physical drives with separate drive letters. Instead, you can refer to all the directories on a specific drive with one path. If the path already existed before you gave the **join** command, you cannot use it while the "join" is in effect. Also, you cannot join a drive if it is being used by another process.

If the *path* does not exist, MS-DOS tries to make a directory with that path. After you give the **join** command, the first drive name becomes invalid, and if you try to use it MS-DOS displays the "Invalid drive" error message.

Examples:

You can join a drive only with a root level directory. For example, this command will work:

```
join d: c:\sales
```

But the following one will *not*:

```
join d: c:\sales\regional
```

To reverse **join** ("unjoin"), use the following format:

```
join drive: /d
```

Here *drive:* represents the source drive, and the **/d** switch turns off the **join** command.

If you just type the **join** command by itself, MS-DOS displays the current drives that are joined.

Joining a drive

Keybxx

Purpose:

Loads a keyboard program.

Syntax:

keybxx

Comments:

xx is one of the following two letter codes:

Code	Keyboard type	Command
DV	Dvorak	keybdv
FR	France	keybfr
GR	Germany	keybgr
IT	Italy	keybit
SP	Spain	keybsp
UK	United Kingdom	keybuk

You should load only one keyboard program after starting MS-DOS.

You can switch from the **keybxx** program to the default (United States) keyboard format at any time by pressing CONTROL-ALT-F1.

You can then return to the memory-resident keyboard program by pressing CONTROL-ALT-F2.

Example:

To use a German keyboard, type the following command:

keybgr

Keybxx



Changing keyboard programs

Label**Label****Purpose:**

Creates, changes, or deletes the volume identification label on a disk.

Syntax:

label [*drive:*][*label*]

Comments:

The volume label may be up to 11 characters in length and may include spaces, but not tabs. Aside from tabs, you also should not use the following characters in a volume label:

* ? / \ | . , ; : + = < > []

If you do not specify a volume label, **label** prompts you with the following message:

```
Volume in drive X is xxxxxxxxxx
Volume label (11 characters, ENTER for none)?
```

If the disk does not already have a volume label, **label** prompts you with the message:

```
Volume in drive X has no label
Volume label (11 characters, ENTER for none)?
```

Type the volume label that you want and press the RETURN key. Or, you can press the RETURN key immediately if you want to delete the volume label. **Label** would then prompt you with the message:

```
Delete current volume label (Y/N)?_
```

If you press Y, **label** deletes the volume label on the disk. Otherwise, the volume label stays the same.

Note You can also use the MS-DOS **dir** command to determine if the disk already has a volume label.

Example:

Suppose you have a disk in drive A that contains sales information for 1986. To label this disk you might type the following:

```
label a:sales1986
```

Labeling a disk

Mkdir

Synonym:

md

Purpose:

Makes a new directory.

Syntax:

mkdir [*drive:*]*path*

Comments:

With this command you can create a multilevel directory structure. For instance, when you are in your root directory, you can create subdirectories. Remember, though, that when you create directories with **mkdir**, they always appear under your working directory unless you explicitly specify a different path with the **mkdir** command.

Examples:

The following command creates a subdirectory named *\user* in your root directory:

```
mkdir \user
```

Now, suppose you want to create a directory named *pete* under the *\user* directory. To do this you could simply type the following command:

```
mkdir \user\pete
```

Mkdir (Make Directory)



Creating a subdirectory

Mode

Mode**Purpose:**

Sets operation modes for devices.

Syntax:

Parallel printer mode:

mode LPT*n*[:][*chars*][,*lines*][,*P*]

Asynchronous communications mode:

mode COM*m*[:]*baud*[,*parity*[,*databits* [,*stopbits*][,*P*]]]

Redirecting parallel printer output:

mode LPT*n*[:] = COM*m*[:]

Display modes:

mode *display*

or

mode [*display*],*shift*[,*T*]

Comments:

Parallel printer mode

For parallel printer mode:

- n* Specifies the printer number: 1, 2, or 3.
- chars* Specifies characters per line: 80 or 132.
- lines* Specifies vertical spacing, lines per inch: 6 or 8.
- P* Specifies that **mode** tries continuously to send output to the printer if a time-out error occurs.
This option causes part of the **mode** program to remain resident in memory.

The default settings are LPT1, 80 characters per line, and 6 lines per inch.

You can break out of a time-out loop by pressing CONTROL-BREAK.

For asynchronous communications modes:

Asynchronous communications modes

- m* Specifies the asynchronous communications (COM) port number: 1 or 2.
- baud* Specifies the transmission rate: 110, 150, 300, 600, 1200, 2400, 4800, or 9600. You need to specify at least the first two digits of each number.
- parity* Specifies the parity: N (none), O (odd), or E (even).
- databits* Specifies the number of databits: 7 or 8.

stopbits Specifies the number of stop bits: 1 or 2.

P Specifies that **mode** is using the COM port for a serial printer and continuously retrying if time-out errors occur.

This option causes part of the **mode** program to remain resident in memory.

The default settings are COM1, even parity, and 7 databits.

If *baud* is 110, then the default number of stop bits is 2; otherwise, the default is 1 stop bit.

For redirecting parallel printer output (to an asynchronous communications port):

n Specifies the parallel printer port number: 1, 2, or 3.

m Specifies the asynchronous communications port number: 1 or 2.

Redirection causes part of the **mode** program to remain resident in memory.

Note You must use the **mode** command to specify the asynchronous communications mode before you can redirect parallel printer output to it.

For setting display modes:

display Specifies one of the following values: 40, 80, BW40, BW80, CO40, CO80, or MONO.

40 indicates 40 characters per line.

80 indicates 80 characters per line.

BW and CO refer to a color graphics monitor adapter with color disabled (BW) or enabled (CO).

MONO specifies a monochrome display adapter with a constant display width of 80 characters per line.

shift Specifies the direction that you want to shift the display: R (right) or L (left).

This option causes part of the **mode** program to remain resident in memory.

T Specifies a test pattern for aligning the display. If you specify *T*, **mode** asks if the screen is aligned properly. If you type *N*, **mode** repeats the shift and asks if the screen is aligned properly. The command ends when you type *Y*.

Redirecting printer output

Display modes

Examples:

If you want your computer to send its printer output to a serial printer, you need to use the **mode** command twice. The first **mode** command specifies the asynchronous communications modes, and the second **mode** command redirects the computer's parallel printer output to the asynchronous communications port specified in the first **mode** command.

For example, if your serial printer operates at 4800 baud with even parity, and if it is connected to the COM1 port (the first serial connection on your computer), you would type the following:

```
mode com1:48,e,,p
```

```
mode lpt1:=com1:
```

If you redirect parallel printer output from LPT1 to COM1, and then decide that you want to print a file using LPT1, you can simply type the following command:

```
mode lpt1:
```

This disables any redirection of LPT1.

Suppose you want your computer to print on a parallel printer that is connected to your computer's second parallel printer port (LPT2). If you want to print with 80 characters per line and 8 characters per inch, you would type one of the following commands:

```
mode lpt2: 80,8
```

or:

```
mode lpt2: ,8
```

If you want your computer to keep trying to print a file until your printer is ready to print it, type this next command:

```
mode lpt2:80,8,P
```

To stop retrying to print, you can press CONTROL-BREAK or type the **mode** command without the **P** option.

Note If you print files every time you start MS-DOS, you may want to include **mode** commands in your *autoexec.bat* file. See Chapter 4, "Batch Processing," for more information on the *autoexec.bat* file.

More

Purpose:

Sends output to the console one screen at a time.

Syntax:

more

Comments:

More is a filter that reads from standard input (for instance, a command from your terminal) and displays one screen of information at a time. The **more** command then pauses and displays the "--More--" message at the bottom of your screen. To continue displaying information, press the RETURN key and keep pressing it until you have read all the data.

Note To hold input information until it is displayed, the **more** command creates a temporary file on the disk. If the disk is full or write-protected, however, the **more** command will not work.

Examples:

More is useful for viewing long files. For example, if you have a long file of customers, you could use the **more** command to view it one screen at a time. Suppose this file is called *clients.new*. To see it you would just type the following command:

```
type clients.new | more
```

You can also redirect input from a file to **more**, for example:

```
more < clients.new
```

This command also sends the file *clients.new* to the screen one screenful at a time.

More



Viewing long files

Redirecting input from a file

Path**I**

Path**Purpose:**

Sets a command search path.

Syntax:

path [*drive:*][*path*][:*drive:*][*path*]...

or

path ;

Comments:

The **path** command lets you tell MS-DOS which directories to search for external commands — after it searches your working directory. The default value is no path.

For instance, to tell MS-DOS to search the *\user\pete* directory for external commands, you would simply type the **path** command followed by the directory name *\user\pete*. Then, until you exit MS-DOS or set another path, MS-DOS searches the *\user\pete* directory for external commands.

You can tell MS-DOS to search more than one path by specifying several paths separated by semicolons. If you use the **path** command without options, it prints the current path. And if you use the following command, MS-DOS sets the NUL path:

path ;

This command means that MS-DOS searches only the working directory for external commands.

Example:

The following command tells MS-DOS to search three directories to find external commands (the three paths for these directories are *\user\pete*, *b:\user\emily*, and *\bin*):

path \user\pete;b:\user\emily;\bin

MS-DOS searches the paths in the order specified in the **path** command.

Setting a search path

Print

Purpose:

Prints a text file on a lineprinter *while you are processing other MS-DOS commands* (usually called *background printing*).

Syntax:

print [*drive:*][*pathname*][*/d:device*] [*/b:size*][*/q:value*] [*/t*][*/c*][*/p*]

Comments:

You can use the **print** command only if you have a lineprinter attached to your computer. If you do have a lineprinter you can use the following switches with this command:

/d:device Specifies the print device name. If you do not give this switch, **print** prompts for a print device. If you do not specify one, the default device is PRN (the first parallel printer connected to your computer).
Other possible print device names are AUX, LPT1, LPT2, LPT3, COM1, and COM2. These represent the first, second, and third parallel printers, and the first and second serial printers connected to your computer.

The following switches are allowed only the first time you run the **print** command after starting MS-DOS:

/b:size Sets the size in bytes of the internal buffer. To speed up the **print** command, you increase the value of **/b**.
/q:value Specifies the number of files allowed in the print queue — if you want more than 10. The minimum value for the **/q** switch is 4, the maximum 32, and the default, 10.
/t Deletes all files in the print queue (those files waiting to be printed).
/c Turns on cancel mode and removes the preceding filename and all following filenames from the print queue.
/p Turns on print mode and adds the preceding filename and all following filenames to the print queue.

Print



The Print switches

The **print** command, when used without options, displays the contents of the print queue on your screen without affecting the queue.

Note Each print queue entry may contain a maximum of 64 characters, including the drive name. So, if you want to print files in deep subdirectories (many levels down), you may need to change directories.

Examples:

The following command empties the print queue for the device named LPT1:

```
print /t /d:LPT1
```

The following command removes the *pencil.tst* file from the default print queue:

```
print a:pencil.tst /c
```

The next two commands show how to remove the file *pencil.tst* from the queue and then add the file *pen.tst* to the queue:

```
print pencil.tst /c  
print pen.tst /p
```

Prompt

Purpose:

Changes the MS-DOS command prompt.

Syntax:

prompt [[*text*][*\$character*]...]

Comments:

This command lets you change the MS-DOS system prompt (for example, A>). If when using the **prompt** command you do not type a new value, the prompt is set to the default value, which includes the default drive designation.

You can use the characters in the prompt command to create special prompts. You must precede each character with a dollar sign (\$):

Specify this character

To get this prompt

\$	The \$ character
t	The current time
d	The current date
p	The working directory of the default drive
v	The version number
n	The default drive
g	The > character
l	The < character
b	The character
—	RETURN-LINEFEED
s	A space (leading only)
e	ASCII code X'1B' (escape)

Examples:

The following example sets the drive prompt to *drive:current directory*:

```
prompt $p
```

Prompt



Setting your prompt

The following command sets a two-line prompt that displays the following:

```
Time = (current time)
Date = (current date)
```

```
prompt time = %t%_date = %d
```

If your terminal has an ANSI escape sequence driver, you can use escape sequences in your prompts. The following command, for example, sets your prompt in inverse video mode and returns to video mode for other text:

```
prompt  %e[7m%n:%e[m
```

Recover

Purpose:

Recovers a file or disk containing bad sectors.

Syntax:

recover [*drive:*]

or

recover *drive:[pathname]*

Comments:

If the **chkdsk** command shows that a sector on your disk is bad, you can use the **recover** command to recover the entire disk or just the file containing the bad sector.

This action causes MS-DOS to read the file sector by sector and to skip the bad sectors. When MS-DOS finds a bad sector, it marks the sector so that it no longer allocates your data to that sector.

Examples:

To recover a disk in drive A you would use the following command:

```
recover a:
```

Suppose you have a file named *pencil.ad* that has a few bad sectors. To recover this file you would use the following command:

```
recover pencil.ad
```

Recover



Recovering a disk

Recovering a file

Ren (Rename)**I**

Ren**Synonym:****rename****Purpose:**

Changes the name of a file.

Syntax:**ren** [*drive:*]*pathname pathname***Comments:**

The **ren** command renames all files matching the first *pathname*. However, because you cannot rename files across disk drives, the **ren** command ignores any drive name that you specify with the second *pathname*.

You may use wildcards (* or ?) in either *pathname* option, but if you use them in the second *pathname*, **ren** will not change the positions of the corresponding character.

Examples:

The following command changes the extension of all filenames ending in *.txt* to *.doc*:

```
ren *.txt *.doc
```

In the next example, **ren** renames a file named *chap10* (on drive B) to *part10*:

```
ren b:chap10 part10
```

The newly renamed file *part10* remains on drive B.

Renaming files

Replace

Purpose:

Updates previous versions of files.

Syntax:

replace [*drive:*]*pathname* [*drive:*]*path* [/a][/d][/p][/r][/s][/w]

Comments:

The **replace** command lets you easily update files on your hard disk with new versions of software.

Replace performs two functions:

- By default, it replaces files in the target directory with files in the source directory that have the same name. You may use wildcards in source filenames.
- When you specify the /a switch, **replace** adds files that exist in the source directory (but *not* in the target directory) to the target directory.

The /a switch adds new files to the target directory instead of replacing existing ones. You may not use this switch with either the /d or /s switch.

The /d switch replaces files in the target directory only if the source files are newer than the corresponding target files. This switch is incompatible with the /a switch.

The /p switch prompts you with the following message before it replaces a target file or adds a source file:

Replace *filename*? (Y/N) _

The /r switch replaces read-only files as well as unprotected files. If you do not specify this switch, any attempt to replace a read-only file causes an error and stops the **replace** process.

The /s switch searches all subdirectories of the target directory while it replaces matching files. This switch is incompatible with the /a switch. **Replace** never searches subdirectories in the source path.

The /w switch waits for you to hit any key before it replaces any files. If you do not specify this switch, **replace** begins replacing or adding files immediately.

Replace



The Replace switches

As files are replaced or added, **replace** displays the filenames on the screen; then at the conclusion of the **replace** operation, it displays a summary line:

```
NNN file(s) added/replaced
```

or:

```
No files added/replaced
```

Note You cannot use the **replace** command to update hidden files or system files.

Examples:

Replacing files

Suppose your hard disk, drive C, contains several old files named *phones.cli* that contain client names and phone numbers. To update these files and replace them with the latest version of the *phones.cli* file on the disk in drive A, you would type the following command:

```
replace a:\phones.cli c:\ /s
```

This command replaces every file on drive C that is named *phones.cli* with the file *phones.cli* from the root directory on drive A.

Adding files

Suppose you want to add some new printer device drivers to a directory called *c:\mstools*, which already contains several printer driver files for a word processor. To do this, you would type the following:

```
replace a:*.prd c:\mstools /a
```

This command searches the default directory of drive A for any files that have the extension *.prd* (that don't currently exist in the *\mstools* directory on drive C) and then adds these files to *c:\mstools*.

Upon completion, **replace** returns one of the following exit codes:

0	Command successful
1	Command line error
2	File not found
3	Path not found
5	Access denied
8	Insufficient memory
15	Invalid drive
Other	Standard MS-DOS error

Replace exit codes

You can test for these codes by using the *errorlevel* condition of the batch processing **if** command.

Restore



Restore

Purpose:

Restores files that were backed up using the Microsoft or the IBM backup program.

Syntax:

restore *drive:* [*drive:*][*pathname*] [/s] [/p] [/a:*date*] [/b:*date*] [/e:*time*] [/L:*time*] [/m] [/n]

Comments:

The **restore** command can restore files from disks of different media types. For example, you can restore files from:

- Hard disk to floppy disk
- Floppy disk to floppy disk
- Floppy disk to hard disk
- Hard disk to hard disk

The first *drive:* option is the drive name for the disk containing the backed up files. The second *drive:* and the *pathname* option are the drive and pathname of the files you want to restore.

This **restore** program and the one supplied by IBM are compatible except for switches /a, /b, /e, /L, /m, and /n, described in the table below.

The Restore switches

You can use the following switches with the MS-DOS **restore** command:

/s	Restores subdirectories also.
/p	Prompts for permission to restore any hidden or read-only files that match the file specification.
/a: <i>date</i>	Restores only those files that were last modified on or after the given date.
/b: <i>date</i>	Restores only those files that were last modified on or before the given date.
/e: <i>time</i>	Restores only those files that were last modified at or earlier than the given time.
/L: <i>time</i>	Restores only those files that were last modified at or later than the given time.
/m	Restores only those files that have been modified since the last backup.
/n	Restores only those files that no longer exist on the target disk.

The **restore** program sets the exit codes in the following manner:

- 0 Normal completion
- 1 No files were found to restore
- 3 Terminated by user
- 4 Terminated due to error

Refer to the **backup** command in this chapter for more information.

Example:

To restore the file *invest.mnt* from the backup disk in drive A to the *\irsharpe* directory on drive C, type the following:

```
restore a: c:\irsharpe\invest.mnt
```

Press the RETURN key to let MS-DOS know that the backup disk is in drive A. Then once MS-DOS has restored the file, use the **dir** or **type** command to make sure that the file was restored properly.

Restore exit codes

Restoring a file

Rmdir (Remove Directory)

Rmdir**Synonym:****rd****Purpose:**

Removes a directory from a multilevel directory structure.

Syntax:

rmdir [*drive:*]*path*

Comments:

This command removes a directory that is empty except for the "." and ".." shorthand symbols. Before you can remove a directory entirely, you must delete its files and subdirectories.

Example:

Suppose you want to remove a directory named `\user\pete`. You first issue a **dir** command for the `\user\pete` path to ensure that the directory is empty; you then type the following command:

```
rmdir \user\pete
```

Removing a directory

Set

Purpose:

Sets one string of characters in the environment equal to another string for later use in programs.

Syntax:

set [*string*=[*string*]]

Comments:

You should use the **set** command only if you want to set values for programs you have written.

When MS-DOS sees a **set** command, it inserts the given *string* and its equivalent into a part of memory reserved for the *environment*. If the *string* already exists in the environment, it is replaced with the new setting.

If you specify just the first *string*, **set** removes any previous setting of that *string* from the environment. Or if you use the **set** command without options, MS-DOS displays the current environment settings.

When batch processing, you can also use the **set** command to define your replaceable parameters by name instead of by number. For example, if your batch file contains the statement "type %file%", you could use the **set** command to set the name that MS-DOS will use for that variable. In the following command, for example, **set** replaces the %*file*% parameter with the filename *taxes.86*:

```
set file=taxes.86
```

To change the replaceable parameter names, you don't need to edit each batch file. Note also that when you use text (instead of a number) as a replaceable parameter, the name must be ended by a percent sign.

The **set** command is especially useful in the *autoexec.bat* file, because it lets you automatically set strings or parameters when you start MS-DOS. See Chapter 4, "Batch Processing," for more information about the *autoexec.bat* file.

Set



Defining replaceable parameters

Setting a string

Example:

The following command sets the string "include" to *c:\inc* until you change it with another **set** command:

```
set include=c:\inc
```

If you just type the **set** command by itself, MS-DOS displays the current environment settings.

Share

Purpose:

Installs file sharing and locking.

Syntax:

share [/f:space][/L:locks]

Comments:

You can see the **share** command only when networking is active. If you want to install shared files, you can include the **share** command in your *autoexec.bat* file. To learn more about shared files, see the *Microsoft Networks Manager's Guide*.

MS-DOS has a storage area that it uses to record file sharing information; to allocate file space (in bytes) for this area, you use the /f:space switch.

Each open file requires enough space for the length of the full filename plus 11 bytes, since the average pathname is around 20 bytes in length. The default value for the /f switch is 2048.

The /L:locks switch allocates the number of locks you want to allow. The default value for the /L switch is 20.

Once you have used the **share** command in an MS-DOS session, all read and write requests are checked by MS-DOS.

Example:

The following example loads file sharing and uses the default values for the /f and /L switches:

```
share
```

Share



Loading the share command

Sort

Sort

Purpose:

Reads input, sorts the data, then writes the sorted data to your terminal screen, to a file, or to another device.

Syntax:

sort [*drive:*][*pathname*][/*r*][/*+n*]

Comments:

The **sort** command lets you alphabetize a file according to the character in a certain column. You specify the file by the *drive:* and *pathname* options. The two other **sort** options, the */r* and */+n* switches, are described as follows:

The Sort switches

<i>/r</i>	Reverses the sort, that is, sorts from Z to A.
<i>/+n</i>	Sorts the file according to the character in column <i>n</i> , where <i>n</i> is some number. If you do not specify this switch, the sort command sorts the file according to the character in the first column.

Note **Sort** does not distinguish between uppercase and lowercase letters.

Examples:

Sorting a file

The following command reads the file *expenses.txt*, sorts it in reverse order, and writes the output to a file named *budget.txt*:

```
sort /r expenses.txt budget.txt
```

The following command pipes the output of the **dir** command to the **sort** filter. This filter sorts the directory listing starting with column 14 (the column in the directory listing that contains the file size) and sends the output to the screen. The result is a directory, sorted by file size:

```
dir | sort /+14
```

The following command does the same thing as the previous one, except that the **more** filter gives you a chance to read the sorted directory one screenful at a time:

```
dir | sort /+14 | more
```

Subst

Purpose:

Substitutes a path with a drive letter.

Syntax:

subst [*drive:*][*path*[/d]

Comments:

The **subst** command lets you associate a *path* with a drive letter. This drive letter then represents a *virtual drive* because you can use the drive letter in commands as if it represented an actual drive.

When MS-DOS finds a command that uses a virtual drive, it replaces the drive letter with the path.

If you type the **subst** command by itself, MS-DOS displays the names of the virtual drives in effect.

To delete a virtual drive you use the /d switch.

Example:

The following command creates a virtual drive, Z, for the pathname *b:\user\betty\forms*:

```
subst z:b:\user\betty\forms
```

Now, instead of typing the full pathname, you can get to this directory by simply typing the name of the virtual drive:

z :

Subst



Creating a virtual drive

Sys (System)

Sys**Purpose:**

Transfers the MS-DOS system files from the disk in the default drive to the disk in the specified drive.

Syntax:

sys *drive*:

Comments:

Usually, you use the **sys** command to update your system or place it on a formatted disk that contains no files. You must type a drive letter with this command.

If the system files *io.sys* and *msdos.sys* are already on the target disk, they must take up the same amount of space on the disk as the new system will need. This means that you cannot transfer system files from an MS-DOS 2.0 disk to an MS-DOS 1.1 disk; instead, before the **sys** command will work, you must reformat the MS-DOS 1.1 disk with the MS-DOS 2.0 **format** command.

The target disk must be completely blank or must already contain the system files *io.sys* and *msdos.sys*.

The transferred files are copied in the following order:

IO.SYS
MSDOS.SYS

Io.sys and *msdos.sys* are both hidden files that *do not appear* when you type the **dir** command.

Sys does not transfer the *command.com* file (the command processor). So to transfer *command.com* to the target disk, you must use the **copy** command.

Updating your system

Time

Purpose:

Displays and sets the time.

Syntax:

time [*hours:minutes*]

Comments:

Time is entered in a 24-hour clock format. If you just type the **time** command by itself, the following message is displayed:

```
Current time is hh:mm:ss.cc
Enter new time:_
```

If you don't want to change the time shown, you simply press the RETURN key. And if you want to change the time after you have started MS-DOS (for example, to 8:20 a.m.), type the **time** command followed by 8:20 in response to the MS-DOS prompt. Note that letters are not allowed; instead, you must type the time using numbers only. The allowed options are:

```
hours = 0-24
minutes = 0-59
```

Separate the hour and minute entries by a colon. You do not have to type the *ss* (seconds) or *cc* (hundredths of a second).

Example:

If you do not type a valid time, MS-DOS displays the following message and then waits for you to type a valid time:

```
Invalid time
Enter new time:_
```

As with the **date** command, you can change the **time** command format by changing the **country** command in the *config.sys* file. See Appendix B, "How to Configure Your System," for more information.

Time



Setting a new time

Tree

Tree**Purpose:**

Displays the path (and, optionally, lists the contents) of each directory and subdirectory on the given drive.

Syntax:

tree [*drive:*] [/f]

Comments:

The **tree** command lists the full path of each directory and subdirectory on the specified *drive*.

The *drive:* option specifies the drive that you want to use. If you do not specify this option, **tree** uses the default drive.

The /f switch displays the names of the files in each directory.

Example:

If you want to print a list of the directory and filenames on the disk in drive B, you can use the following command:

```
tree b: /f > prn
```

Type

Type



Purpose:

Displays the contents of a text file on the screen.

Syntax:

type [*drive:*]*filename*

Comments:

To view a text file without modifying it, you can use the **type** command. (Use **dir** to find the name of a file, and **edlin** to change the contents of a file.)

Note that when you use **type** to display a file that contains tabs, all the tabs are expanded to 8 spaces wide. Also, if you try to display a binary file, you may see strange characters on the screen, including bells, formfeeds, and escape sequences.

Example:

If you want to display the contents of a file called *holiday.mar*, you would type the following command:

```
type holiday.mar
```

Displaying a file

Ver



Ver

Purpose:

Prints the MS-DOS version number.

Syntax:

ver

Comment:

If you want to know what version of MS-DOS you are using, you simply type the **ver** command. The version number will then be displayed on your screen.

Example:

When you type the **ver** command, the following message is displayed:

MS-DOS Version 3.20

Displaying the MS-DOS version

Verify

Purpose:

Turns the verify switch on or off when writing to a disk.

Syntax:

verify [ON]

or

verify [OFF]

Comments:

You can use this command to verify that your files are written correctly to the disk (no bad sectors, for example). MS-DOS performs a **verify** each time you write data to a disk. You will receive an error message only if MS-DOS is unable to successfully write your data to a disk.

Examples:

If you want to know the current setting of **verify**, use the **verify** command without an option:

```
verify
```

Verify ON remains in effect until a program changes it (by a **Set Verify** system call), or until you type the following:

```
verify OFF
```

This command has the same purpose as the /v switch in the **copy** command.

Verify

I

Vol (Volume)

Displaying a volume label

Vol**Purpose:**

Displays the disk volume label or volume ID, if it exists.

Syntax:

vol [*drive:*]

Comments:

This command displays the volume label of the disk in the specified *drive*. If you do not type a drive letter, MS-DOS displays the volume label of the disk in the default drive.

Example:

If you want to see the volume label for a disk in drive A, you could type the following:

```
vol a:
```

If the volume label is "DOS 3-2" MS-DOS responds by displaying the message:

```
Volume in drive A is DOS 3-2
```

Xcopy

Xcopy



Purpose:

Copies files and directories, including lower level directories, if they exist.

Syntax:

xcopy [*drive:*][*path*]*filename*[*drive:*] [*path*][*filename*] [/a]
[/*d:date*] [/e] [/m] [/p] [/s] [/v] [/w]

Comments:

The first *drive:*, *path*, and *filename* parameters specify the source file or directory that you want to copy. The second *drive:*, *path*, and *filename* parameters specify the target. You must include at least one of the source parameters. If you omit the target parameters, **xcopy** assumes you want to copy the files to the default directory.

If you do not specify the *path* option, **xcopy** uses the default directory with the default filename, *.*.

The /a switch copies source files that have their archive bit set. The switch does not modify the archive bit of the source file. See the **attrib** command for information on how to set the archive attribute.

The /d switch copies source files that you modified on or after the date specified by *date*. Note that the date format may vary depending on the country code that you are using. See the **date** command for more information.

The /e switch copies any subdirectories, even if they are empty. You must use this switch with the /s switch.

The /m switch is similar to the /a switch in that it copies archived files only; however, it turns off the archive bit in the source file. See the **attrib** command for information on how to set the archive attribute.

The /p switch prompts you with "(Y/N?)" to let you confirm whether you want to create each target file.

The /s switch copies directories and lower level subdirectories, unless they are empty. If you omit this switch, **xcopy** works within a single directory.

The /v switch causes **xcopy** to verify each file as it is written to the target to make sure that the target files are identical to the source files.

The Xcopy switches

The **/w** switch causes **xcopy** to wait before it starts copying files. **Xcopy** displays the following message:

Press any key when ready to start copying files

You must press a key to continue, or press CONTROL-C to abort the **xcopy** command.

Xcopy exit codes

When correctly written programs exit back to MS-DOS, they return an exit code: 0 if no error occurred, or a value greater than zero if there was a problem. This exit code, which you can test in batch files, lets you "branch" to an error-handling routine in the batch file.

If **xcopy** encounters an error, it returns one of the following exit codes:

- 0 Copy without error
- 1 No files found to copy
- 2 CONTROL-C entered by user to terminate **xcopy**
- 4 Initialization error
There is not enough memory — invalid drive or command line syntax, file not found, or path not found.
- 5 Int 24 error occurred
The user aborted from INT24 error reading or writing disk.

You can test for these codes by using the *errorlevel* condition of the batch processing **if** command.

Examples:

Copying to a disk with a different format

Because the **diskcopy** command copies disks track by track, it requires your source and target disks to have the same format. If you have a disk that contains files in subdirectories and you want to copy it to a target disk that has a different format, you must use the **xcopy** command. For example, the following example copies all the files and subdirectories (including any empty subdirectories) on the disk in drive A to the disk in drive B:

```
xcopy a: b: /s /e
```

The **xcopy** command may prompt you to specify whether the target is a file or a directory. If you don't want to receive this prompt, type the following command:

```
copy /b xcopy.exe mcopy.exe
```

This command creates a new command called **mcopy.exe**. Now you can use the **mcopy** command the same way you use the **xcopy** command, except that **mcopy** automatically determines whether the target is a file or a directory.

Mcopy uses the following rules for copying files:

- If the source is a directory, the target is a directory.
- If the source includes multiple files, the target is a directory.
- If you append a backslash (\) to the end of the target name, the target is a directory. For example, the following command creates the directory *a:\workers*, if it doesn't already exist, and copies the file *payroll* to it:

```
xcopy payroll a:\workers\
```



4 Batch Processing

In this chapter you will learn:

- How to create a batch file
- How an *autoexec.bat* file works
- How to use replaceable parameters in a batch file
- How to run a batch file
- How to do multitasking with batch files

Note If you are not writing batch programs you do not need to read this chapter.

Why Use Batch Files?

You may often find yourself repeatedly typing the same sequence of commands to perform some common task. With MS-DOS you can put this command sequence into a special file called a *batch file*, and then run the whole sequence of commands by simply typing the name of the batch file. Note that you don't need to type the batch file's extension, even though all your batch files must include the *.bat* extension in their filenames.

MS-DOS performs these "batches" of your commands just as if you had typed them from the keyboard. This is called *batch processing*. By using a batch file, you only have to remember to type one command, instead of several. In effect, you use batch files to create personalized commands.

How to Create Batch Files

You can create a batch file by using **edlin**, the MS-DOS line editor, or by using the **copy** command. If you want to create files with **edlin**, you should refer to Chapters 6 and 7 for more information. The examples in this chapter show you how to use the **copy** command to create batch files.

Creating a batch file

Suppose, for example, that you want to create a batch file to format and check a new disk. To do this you simply follow these steps:

- ❶ First, type the following:

```
copy con checknew.bat
```

Press RETURN. This command tells MS-DOS to copy the information from the console (keyboard) to the file *checknew.bat*.

- ❷ Next, type the following lines, pressing RETURN after each:

```
rem This is a file to format and
rem check new disks.
rem It is named CHECKNEW.BAT.
pause Insert new disk in drive B:
format b: /v
chkdsk b:
```

- ❸ After the last line, press CONTROL-Z and then press RETURN to save the batch file. MS-DOS displays the message "1 File(s) copied" to show that it created the file.

- ❹ Now, to execute the file, simply type the following command:

```
checknew
```

The result is the same as if the lines in the *.bat* file were entered from the keyboard as individual commands.

About Batch Processing

Here are a few things you should know before you run a batch process with MS-DOS:

- You must name each batch file with an extension of *.bat*.
- To execute a batch file, you type only its filename and not the extension.
- If you press CONTROL-C while the batch file is running, MS-DOS asks you to confirm that you want to terminate the batch process.
- If you remove the disk that contains a batch file being run, MS-DOS prompts you to reinsert the disk so that it can continue processing the file.
- You can specify the name of another batch file as the last command in a batch file. This feature allows you to call one batch file from another when the first has finished.

Running a batch file

- You can use any of the redirection symbols (< > << >>) in a batch file. See Chapter 2, "About Commands," for more information on using these symbols.
- You cannot use the pipe symbol (|) in a batch file.
- Setting the directory or drive affects every subsequent command in the batch file.
- Setting environment strings also affects every subsequent command in the batch file.

Note If you have more than one external command with the same name, MS-DOS will run only one of them, according to the following order of precedence: *.com*, *.exe*, *.bat*.

Suppose, for example, that your disk includes the files *format.exe* and *format.bat*. If you were to type the external command **format**, MS-DOS would always run the program *format.exe* first. In order to run the batch file *format.bat*, you would have to place it in a separate directory and give a path along with the external command.

What is an Autoexec.bat File?

The autoexec.bat file

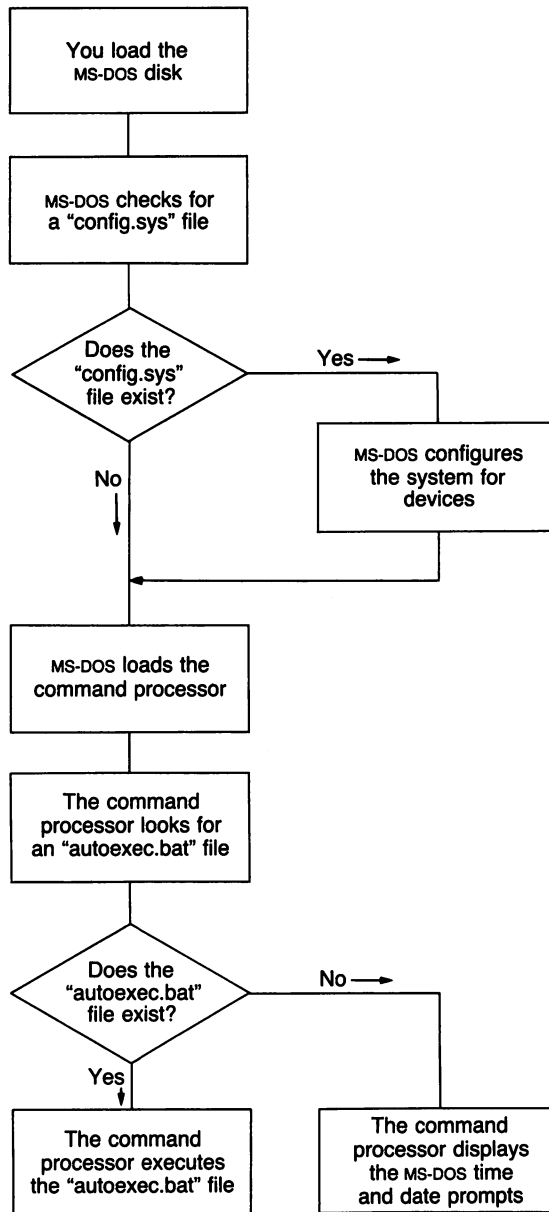
An *autoexec.bat* file lets you run programs automatically when you start MS-DOS. This can be useful when you want to run a specific application under MS-DOS, and when you want MS-DOS to execute a batch program each time you start your computer. By using an *autoexec.bat* file you can avoid loading two separate disks just to perform these tasks.

When you start your computer, MS-DOS searches the root directory of the default disk drive for a file named *autoexec.bat*. If it finds the *autoexec.bat* file, MS-DOS immediately processes it, bypassing the date and time prompts. If MS-DOS does not find an *autoexec.bat* file, then the date and time prompts appear automatically.

Hint MS-DOS does not prompt you for a current date and time unless you include the **date** and **time** commands in your *autoexec.bat* file.

It's a good idea to add these two commands to your *autoexec.bat* file, since MS-DOS uses this information to keep your directory current. See Chapter 3, "MS-DOS Commands," for more information on the **date** and **time** commands.

The following figure shows what happens when you start MS-DOS:



What Happens When You Start MS-DOS

How to Create an Autoexec.bat File

There are many things you can do with an *autoexec.bat* file to help you use MS-DOS more efficiently. For instance, you will probably want to set the **time** and **date**, your **path**, and any other options that you plan to use on a regular basis.

If, for example, you want to automatically load GW-BASIC and run a program called **menu** each time you start MS-DOS, you could create an *autoexec.bat* file as follows:

- 1 Type the following command and then press RETURN:

```
copy con autoexec.bat
```

This command tells MS-DOS to copy what you type from the keyboard into the *autoexec.bat* file. Note that you *must* put the *autoexec.bat* file in the root directory of your MS-DOS disk.

- 2 Now type the following lines:

```
date  
time  
path=c:\;c:\bin;a:\  
prompt [$p]  
cls  
gwbasic menu
```

- 3 After the last line, press CONTROL-Z and press RETURN to copy these lines into the *autoexec.bat* file.
- 4 The **menu** program will now run automatically whenever you start MS-DOS.

Once your *autoexec.bat* file is set up as above, it will perform the following actions when you start MS-DOS: it will ask you to enter the date and time; it will set your command search path; and it will set your prompt to display the default drive and directory.

Finally, the *autoexec.bat* file will clear the screen and tell MS-DOS to load GW-BASIC and run the **menu** program. To run your own GW-BASIC program, type its name in place of **menu** in the example. In addition to GW-BASIC programs, you can also put any MS-DOS command or series of commands in the *autoexec.bat* file.

Creating an autoexec.bat file

How to Create a Batch File with Replaceable Parameters

There may be times when you want to create a program and run it with different sets of data. These data may be stored in various MS-DOS files.

With MS-DOS you can create a batch (*.bat*) file with *replaceable* (dummy) *parameters*, where a *parameter* is a command option that you define. These parameters, named %0-%9, hold the places for the values that you supply when you give the batch command.

Replaceable parameters make batch files more flexible and easy to use. For example, you can create a batch file called *sorter.bat* that sorts a file containing a specific sequence of characters or strings. Each time you execute the *sorter* batch file, you tell MS-DOS which string you want, which file to search to find that string, and which temporary file to use for sorting. *Sorter* would then print the resulting list on the printer.

Using replaceable parameters

- 1 To create the *sorter.bat* file, type the following command and then press the RETURN key:

```
copy con sorter.bat
```

- 2 Now type the following lines:

```
type %2 | find "%1" > %3
type %3 | sort > prn
del %3
```

- 3 To save the batch file, press CONTROL-Z and then RETURN.
The batch file *sorter.bat* now consists of three command lines and is on the disk in the default drive.

When you execute the file, MS-DOS sequentially replaces %1, %2, and %3 with the parameters you supply. If you use the dummy parameter %0, MS-DOS always replaces it with the drive name (if specified) and the filename of the batch file (for example, *sorter*).

Notes

- You can specify up to ten replaceable parameters (%0-%9). If you want to specify more than ten, refer to the **shift** command later in this chapter.
- If you use the percent sign as part of a filename *within a batch file*, you must type it twice. For example, to specify the file *abc%.exe*, you must type it as *abc%%.exe* in the batch file.

How to Run a Batch File

To run the batch file *sorter.bat*, type the batch filename followed by the parameters that you want MS-DOS to substitute for %1, %2, and %3.

Suppose that on the disk in drive A you have a file that lists your customers' names and regions. The file might look something like this:

Shores, Sandy	north
Poster, Emily	south
Sharpe, Isabel R.	north
Fisher, Pete	east
Conrad, Betty	south
Rey, Fernando	north
Shaw, Rick	west
Moss, Chris	north

If you want to print an alphabetical list of the customers in the north, you can run the *sorter* batch file, with the appropriate parameters, by typing the following command and then pressing RETURN:

```
sorter north a:customer temp.fil
```

The output on the printer should look like this:

Moss, Chris	north
Rey, Fernando	north
Sharpe, Isabel R.	north
Shores, Sandy	north

The following table shows how MS-DOS replaces each of the parameters in the previous example:

Batch filename	(%0)	sorter
Parameter1	(%1)	north
Parameter2	(%2)	a:customer
Parameter3	(%3)	temp.fil

The result is the same as if you had typed each of the commands in *sorter* with its parameters, as follows:

```
type a:customer | find "north" > temp.fil
type temp.fil | sort > prn
del temp.fil
```

Using the batch file, however, saves typing time and is much easier to remember.

Running a batch file

How to Use Temporary Files

Using temporary files

When using batch files, you may often want to use a temporary file to hold your work. You could use the same name each time you wanted to use a temporary file.

However, if you are using more than one batch file that uses the same temporary file, you might lose the contents of this temporary file. To avoid this problem, you should use a replaceable parameter to specify the name of the temporary file. Then each time you run the batch file, you'll be able to substitute a unique filename and you won't have to worry about information from one batch file getting into another.

It's also a good idea to delete temporary files once you finish using them. Otherwise, these files would eventually take up all the space on your disk.

Using the batch processing commands

Batch Processing Commands

Now that you have seen some of the capabilities of batch files, in this section you'll find out how to add power and flexibility to your batch programs by using batch processing commands. The following table lists these batch commands and describes what they do:

Command	What it does
echo	Turns the batch file echo feature on or off, or displays the current setting.
for	Performs a command for a set of files.
goto	Processes commands starting with the line after the specified label.
if	Performs a command if a condition is met.
pause	Pauses during the processing of a batch file.
rem	Displays a comment in a batch file.
shift	Increases the number of replaceable parameters in a batch process.

Echo

Purpose:

Turns the batch echo feature on and off.

Syntax:

echo [ON]

or

echo [OFF]

or

echo [*message*]

Comments:

Normally, commands in a batch file are displayed (“echoed”) on the screen when they are received by MS-DOS. You can turn off this feature by using the OFF option with the **echo** command. Similarly, you can turn the echo feature back on by using the ON option with **echo**.

If you do not specify ON or OFF, **echo** displays the current setting.

The command, **echo message** (where *message* is a line of text), is only useful if **echo** is off and if you are using a batch file. If, in your batch file, you type the **echo** command followed by a message, you can print messages on your screen. You can also put several echo message commands in your batch file to display a message that is several lines in length.

Example:

The following is an example of a batch file message of more than one line:

```
echo off
echo This batch file
echo formats and checks
echo new disks.
```

Echo



**Creating messages
for a batch file**

For

I**For****Purpose:**

Performs a command for a set of files.

Syntax:

for %%c in set do command
(for batch processing)

for %c in set do command
(for interactive processing)

Comments:

To avoid confusion with the %0–%9 batch parameters, the variable *c* can be any character except 0,1,2,3,...,9.

set is (*item**)

This command sequentially sets the %%c variable to each member of *set*, and uses the variable to evaluate *command*. If a member of *set* is an expression involving a wildcard (* or ?), then the variable is set to each matching *item* from the disk. In this case, only one such item is in *set*, so the command ignores any item other than the first.

Examples:

The following example binds the variable %f to files ending with *.asm in the working directory.

```
for %%f in ( *.asm ) do masm %%f
```

It then executes a command of the following form:

```
masm filename
```

Filename could be any one of the following:

```
invoice.asm  
receipts.asm  
taxes.asm
```

The following example binds the variable %f to the files named *report*, *memo*, and *address*; it then deletes each of these files:

```
for %%f in (report memo address) do del %%f
```

You must use two percent signs (%%) so that one will remain after the batch parameter (%0-%9) processing is complete. If you had only %f, instead of %%f, then the batch parameter processor would see the %, look at f, decide that %f was an error (a bad parameter reference), and throw out the %f so that the **for** command would never see it.

If the **for** command is *not* in a batch file, you should use only one percent sign.

Goto

Goto**Purpose:**

Processes commands starting with the line after the specified label.

Syntax:

goto *label*

Comments:

Goto lets you take commands from the batch file beginning with the line after the *label*, where a *label* is defined as the characters following **goto**. This *label* may include spaces, but not other separators, such as semicolons or equal signs. If your batch file does not contain the *label*, the batch file terminates.

Note Any line in a batch file that starts with a colon (:) is ignored during batch processing.

Example:

The following example sends the program processor to the label named *end* only if no errors occur when you format the disk in drive A:

```
:begin
echo off
format a: /s
if errorlevel 0 goto end
echo An error occurred during formatting.
:end
echo End of batch file.
```

If**Purpose:**

Performs a command based on the result of a condition.

Syntax:

if [NOT] *errorlevel number command*

or

if [NOT] *string1 == string2 command*

or

if [NOT] **exist** *filename command*

Comments:

The **if** statement allows conditional execution of commands. When the condition is true, MS-DOS executes the *command*, otherwise it ignores the *command*.

The conditions are described as follows:

errorlevel number

True if, and only if, the previous program executed by **command.com** had an exit code of *number* or higher. (When a program finishes, it returns an exit code via MS-DOS.) You can use this condition to perform other tasks that are based on the previous program's exit code.

string1 == string2

True if, and only if, *string1* and *string2* are identical after parameter substitution. Strings may not contain separators, such as commas, semicolons, equal signs, or spaces.

exist filename

True if, and only if, *filename* exists.

If you specify the NOT parameter, MS-DOS executes the *command* when the condition is false.

Example:

The following example prints the message "can't find datafile" if the file *product.dat* does not exist on the disk:

```
if not exist product.dat echo can't find datafile
```

If**I**

Displaying a conditional message

Pause

Pause**Purpose:**

Suspends execution of a batch file.

Syntax:

pause [*comment*]

Comments:

When a batch file is running, you may need to change disks or perform some other action. The **pause** command suspends execution of the batch file until you press any key, unless you press the CONTROL-C key sequence.

When the command processor encounters **pause**, it prints the following message:

Strike a key when ready . . .

If you press CONTROL-C, MS-DOS displays the following message:

Terminate batch job (Y/N)?

If you type *Y* in response to this prompt, the batch file ends and control returns to the operating system. Therefore, you can use **pause** to divide a batch file into pieces that allow you to end the batch command file at any intermediate point.

The *comment* parameter is useful when you want to display a special message. Unless **echo** is off, **pause** displays this comment before the "Strike a key" message.

Note The pause and comment line of your batch file will not appear if **echo** is off.

Example:

Suppose you want a program to display a message that asks the user to change disks in one of the drives. To do this you might use the following command:

Pause Please put a new disk into drive A

If **echo** is on, this line will precede the "Strike a key" message when you run the batch file.

Using pause messages in a batch file

Rem

Purpose:

During execution of a batch file, **rem** displays remarks that are on the same line as the **rem** command in that batch file.

Syntax:

rem [*comment*]

Comments:

The *comment* parameter is a line of text that helps you identify and remember what your batch file does.

The only separators allowed in the *comment* are spaces, tabs, and commas.

In your batch file, you can use **rem** without a comment to add spacing for readability.

Example:

The following example shows a batch file that uses remarks for both explanation and spacing:

```
rem This file formats and checks new disks
rem It is named checknew.bat
rem
pause Insert new disk in drive B
format B: /v
chkdsk B:
```

Rem

I

Using remarks in a batch file

Shift**I**

Shift**Purpose:**

Lets you change the position of replaceable parameters in batch file processing.

Syntax:

shift

Comments:

You can use the **shift** command to change the positions of (replaceable) command line parameters.

Usually command files are limited to handling ten parameters, %0 through %9. But by using **shift**, you can access more than ten parameters. This means that if there are more than ten parameters given on a command line, those that appear after the tenth (%9) will be shifted one at a time into %9.

You can use the **shift** command even if you have less than ten parameters.

Warning There is no backward **shift** command. Once you have executed **shift**, you cannot recover the first parameter (%0) that existed before the shift.

Example:

The following 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.

```
rem mycopy.bat copies
rem any number of files
rem to a directory.
rem The command is
rem mycopy dir files
:one
if "%1" = " " goto two
set todir = %1
shift
copy %1 %todir%
goto one
:two
set todir=
echo All done
```

Shifting replaceable parameters

5 MS-DOS Editing and Function Keys

In this chapter you will learn about:

- The MS-DOS editing and function keys
- The editing template
- The MS-DOS control characters

Special MS-DOS Editing Keys

Many operating systems handle command input differently than MS-DOS does. One difference in particular that sets MS-DOS apart is its set of special editing keys. For instance, with MS-DOS you don't have to type the same sequences of keys repeatedly, because the most recently typed command line is automatically placed in a special storage area called a *template*.

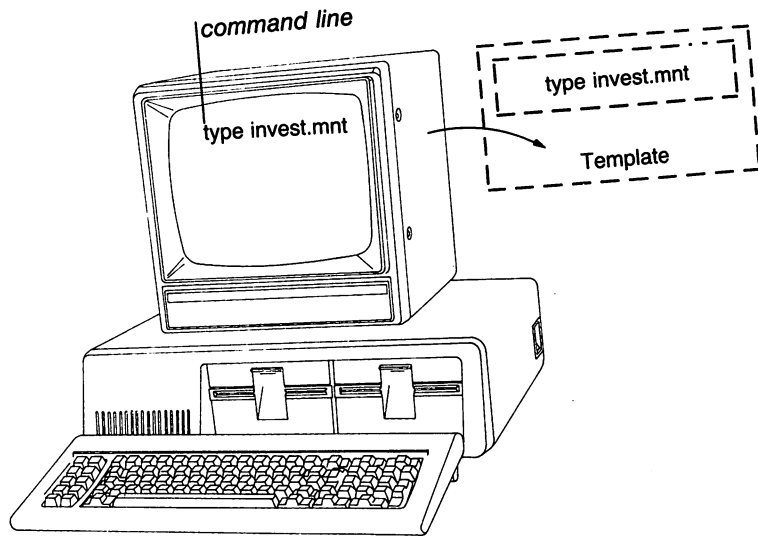
By using the template and the special editing keys, you can take advantage of the following MS-DOS features:

- You can repeat a command instantly by pressing two keys.
- If you make a mistake in a command line, you can edit and retry it without having to retype the entire line.
- With a minimum of typing, you can edit and execute a command line that is similar to a previous one.

How MS-DOS Uses the Template

When you type a command and press the RETURN key, MS-DOS automatically sends it to the command processor (*command.com*) for execution. At the same time, MS-DOS also sends a copy of this command to the template. You can then recall or modify the command by using the MS-DOS special editing keys.

The following figure shows how the template relates to the command line:



Special Editing Functions

Key	Editing function
F1	Copies one character from the template to the command line.
F2	Copies characters up to the character specified in the template and puts these characters on the command line.
F3	Copies all remaining characters in the template to the command line.
DEL	Skips over (does not copy) a character in the template.
F4	Skips over (does not copy) the characters in the template up to the character specified.
ESC	Voids the current input and leaves the template unchanged.
INS	Enters/exits insert mode.
F5	Makes the new line the new template.
F6	Puts a CONTROL-Z (1AH) end-of-file character in the new template.

The MS-DOS editing keys

Examples:

Using the template

Suppose you want to see the directory information for a file named *invest.mnt*. To get this information you could type the following command:

```
dir invest.mnt
```

This command line (*dir invest.mnt*) is also saved in the template. If you want to repeat the command, just press two keys: F3 and RETURN.

MS-DOS displays the repeated command on the screen when you press F3 as shown below:

```
dir invest.mnt
```

Notice that when you press the F3 key, MS-DOS copies the contents of the template to the command line; pressing the RETURN key then sends the command line to the command processor for execution.

If you want to display information about a file named *invest.rpt*, you can use the contents of the template. Pressing F2 followed by the letter *m* copies all characters from the template to the command line, up to but not including the *m*. MS-DOS displays:

```
dir invest._
```

Note that the underline is your cursor. Now type the letters *rpt* to get the following result:

```
dir invest.rpt_
```

Changing the template

The command line (*dir invest.rpt*) is now in the template and ready to be sent to the command processor for execution. To run the command, press the RETURN key.

Now, assume that you want to run the following command:

```
type invest.rpt
```

To do this, type the word *type* and then press the following sequence of keys: INS, SPACEBAR, F3, RETURN.

As you type, the characters appear directly on the command line, overwriting their corresponding characters in the template. Before you press the INS key, the word *type* replaces the word *dir* (and the space following it) in the template. After you press the INS key, this automatic replacement feature is turned off.

To insert a space between the word *type* and the filename *invest.rpt*, you pressed INS and then the SPACEBAR. Finally, to copy the rest of the template to the command line, you pressed F3 and then the RETURN key.

The command line *type invest.rpt* has been processed by MS-DOS, and the template now looks like this:

```
type invest.rpt.
```

If you had misspelled *type* as *pyte*, for example, a command error would have occurred. Still, instead of throwing away the whole command, you could save the misspelled line before pressing the RETURN key. You can do this by pressing the F5 key *before* the RETURN key, creating a new template:

```
pyte invest.rpt
```

You can then edit this error by pressing the following two keys: F1 and F3. The F1 key copies a single character from the template to the command line, resulting in the command that you want:

```
type invest.rpt
```

As an alternative, you can use the template that contains:

```
pyte invest.rpt
```

Then you can use the DEL and INS keys to get the same result, as follows:

```
DEL DEL F1 INS yp F3
```

Correcting errors in the template

To illustrate how the keys you type affect the command line, compare the key pressed with its result (shown beneath it along with a description of the effect).

DEL

-

Skips over 1st template character

DEL

-

Skips over 2nd template character

F1

t _

Copies 3rd template character

INS yp

typ _

Inserts two characters, *y* and *p*

F3

type invest.rpt _

Copies rest of template

Notice that DEL does not affect the command line. Instead, it affects the template by deleting the first character. Similarly, F4 deletes characters in the template, up to, but not including, a given character.

These special editing keys do give you more power and flexibility when you are typing. But, in addition to these keys, MS-DOS also has control characters that help you control the output from a command, or control the contents of the current command line. The next section describes how to use the MS-DOS control characters.

Using the MS-DOS control characters

Control Characters

A control character affects the command line in a special way. For example, you use CONTROL-C to stop running the current command, and you use CONTROL-S to suspend the screen output from a command.

Note When you type a control sequence, such as CONTROL-C, you must hold down the CONTROL key and then press the C key.

The following table shows the MS-DOS control characters and describes what they do.

Control character	What it does
CONTROL-C	Aborts the current command.
CONTROL-H	Removes the last character from a command line, and erases that character from the terminal screen.
CONTROL-J	Inserts a physical end-of-line, but does not empty the command line. Use the LINEFEED key to extend the current logical line beyond the physical limits of the terminal screen.
CONTROL-N	Causes echoing of output to a lineprinter.
CONTROL-P	Causes terminal output to a lineprinter.
CONTROL-S	Suspends output display on the screen. Press CONTROL-S again to resume.
CONTROL-X	Cancels the current line, empties the command line, and then outputs a backslash (\), RETURN, and LINEFEED. CONTROL-X does not affect the template used by the special editing commands.



6 The Line Editor (Edlin)

In this chapter you will learn:

- How to start **edlin**, the line editor program
- How to quit **edlin**
- How to use the MS-DOS special editing keys with **edlin**

For information on specific **edlin** commands, see Chapter 7, "Edlin Commands."

About Edlin

You can use the MS-DOS line editor, **edlin**, to create text files and save them on your disks, or to update existing files, saving both the original and the updated files. Or, with **edlin** you can delete, edit, insert, and display lines in files. It will also help you search for, and delete or replace, text within your files. And, though it isn't a word processor, **edlin** does make it easy for you to create and revise files such as memos, letters, reports, or GW-BASIC programs.

How Edlin Works

Edlin divides the text from a file into lines, each line containing up to 253 characters. It gives each line a number and always numbers the lines consecutively. But, even though you see these line numbers on the screen when you use **edlin**, they are not part of the file.

When you insert lines of text in a file, the line numbers after the inserted text are automatically adjusted. Similarly, when you delete lines in a file, the line numbers following the deleted text are automatically renumbered.

What can Edlin do?

How does Edlin work?

How to Start Edlin

Starting Edlin

To start **edlin**, you simply type the word **edlin** followed by a *filename*. If you are creating a new file, *filename* should be the name or pathname of the file you wish to create. If **edlin** does not find this file on the default disk drive, it creates a new file with the name or pathname that you specify. For example, if you want to create a file called *budget.jun*, you would type the following command and then press the RETURN key:

```
edlin budget.jun
```

Creating a new file with Edlin

Edlin would then display the following:

```
New file
*_
```

Note that the **edlin** prompt is an asterisk (*).

To begin entering text you must type an **I** (insert) command to insert lines. The **I** command is discussed later in this chapter. For now you can type lines of text into your file, or use any of the **edlin** commands. These are discussed in more detail in Chapter 7, "Edlin Commands."

Note Be sure to press the RETURN key at the end of each line.

Editing an existing file with Edlin

Suppose you want to edit an existing file called *budget.may*. To do this you would type the following:

```
edlin budget.may
```

Then, when **edlin** finds the *budget.may* file, it loads it into memory. If your computer has enough memory to load the entire file, **edlin** displays the following message:

```
End of input file
*
```

You can then edit the file by using **edlin** commands.

If the file is too large to be loaded into memory, **edlin** loads lines from the file until memory is 3/4 full, and displays the asterisk (*) prompt. You can then edit the portion of the file that is in memory.

To edit the rest of the file, you must save some of the edited lines on a disk to free memory. **Edlin** will then be able to load the remaining unedited lines from the disk into memory. Refer to the **W** (write) and **A** (append) commands in Chapter 7, "Edlin Commands," for instructions on editing large files.

How to Quit Edlin

Saving a file

When you finish your editing session, you can save your original file and the updated (new) file by using the **E** (end) command, discussed in Chapter 7, "Edlin Commands." **Edlin** renames your original file with the extension *.bak*, and saves the updated file with the filename and extension you gave when you started **edlin**.

Warning You cannot update a file with an extension of *.bak* because when you try to save your file, **edlin** will always save the original file as *.bak*, thereby losing your changes. If you need to edit such a file, rename it with another extension (using the MS-DOS **ren** command discussed in Chapter 3, "MS-DOS Commands"), and start **edlin** by using the new filename.

Using the special editing keys with Edlin

Special Editing Keys

To edit your text files you can also use the special editing keys and template introduced in Chapter 5, "MS-DOS Function and Editing Keys."

The following table summarizes the commands, codes, and functions of the special editing keys. Descriptions of these special editing keys follow the table:

Key	What it does
F1	Copies one character from the template to the new line.
F2	Copies all characters from the template to the new line, up to the character specified.
F3	Copies all remaining characters in the template to the screen.
DEL	Does not copy (skips over) a character.
F4	Does not copy (skips over) the characters in the template, up to the character specified.
ESC	Clears the current input and leaves the template unchanged.
INS	Enters/exits insert mode.
F5	Makes the new line the new template.

F1

F1**Purpose:**

Copies one character from the template to the current line.

Comment:

When you press the F1 key, **edlin** copies one character from the template to the current line, and turns insert mode off.

Example:

As an example of how to use the F1 key with **edlin**, type the following line:

```
1:*Sharpe Office Supplies.  
2:*_
```

At the beginning of the editing session, the cursor (shown by the underline) is at the beginning of the line. When you press the F1 key, **edlin** copies the first character *S* to line 2 as shown here:

```
F1  
2:*S_
```

Each time you press the F1 key one more character appears:

```
F1  
2:*Sh_
```

```
F1  
2:*Sha_
```

```
F1  
2:*Shar_
```

F2

F2**Purpose:**

Copies multiple characters up to a given character.

Comments:

When you press the F2 key, **edlin** copies all the characters, up to a given character, from the template to the current line. The given character is the one that you type immediately after F2. **Edlin** does not copy or display the given character on the screen, but it does copy and display the characters from the template up to the position of that character. Of course, if the template does not contain the character, **edlin** does not copy anything.

If you use the F2 key, you automatically turn off insert mode.

Example:

As an example of how to use the F2 key with **edlin**, type the following line:

```
1:*Sharpe Office Supplies.  
2:*_
```

Remember that at the beginning of the editing session, the cursor (shown by the underline) is at the beginning of the line. When you press the F2 key followed by the letter *c*, **edlin** copies the characters up to the *c* in the word *Office*.

```
F2 c  
2:*Sharpe Offi_
```

F3**F3****Purpose:**

Copies the template to the current line.

Comments:

When you press the F3 key, **edlin** copies the remaining characters in the template to the current line. No matter where the cursor is when you press the F3 key, **edlin** displays the rest of the line and leaves the cursor at the end of the line.

Example:

As an example of how to use the F3 key with **edlin**, type the following line:

```
1:*Sharpe Office Supplies.  
2:*_
```

Remember that at the beginning of the editing session, the cursor (shown by the underline) is at the beginning of the line. When you press the F3 key, **edlin** copies the characters in the template (shown in line 1) to the line with the cursor (shown in line 2):

```
F3  
2:*Sharpe Office Supplies._
```

Also, this command automatically turns off insert mode.

DEL

DEL**Purpose:**

Skips over one character in the template.

Comments:

Each time you press the DEL key, **edlin** skips over and does not copy the next character in the template. The action of the DEL key is similar to the F1 key, except that DEL skips a character in the template instead of copying it to the current line.

Example:

As an example of how to use the DEL key with **edlin**, type the following line:

```
1:*Sharpe Office Supplies.  
2:*_
```

Remember that at the beginning of the editing session, the cursor (shown by the underline) is at the beginning of the line. When you press the DEL key, **edlin** skips over the first character, S:

```
DEL  
2:*_
```

The cursor does not move as **edlin** changes the template. To see how much of the line has been skipped over, press the F3 key. This action moves the cursor past the last character of the line:

```
F3  
2:*harpe Office Supplies._
```

F4

F4**Purpose:**

Skips multiple characters in the template up to the specified character.

Comments:

When you press the F4 key, **edlin** skips over all characters up to a given character in the template. **Edlin** does not copy or display any of the characters up to and including the given character. Of course, if the template does not contain that character, **edlin** does not skip any characters.

Note that the action of the F4 key is similar to that of the F2 key, except F4 skips over characters in the template instead of copying them to the current line.

Example:

As an example of how to use the F4 key with **edlin**, type the following line:

```
1:*Sharpe Office Supplies.
2:*_
```

Remember that at the beginning of the editing session, the cursor (shown by the underline) is at the beginning of the line. When you press the F4 key followed by the letter *c*, **edlin** skips over all the characters in the template up to the *c* in the word *Office*:

```
F4 c
2:*_
```

The cursor does not move as **edlin** changes the template. To see how much of the line has been skipped over, press the F3 key to copy the template. This action displays the rest of the line and moves the cursor to the end of the line:

```
F3
2:*ce Supplies._
```

ESC

ESC**Purpose:**

Quits input and clears the current line.

Comments:

When you press the ESC key, **edlin** empties the current line and leaves the template unchanged. ESC also prints a backslash (\), RETURN, and LINEFEED, and turns insert mode off. The cursor (shown by the underline) is at the beginning of the line. If you then press the F3 key, **edlin** copies the template to the current line, making it appear as it was before you pressed the ESC key.

Example:

As an example of how to use the ESC key with **edlin**, type the following lines:

```
1: Sharpe Office Supplies.  
2:*The World Leader_
```

To cancel the current line, line 2, press ESC. Notice that a backslash appears on line 2 to tell you it has been canceled:

```
ESC  
2:*The World Leader\
```

Press the RETURN key to keep line 1, or to perform any other editing functions. Now if you press F3, **edlin** copies the original template to the line:

```
F3  
2:Sharpe Office Supplies.
```

INS

INS

Purpose:

Enters insert mode or replace mode.

Comments:

When you start **edlin**, you are automatically in replace mode.

The first time you press the INS key, **edlin** enters insert mode. In insert mode the cursor in the template does not move, but in the current line it moves as you insert each character. When you finish inserting characters and press the INS key again, **edlin** reenters replace mode with the cursor at the same character in the template as when you entered insert mode.

In insert mode, **edlin** puts characters that you type at the keyboard into the template *and* in front of the cursor on the current line.

In replace mode, **edlin** replaces characters in the template and on the current line with characters that you type at the keyboard.

Examples:

As an example of how to use the INS key with **edlin**, type the following line:

```
1:*Sharpe Office Supplies.
2:*_
```

Remember that at the beginning of the editing session, the cursor (shown by the underline) is at the beginning of the line. First, press the F2 and O keys:

```
F2 O
2:*Sharpe _
```

Second, press the INS key and type the word *Automatic* followed by a space:

```
INS Automatic
2:*Sharpe Automatic _
```

If you now press the F3 key, **edlin** copies the rest of the template to the line:

```
F3
2:*Sharpe Automatic Office Supplies._
```

If you press the RETURN key after entering insert mode, **edlin** does not copy the remainder of the template and ends the current line after the inserted text:

```
1: Sharpe Office Supplies.
2:*Sharpe Automatic
3:*_
```

To exit insert mode and enter replace mode, simply press the INS key again. Now all the characters you type will replace characters in the template.

For a further example of how to use the INS key, type the following line:

```
1:*Sharpe Office Supplies.
2:*_
```

Once again, remember that at the beginning of the editing session, the cursor (shown by the underline) is at the beginning of the line. Now type the following sequence of keys and words (the results are shown below each key sequence):

```
F2 c
2:*Sharpe Offi_

INS cial
2:*Sharpe Official_

INS Sharpeware.
2:*Sharpe Official Sharpeware._
```

Notice that you inserted *cial* and replaced *ce Supplies* with *Sharpeware*.

The template now contains *Sharpe Official Sharpeware*.

F5**Purpose:**

Creates a new template.

Comments:

When you press the F5 key, **edlin** copies the current line to the template and deletes the previous contents. Pressing F5 also displays an @ ("at" symbol), and outputs a RETURN and a LINEFEED. When you press F5, **edlin** empties the current line and turns off insert mode.

Note F5 performs the same function as the ESC key, except that it changes the template, printing an @ instead of a backslash.

Example:

As an example of how to use the F5 key with **edlin**, type the following line:

```
1:*Sharpe Office Supplies.
2:*_
```

Remember that at the beginning of the editing session, the cursor (shown by the underline) is at the beginning of the line. Now type the following sequence of keys and words (the results are shown below each key sequence):

```
F2 c
2:*Sharpe Offi_

INS cial
2:*Sharpe Official_

INS Sharpeware.
2:*Sharpe Official Sharpeware._
```

At this point, suppose you want to add a word at the beginning of this line, but you don't want to backspace and retype the whole line. Just press the F5 key to put the current line into the template:

```
1: Sharpe Office Supplies.
2:*Sharpe Official Sharpeware.@
```

F5

The @ shows that this new line is now the new template. To add the word *Introducing*, followed by a space, at the beginning of the line, press INS and type the following sequence of keys and words (the results are shown below each key sequence):

INS Introducing:
2:*Introducing: _

Then press the F3 key to insert the contents of the template.

F3
2:*Introducing: Sharpe Official Sharpeware._

7 Edlin Commands

This chapter describes in detail the **edlin** commands, listed in the following table.

Command	Name	What it does
A	Append	Appends lines.
C	Copy	Copies lines.
D	Delete	Deletes lines.
<i>line</i>	Edit	Edits a line or lines.
E	End	Ends editing.
I	Insert	Inserts lines of text.
L	List	Lists a range of lines.
M	Move	Moves a range of text to a specified line.
P	Page	Pages through a file 23 lines at a time.
Q	Quit	Quits the editing session without saving the file.
R	Replace	Replaces text.
S	Search	Searches for text.
T	Transfer	Transfers the contents of another file into the file being edited.
W	Write	Writes specified lines to disk.

Some Tips for Using Edlin Commands

Once you have started editing a file with **edlin**, you can use the **edlin** commands to edit lines of text in it. Here are a few things to remember when using **edlin** commands:

- You can use pathnames in commands. For example, by typing the following command you can edit a file named *report.may* in a subdirectory named *\sharpe\budget*:

```
edlin \sharpe\budget\report.may
```

- You can reference line numbers relative to the current line, which **edlin** shows with an asterisk (*). To indicate lines before the current line, use a *minus* sign with a number; to indicate lines after the current line, use a *plus* sign with a number. The following command, for example, lists 10 lines before the current line, the current line, and 10 lines after the current line:

```
-10,+10L
```

Note A capital L is used here for the L (list) command to avoid confusion with the number one. A lowercase l would work just as well.

- You can type **edlin** commands *with* or *without* a space between the line number and command. For example, for deleting line 6, the command **6d** is the same as **6 d**.
- You can type multiple commands on one command line. Just type them one after another.

But if you want to use the **edlin line** (edit) command to edit a specific line, you have to separate the line number from the other commands with a semicolon.

To end a string in an S (search) or R (replace) command, just press CONTROL-Z instead of the RETURN key. The following command line, for example, edits line 15, then displays lines 10 through 20 on the screen:

```
15;-5,+5L
```

The command line in the next example searches for the words *monthly budget* and then displays five lines before and five lines after the line that contains *monthly budget*. Also, in this example and in those that follow, you'll see control key

sequences. You don't really type the entire word, **CONTROL**; instead, you simply press the **CONTROL** key followed by the control character, **Z** or **C** or **V**.

```
smonthly budgetCONTROL-Z-5,+5L
```

If **edlin** can find any lines that contain the words *monthly budget*, then the displayed lines are the five lines before and the five lines after the current line. Note that the current line (the line with the asterisk) must be *before* the line or lines that contain the search string.

- You can insert a control character, such as **CONTROL-C**, into text by using the quote character, **CONTROL-V**, before it while in insert mode. **CONTROL-V** tells MS-DOS to recognize the next *capital* letter typed as a control character. You can also put a control character in an **S** (search) or **R** (replace) command by using the quote character. For example, the following command finds the first occurrence of **CONTROL-Z** in a file:

```
sCONTROL-V Z
```

The next example replaces all occurrences of **CONTROL-Z** in a file with *pencil*.

```
rCONTROL-VZCONTROL-Z pencil
```

And this next command replaces all occurrences of **CONTROL-C** with *pen*.

```
sCONTROL-VCCONTROL-Z pen
```

It is also possible to insert **CONTROL-V** into the text by typing **CONTROL-VV**.

- The **CONTROL-Z** character ordinarily tells **edlin**, "This is the end of the file." If you have **CONTROL-Z** characters elsewhere in your file, you must tell **edlin** that these other control characters do not mean end-of-file. To tell **edlin** to ignore the **CONTROL-Z** characters in the file and to show you the entire file, use the **/b** switch. For example, when you start **edlin** and want to ignore all **CONTROL-Z** characters in a file, you could use the **/b** option with the **edlin** command and your filename.

Using command options with Edlin

Edlin Command Options

Many **edlin** commands accept one or more options. The effect of a command option varies, depending on which command you use it with. The following list describes each option.

The Line Option

The *line* option is a line number that you type. Use a comma or space to separate the numbers from other line numbers, other options, and from the command.

You can specify *line* in one of three ways:

- | | |
|----------------|---|
| <i>number</i> | Any number less than 65534. If you specify a number larger than the largest existing line number, then <i>line</i> refers to the line after the last line number. |
| . (period) | If you specify a period for <i>line</i> , it refers to the current line number. <i>The current line is the last line you edited, not necessarily the last line you displayed.</i> Edlin marks the current line with an asterisk (*) between the line number and the first character. |
| # (pound sign) | The pound sign indicates the line after the last line number. If you type # for <i>line</i> , it is the same as typing the last line number plus one. |
| RETURN key | If you type a command and then press the RETURN key without any of the line markers in this list, edlin uses a default value for each command (default values may be different for each command). |

The Question Mark Option

The question mark (?) option tells **edlin** to ask you if the correct string has been found. You use the question mark only with the **R** (replace) and **S** (search) commands. Before continuing, **edlin** waits for you to type a *Y* or press the RETURN key for a "Yes" response, or to press any other key for a "No" response.

The Text Option

The *text* option specifies text to be found, to be replaced, or to replace other text. Use the *text* option only with the **S** (search) and **R** (replace) commands. You must end each string of text with a CONTROL-Z or a RETURN (see the **R** command for details). You shouldn't leave any spaces between strings of text or between a string of text and its command letter, unless you want those spaces to be part of the text.

The remaining pages in this chapter describe the **edlin** commands. Each description explains the purpose and correct usage (syntax) of the command. Also, each command has several comments and examples, which offer advice, help, and even some shortcuts for using **edlin**.

(A)ppend

Append**Purpose:**

Adds a specified number of lines from your disk to the file being edited in memory.

Syntax:

[*n*]**a**

Comments:

The *n* option specifies the number of lines that you want to read into memory. You need this command only if the file you want to edit is too large to fit into memory. When you start **edlin**, it reads as many lines as possible into memory.

To edit the remainder of the file that will not fit into memory, you must write lines that you have already edited onto your disk. Then you can load unedited lines from your disk into memory by using the **A** (append) command. Refer to the **W** (write) command in this chapter for information on how to write edited lines to your disk.

Note If you do not specify the number of lines to append, **edlin** adds lines to the available memory until it is 3/4 full, but does nothing if available memory is already 3/4 full.

After the **A** command reads the last line of the file into memory, **edlin** displays the message "End of input file."

Copy

(C)opy

Purpose:

Copies a range of lines to a specified line number, and when used with the *count* option, copies this range as many times as you want.

Syntax:

[line],[line],line[,count]c

Comments:

The first and second *line* options specify the range of lines that you want to copy. If you omit the first or second *line* option, **edlin** defaults to the current line. The third *line* option specifies the line before which **edlin** will place the copied lines.

You must not overlap the line numbers or you will get an "Entry error" message. For example, the following command would result in an error message:

```
3,20,15c
```

If you do not specify a number for the *count* option, **edlin** copies the lines one time and automatically rennumbers the file after the copy.

Examples:

As an example of how to use the C (copy) command with **edlin**, type the following lines:

```
1: Sharpe Office Supplies
2: The World Leader in Office Sharpeware
3: I.R. Sharpe, President
4:
5: "You oughta be Sharpe too."
```

You could copy this entire block of text by typing the following command:

```
1,5,6c
```

Copying text

This command copies lines 1 through 5 and duplicates them one time, beginning on line 6. The result is:

```
1: Sharpe Office Supplies
2: The World Leader in Office Sharpeware
3: I.R. Sharpe, President
4:
5: "You oughta be Sharpe too."
6: Sharpe Office Supplies
7: The World Leader in Office Sharpeware
8: I.R. Sharpe, President
9:
10: "You oughta be Sharpe too."
```

Inserting the copied text

Now if you want to place this text within other text, you would specify the third *line* option as the line you want the copied text to appear before.

For example, suppose you want to copy lines and insert them in the middle of the file. The command **7,10,5c** would result in the following:

```
1: Sharpe Office Supplies
2: The World Leader in Office Sharpeware
3: I.R. Sharpe, President
4:
5: The World Leader in Office Sharpeware
6: I.R. Sharpe, President
7:
8: "You oughta be Sharpe too."
9: "You oughta be Sharpe too."
10: Sharpe Office Supplies
11: The World Leader in Office Sharpeware
12: I.R. Sharpe, President
13:
14: "You oughta be Sharpe too."
```

Delete

(D)elete

Purpose:

Deletes a specified range of lines in a file.

Syntax:

[*line*][,*line*]d

Comments:

If you omit the first *line* option, **edlin** defaults to the current line (the line with the asterisk next to the line number). If you omit the second *line* option, then **edlin** deletes just the first *line*. Remember, too, that when you delete lines, **edlin** automatically renumbers the file.

Examples:

Suppose that the following file exists and is ready to edit:

```
1: Dear Mr. Dimm,  
2:  
3: I was sorry to hear of your recent  
4: hospitalization due to electrical  
5: shock from our X-1000 Automatic  
6: Pencil Sharpener.  
7:  
8: As a result of your accident, we  
9: are redesigning our manual  
10: to warn our customers against trying  
11: to sharpen metal objects.  
12:  
13: We hope that you will be more  
14: careful with electrical appliances  
15: in the future.  
16:  
17: Sincerely,  
18:  
19: *I.R. Sharpe, President
```

But now, say you decide not to warn Mr. Dimm about being careful. To delete lines 12 through 15, type the following:

12,15d

**Deleting text from
a file**

The result of this command is:

```
1: Dear Mr. Dimm,  
2:  
3: I was sorry to hear of your recent  
4: hospitalization due to electrical  
5: shock from our X-1000 Automatic  
6: Pencil Sharpener.  
7:  
8: As a result of your accident, we  
9: are redesigning our manual  
10: to warn our customers against trying  
11: to sharpen metal objects.  
12:  
13: Sincerely,  
14:  
15:*I.R. Sharpe, President
```

Closing up empty space in a file

To close up the space you might decide to delete line 7. You could just type the command **7d**. The result would be:

```
1: Dear Mr. Dimm,  
2:  
3: I was sorry to hear of your recent  
4: hospitalization due to electrical  
5: shock from our X-1000 Automatic  
6: Pencil Sharpener.  
7:*As a result of your accident, we  
8: are redesigning our manual  
9: to warn our customers against trying  
10: to sharpen metal objects.  
11:  
12: Sincerely,  
13:  
14: I.R. Sharpe, President
```

Suppose, though, that you just want a quick message that doesn't take responsibility for the accident. To delete a range of lines beginning with the current line, line 7, through line 11, type the following:

```
,11d
```


The result of this command is:

```
1: Dear Mr. Dimm,  
2:  
3: I was sorry to hear of your recent  
4: hospitalization due to electrical  
5: shock from our X-1000 Automatic  
6: Pencil Sharpener.  
7:  
8: Sincerely,  
9:  
10: I.R. Sharpe, President
```

Notice that as you delete lines, **edlin** automatically renumbers the remaining lines in the file.

Edit

Edit**Purpose:**

Edits a line of text.

Syntax:

[*line*]

Comments:

The *line* option specifies the line of text you want to edit. When you type a line number as a command, **edlin** displays the line number and the text on that line; then, on the line below, **edlin** reprints the line number. Now you can retype the line, or use the **edlin** editing keys to edit it. The existing text of the line serves as the template until you press the RETURN key.

If you do not type a line number (that is, if you only press the RETURN key), **edlin** edits the line after the current line. (The current line is shown by an asterisk (*)).

If you do not need to change the current line and if the cursor is at the beginning or end of the line, you can simply press the RETURN key to accept the line.

Warning If you press the RETURN key while the cursor is in the middle of a line, **edlin** deletes the remainder of the line.

Example:

Suppose that the following file exists and is ready to edit:

```
1: Dear Mr. Dimm,  
2:  
3: I was sorry to hear of your recent  
4: hospitalization due to electrical  
5: shock from our Automatic  
6: Pencil Sharpener.
```

Editing a file

In line 5, say you want to insert the product's name, X-1000. To edit line 5, type the number 5. **Edlin** then displays the contents of the line with the cursor below the line:

```
5:*shock from our Automatic  
5:*_
```

Now you simply use the F2 key to skip to the A in the word *Automatic*, and type:

F2 A INS X-1000

5:*shock from our X-1000

F3 RETURN

5:*shock from our X-1000 Automatic

At the **edlin** prompt, type **L** to see the file:

1: Dear Mr. Dimm,

2:

3: I was sorry to hear of your recent

4: hospitalization due to electrical

5:*shock from our X-1000 Automatic

6: Pencil Sharpener.

Inserting text in a file

Displaying a file

(E)nd

End**Purpose:**

Ends the editing session.

Syntax:

e

Comments:

The **E** (end) command saves the edited file on your disk, renames the original input file *filename.bak*, and then exits **edlin**. If you created the file during the current editing session, **edlin** does not create a backup (*.bak*) file.

The **E** command takes no options. This means that you must select the drive that you want to save the file on *when you start edlin*; if you don't, **edlin** saves the file on the disk in the default drive. However, you can still copy the file to a different drive by using the MS-DOS **copy** command.

Before using the **E** command to save your file, make sure that the disk contains enough free space for the entire file. If it doesn't, **edlin** may not be able to write the entire file to the disk. The edited file will be lost, even though **edlin** may have saved part of it on the disk.

Example:

To end your editing session, type **E** and then press RETURN. After **edlin** processes the **E** command, the screen displays the MS-DOS prompt (for example, **A>**).

Ending your editing session

Insert

(I)insert

Purpose:

Inserts text immediately before the specified *line*.

Syntax:

[*line*]i

Comments:

If you are creating a new file, you must type the **I** (insert) command before you can insert a new line of text. Text begins on line 1, and each time you press the RETURN key the next line number appears automatically.

Edlin remains in insert mode until you press CONTROL-C. When you finish the insertion and exit insert mode, the line immediately following the inserted lines becomes the current line. **Edlin** automatically increments the line numbers that follow the inserted section by the number of lines that you inserted.

If you do not specify *line*, the default is the current line number and **edlin** inserts the lines before the current line. If *line* is a number larger than the last line number, or if you specify a pound sign (#) as *line*, **edlin** appends the inserted lines to the end of the file. In this case, the last line that you inserted becomes the current line.

Examples:

Suppose the following file exists and is ready to edit:

```
1: Dear Mr. Dimm,
2:
3: I was sorry to hear of your recent
4: hospitalization due to electrical
5: shock from our X-1000 Automatic
6: Pencil Sharpener.
7:
8: Sincerely,
9:
10: I.R. Sharpe, President
```

Since this letter doesn't really offer any compensation for the accident, you might decide to add a brief note about a small gift that you're enclosing for Mr. Dimm. To insert text before line 8, type **8i**. The result is:

```
8: *_
```

Inserting text in a file

Now type the following lines, which will begin on line 7:

```
8:*As a result of your accident, we
```

Press the RETURN key at the end of each line, and continue typing the next line:

```
9:*are redesigning our manual to
10:*warn our customers against trying
11:*to sharpen metal objects.
```

Displaying the changed file

To end the insertion, press CONTROL-C on the *next* line, and then type L to list the file. The result is:

```
1: Dear Mr. Dimm,
2:
3: I was sorry to hear of your recent
4: hospitalization due to electrical
5: shock from our X-1000 Automatic
6: Pencil Sharpener.
7:
8: As a result of your accident, we
9: are redesigning our manual to
10: warn our customers against trying
11: to sharpen metal objects.
12:*Sincerely,
13:
14: I.R. Sharpe, President
```

Inserting a blank line in a file

To insert a blank line immediately before the current line (line 12), type i. The result is:

```
12:*_
```

Insert a blank line by pressing RETURN, and end the insertion by pressing CONTROL-C on the next line. Then, to list the file and see the result, type L.

1: Dear Mr. Dimm,
2:
3: I was sorry to hear of your recent
4: hospitalization due to electrical
5: shock from our X-1000 Automatic
6: Pencil Sharpener.
7:
8: As a result of your accident, we
9: are redesigning our manual to
10: warn our customers against trying
11: to sharpen metal objects.
12:
13: *Sincerely,
14:
15: I.R. Sharpe, President

To append new lines to the end of the file, type 16i. This produces the following:

**Adding lines to the
end of a file**

16: *_

Now type the following new lines:

16: Sharpe Office Supplies
17: The World Leader in Office Sharpeware
18: Our motto: "You oughta be Sharpe too"

To get out of insert mode, press CONTROL-C on line 19. The new lines will appear at the end of all previous lines in the file. Now list all the lines by typing 1,23L. The result is:

Quitting insert mode

1: Dear Mr. Dimm,
2:
3: I was sorry to hear of your recent
4: hospitalization due to electrical
5: shock from our X-1000 Automatic
6: Pencil Sharpener.
7:
8: As a result of your accident, we
9: are redesigning our manual to
10: warn our customers against trying
11: to sharpen metal objects.
12:
13: Sincerely,
14:
15: I.R. Sharpe, President
16: Sharpe Office Supplies
17: The World Leader in Office Sharpeware
18: Our motto: "You oughta be Sharpe too"

(L)ist

List**Purpose:**

Lists a range of lines, including the two lines specified.

Syntax:

[*line*][,*line*]**L**

Comments:

A capital letter **L** has been used here to avoid confusion with the number one. A small letter **l** would work just as well.

Edlin provides default values if you do not type either option. If you do not type the first *line* option, as in the following command, the display will start 11 lines before the current line and end with the specified *line*:

,*line* **L**

The beginning comma is needed to show that you omitted the first *line* option.

Note If the specified *line* is more than 11 lines before the current line, the display will be the same as if you omitted both options.

Edlin displays 23 lines, starting with the specified *line*, if you omit the second option, as in the following command:

line **L**

If you just type **L**, **edlin** displays 23 lines—the 11 lines before the current line, the current line, and the 11 lines after the current line. If there are less than 11 lines before the current line, **edlin** displays more than 11 lines after the current line, up to a total of 23 lines.

Examples:

Suppose the following file exists and is ready to edit:

```

1: Dear Mr. Dimm,
2:
3: I was sorry to hear of your recent
4: hospitalization due to electrical
5: shock from our X-1000 Automatic
6: Pencil Sharpener.
.
.
.
15: We also intend to attach a metal
16: detector to future models.
.
.
.
26: I.R. Sharpe, President
27: Sharpe Office Supplies
28: The World Leader in Office Sharpeware
29: Our motto: "You oughta be Sharpe too"

```

To list a range of lines *without* referring to the current line, type **2,6L**. The result is:

**Listing ranges of
lines in a file**

```

2:
3: I was sorry to hear of your recent
4: hospitalization due to electrical
5: shock from our X-1000 Automatic
6: Pencil Sharpener.

```

To list a range of lines *beginning* with the current line, type **16,L** or **,L**. The result is:

```

16:*detector to future models.
.
.
.
26: I.M. Sharpe, President
27: Sharpe Office Supplies
28: The World Leader in Office Sharpeware
29: Our motto: "You oughta be Sharpe too"

```

To list a range of 23 lines *centered* around the current line, type

L. The result is:

5: shock from our X-1000 Automatic

6: Pencil Sharpener.

.

.

.

15: We also intend to attach a metal

16:*detector to future models.

.

.

.

26: I.R. Sharpe, President

27: Sharpe Office Supplies

Move

(M)ove

Purpose:

Moves a range of text to the specified line.

Syntax:

[*line*],+*line*,*linem*

Comments:

The **M** (move) command lets you move a block of text to another location in a file. The first and second *line* options specify the range of lines that you want to move. The third *line* option specifies the line you want to move the first line in the range to.

Edlin automatically renumbers the lines after it moves them. For example, the following command moves the text from the current line—plus 25 lines—to line 100:

```
,+25,100m
```

If the line numbers that you specify overlap, **edlin** displays an "Entry error" message.

Example:

Suppose the following file exists and is ready to edit.

```
1: Dear Mr. Dimm,
2:
3: I was sorry to hear of your recent
4: hospitalization due to electrical
5: shock from our X-1000 Automatic
6: Pencil Sharpener.
7:
8: As a result of your accident, we
9: are redesigning our manual to
10: warn our customers against trying
11: to sharpen metal objects.
12:
13: Sincerely,
14:
15: I.R. Sharpe, President
16: Sharpe Office Supplies
17: The World Leader in Office Sharpeware
18: Our motto: "You oughta be Sharpe too"
```

**Moving a block of text
in a file**

What if you prefer to have the motto at the start of the letter? If so, you could move lines 16–18 to line 1 by typing the following command:

16,18,1m

The result of this command is:

```
1: Sharpe Office Supplies
2: The World Leader in Office Sharpeware
3: Our motto: "You oughta be Sharpe too"
4: Dear Mr. Dimm,
5:
6: I was sorry to hear of your recent
7: hospitalization due to electrical
8: shock from our X-1000 Automatic
9: Pencil Sharpener.
10:
11: As a result of your accident, we
12: are redesigning our manual to
13: warn our customers against trying
14: to sharpen metal objects.
15:
16: Sincerely,
17:
18: I.R. Sharpe, President
```

Page

Purpose:

Displays a file one page (23 lines) at a time.

Syntax:

[*line*][,*line*]p

Comments:

The first *line* option specifies the line at which **edlin** starts displaying. The second *line* option specifies how many lines appear on each page. If you do not type the first *line*, **edlin** starts the page at the line after the current line. If you do not type the second *line* option, **edlin** lists 23 lines on each page.

(P)age

Displaying a file

(Q)uit

Quit**Purpose:**

Quits the editing session, does *not* save any editing changes, and exits to the MS-DOS operating system.

Syntax:

q

Comments:

This command is useful if you don't want to make any changes to a file. If you use the **Q** (quit) command, **edlin** prompts you to make sure you don't want to save the changes. If you do want to save your changes, you should refer to the **E** (end) command for more information.

If you want to quit the editing session, type *Y* (for Yes) when prompted; remember, though, that **edlin** will not save your editing changes and will not create a backup (*.bak*) file. Refer to the **E** command in this chapter for information about the *.bak* file.

If you want to continue the editing session, type *N* (for No).

Warning When you exit **edlin**, it erases any previous copy of the file that has a *.bak* extension. If you reply *Y* to the "Abort edit (Y/N)?" message, **edlin** deletes your previous backup copy.

Example:

The following example shows how to quit **edlin** without saving your changes. First, to get the "Abort edit (Y/N)?" message, type **q**. Then, when the message appears, type *Y* and press RETURN.

Quitting Edlin

Replace

(R)eplace

Purpose:

Replaces all occurrences of a string of text in a range with a different string of text.

Syntax:

`[line][,line][?]rtext1 CONTROL-Z text2`

Comments:

The first and second *line* options specify the range of lines that the **R** (replace) command uses. Each time **edlin** finds *text1*, it replaces it with *text2*, displaying each line that changes. If a line contains two or more replacements, it is displayed once for each change. When **edlin** has made all the changes, the **R** command ends and the asterisk prompt reappears.

If you want to replace one string of text with another, you must separate the two with a CONTROL-Z. You can end the second string by pressing the RETURN key.

If you do not specify *text1*, the **R** command assumes the old (the previous) value. If this is the first replacement that you have made during the current editing session, and if you do not specify *text1*, the command ends. If you do not specify *text2*, you must end *text1* by pressing the RETURN key.

If you omit the first *line* option, **edlin** uses the line after the current line, by default. The default for the second *line* option is *#*. Remember that *#* refers to the line after the last line of the file.

If you end *text1* with a CONTROL-Z and do not specify *text2*, **edlin** assumes you want blank spaces for *text2*. For example, suppose you want to delete all occurrences of the word *clients* from your file. To do this you could simply type the following command, then press CONTROL-Z and RETURN:

```
rclients
```

The next command replaces *clients* with the previous *text2*:

```
rclients
```

The following command makes the previous *text1* become the previous *text2*:

Deleting all occurrences of a word from a file

Replacing a string of text throughout a file

Note that "previous" refers to an earlier string of text specified in an **S** or **R** command.

If you specify a question mark (?) in the **R** command, **edlin** stops at each line with text that matches *text1*, displays the line with *text2*, and then displays the prompt "O.K.?". If you press **Y** or **RETURN**, *text2* replaces *text1*, and **edlin** looks for the next occurrence of *text1*. It continues this process until it reaches the end of the range or the end of the file. After **edlin** finds the last *text1*, it displays the asterisk prompt.

If you press any key other than **Y** or the **RETURN** key after the "O.K.?" prompt, *text1* is left as is, and **edlin** moves to the next occurrence of *text1*. If *text1* occurs more than once in a line, **edlin** individually replaces each occurrence of *text1*, and displays the "O.K.?" prompt after each replacement. In this way you can replace only the desired *text1* strings and prevent unwanted substitutions.

Examples:

Suppose the following file exists and is ready for editing:

```

1: Dear Mr. Dimm,
2:
3: I was sorry to hear of your recent
4: hospitalization due to electrical
5: shock from our X-1000 Automatic
6: Pencil Sharpener.
7:
8: As a result of your accident, we
9: are redesigning our manual to
10: warn our customers against trying
11: to sharpen metal objects.
12:
13: Sincerely,
14:
15: I.R. Sharpe, President

```

Now, suppose that in lines 5 through 10 you want to replace all occurrences of the word *our* with the word *the*. To do this you would simply type **5,10 rour**, press **CONTROL-Z**, type *the*, and press the **RETURN** key. The result is:

```

5: shock from the X-1000 Automatic
8: As a result of ythe accident,
9: are redesigning the manual to
10: warn the customers against

```


In the previous example, some unwanted changes occurred. To avoid these and to confirm each replacement, you can use the same file with a slightly different command.

In the next example you will see how to replace only certain occurrences of *our* with *the*. At the **edlin** prompt, type the following sequence of keys and words, and then press the RETURN key:

```
1,15? rour CONTROL-Z the
```

The result is:

```
5: shock from the X-1000 Automatic
0.K.? y
8: As a result of ythe accident, we
0.K.? n
9: are redesigning the manual to
0.K.? y
10: warn the customers against trying
0.K.? n
*_
```

Type the list command, **L**, to see the result of all these changes:

```
.
5: shock from the X-1000 Automatic
.
8: As a result of your accident, we
9: are redesigning the manual to
10: warn our customers against trying
.
```

Replacing text selectively

(S)earch

Search**Purpose:**

Searches a range of lines for a string of text.

Syntax:

[*line*][,*line*][?]*stext*

Comments:

The first and second *line* options specify the range of lines for **edlin** to search. You end the *text* option by pressing the RETURN key. **Edlin** displays the first line that matches the string; that line then becomes the current line. Unless you type the question mark (?) option, the **S** (search) command ends when it finds the first match. If **edlin** cannot find a line with a match, it displays the message "Not found."

If you include the question mark option (?), **edlin** displays the first line with matching text and prompts you with the message "O.K.?". If you press either *Y* (Yes) or the RETURN key, this line becomes the current line and the search ends. If you press any other key, the search continues until another match is found, or until all lines have been searched. (The search ends when **edlin** displays the "Not found" message.)

If you don't type the first line number, **edlin** defaults to the line *after* the current line; if you don't type the second line number, it defaults to # (the line after the last line of the file).

If you omit the *text* option, **edlin** uses the text from the previous **S** or **R** (replace) command. If this is the first **S** or **R** command you have used during the current session, and you have not specified a search string, the **S** command ends immediately.

Examples:

Suppose the following file exists and is ready for editing:

```

1: Dear Mr. Dimm,
2:
3: I was sorry to hear of your recent
4: hospitalization due to electrical
5: shock from our X-1000 Automatic
6: Pencil Sharpener.
7:
8: As a result of your accident, we
9: are redesigning our manual to
10: warn our customers against trying
11: to sharpen metal objects.
12:
13: Sincerely,
14:
15: I.R. Sharpe, President

```

To search for the first occurrence of the word *to*, type the command **2,12 sto** and press the RETURN key. **Edlin** displays the following lines:

```

3: I was sorry to hear of your recent

```

To get the *to* in line 4, modify the S command by pressing DEL F3 followed by the RETURN key. The search continues at the line after the current line (i.e., the search starts at line 4), because you deleted the first line number in the template. The result is:

```

4: hospitalization due to electrical

```

To search through several occurrences of a string until the correct string is found, type the command **1, ? sto**. The result is:

```

3: I was sorry to hear of your recent
D.K.?_

```

If you press any key (except Y or the RETURN key), the search continues, so type N here:

```

D.K.? n

```

Continue:

```

4: hospitalization due to electrical
D.K.?_

```

**Searching for a word
in a file**

Now press *Y* to terminate the search:

O.K.? y

* _

Edlin reports a match and continues to search for the same string when you retype the **S** command and press RETURN:

S

Edlin reports another match, if there is one.

S

Edlin displays the "Not found" message at the end of the search.

Note that the text string defaults to any string specified by a previous **R** or **S** command.

Transfer

(T)ransfer

Purpose:

Inserts, at a specific line number, the contents of another file into the file that you are currently editing.

Syntax:

[line]tfilename

Comments:

The **T** (transfer) command puts the contents of one file into another file, or into the text you are typing. **Edlin** inserts *filename* at the line number you give in the *line* option, and then automatically renumbers the lines. If you omit the line number, **edlin** inserts the text, beginning on the current line.

Example:

To copy a file named *irsharpe.mem* to line 12 of the file you are editing, use the following command:

```
12 t irsharpe.mem
```

Inserting one file into another

(W)rite

Write**Purpose:**

Writes a specific number of lines to a disk.

Syntax:

[*n*]w

Comments:

The *n* option specifies the number of lines that you want to write to the disk. You need this command only if the file you are editing is too large to fit into memory. When you start **edlin**, it reads lines from your file until memory is 3/4 full.

To edit the remainder of your file, you must write the edited lines in memory to your disk. Then you can load additional unedited lines from your disk into memory by using the **A** (append) command, which is described earlier in this chapter.

Note If you do not specify the number of lines for **edlin** to write, it writes lines until memory is 3/4 full. But it does not write any lines to your disk until memory is more than 3/4 full. Also, **edlin** rennumbers all of the lines so that the first remaining line becomes line number 1.

8 File Comparison Utility (FC)

In this chapter you will learn:

- How to compare files line by line
- How to compare files byte by byte

Introduction

What is FC?

It is sometimes useful to compare files on your disk. If you have copied a file and later want to compare copies to see which one is current, you can use the MS-DOS file comparison utility, FC.

The FC utility compares the contents of two files, or two sets of files. The differences between the two files can be sent to the screen or to a third file. The files being compared must be text files—unless you specify the **/b** switch.

The comparisons are made in one of two ways:

- line by line
- byte by byte

The line-by-line comparison isolates blocks of lines that are different between two files and prints those blocks of lines. You use this type of comparison to check for differences between text files (also known as ASCII files). The byte-by-byte comparison displays the bytes that are different between two files. You use this type of comparison to check for differences between binary files.

Limitations on Comparisons

FC uses a large amount of memory (enough to hold 100 lines) as buffer storage space to hold the text files. If these files are larger than available memory, FC will compare what it can load into the buffer space. If it doesn't find a match in those portions of the files in the buffer space, FC stops and displays the following message:

```
resynch failed. Files are too different.
```

For binary files larger than available memory, FC compares both files completely, overlaying the portion in memory with the next portion from disk. All differences are output in the same manner as for those files that fit completely in memory.

How to Use FC

The syntax of FC is as follows:

For ASCII comparison —

```
fc [/a] [/c] [/L] [/Lb n] [/n] [/t] [/w][/nnnn] filename1 filename2
```

For binary comparison —

```
fc [/b] [/nnnnn] filename1 filename2
```

The *filename1* option specifies the first file or set of files that you want to compare, while the *filename2* option specifies the second file or set of files that you want to compare. FC matches the first file against the second and reports any differences between them.

For example, to compare two text files called *monthly.rep* and *sales.rep*, you would type the following command:

```
fc /a monthly.rep sales.rep
```

If you want to specify a set of files, you simply use a wildcard as part of the filename. For example, if Pete and Betty both have files called *report1.txt*, *report2.txt*, and *report3.txt*, they might use FC to see if their files are identical. They would use the following command:

```
fc c:\user\pete\report?.txt \user\betty\report?.txt
```

Comparing text files

FC would take *report1.txt* in the `\user\pete` directory of disk drive C and compare it with the corresponding *report1.txt* in the `\user\betty` directory. It would then compare Pete's *report2.txt* against Betty's, and, finally, compare their *report3.txt* files.

Since in this example no drive was specified for the second set of files, FC would have assumed that the `\user\betty` directory was on the disk in the default drive.

FC Switches

Using switches with FC

There are nine switches that you can use with the File Comparison utility. They are described in the following paragraphs.

The `/a` switch abbreviates the output of an ASCII comparison. Instead of displaying all of the lines that are different, only the lines that begin and end each set of differences are displayed. The intermediate lines are represented by ellipses (...).

The `/b` switch forces a binary comparison of both files. The two files are compared byte by byte, with no attempt to resynchronize after a mismatch. The mismatches are printed as follows:

```
xxxxxxxx yy zz
```

(where `xxxxxxxx` is the relative address of the pair of bytes from the beginning of the file). Addresses start at 00000000; `yy` and `zz` are the mismatched bytes from *file1* and *file2*, respectively. If one of the files contains less data than the other, a message is displayed. For example, if *file1.txt* ends before *file2.txt*, FC displays:

```
fc: file2.txt longer than file1.txt
```

This switch is the default when you compare `.exe`, `.com`, `.sys`, `.obj`, `.lib`, or `.bin` files.

The `/c` switch causes the matching process to ignore the case of letters. All letters in the files are considered uppercase letters. For example, the following two lines would be considered matching:

```
Much MORE data IS NOT FOUND
```

```
much more data is not found
```

If both the `/w` and `/c` options are specified, then FC compresses white space and ignores case. For example, if an underscore represents a white space, then the following two lines match:

```
___DATA_was_found___
```

```
data_was_found
```

The `/L` switch compares the files in ASCII mode. This switch is the default when you compare files that do not have extensions of `.exe`, `.com`, `.sys`, `.obj`, `.lib`, or `.bin`.

The `/Lb` switch sets the internal line buffer to *n* lines. The default length of the internal buffer is 100 lines. Files that have more than this number of consecutive, differing lines will abort the comparison.

The `/n` switch displays the line numbers on an ASCII comparison.

The `/t` switch does not expand tabs to spaces. The default is to treat tabs as spaces to 8-column positions.

The `/w` switch causes FC to compress white space (tabs and spaces) during the comparison. If a line contains several spaces or tabs in a row, these characters are considered a single white space.

Note that although FC *compresses* white space, it does not ignore it. The exception is beginning or ending white space in a line, which is ignored. For example, of the next four examples, the first three match, but the fourth does not (note that an underscore represents a white space):

```
___More__data_to_be_found___
```

```
More_data_to_be_found
```

```
_____More_____data_to_be_____found_____
```

```
____Moredata_to_be_found
```

The `/nnnn` switch specifies the number of lines that must match after FC finds a difference between files. If the number of matching lines in the files is less than this number, FC displays these matching lines as differences.

How FC Reports Differences

FC reports the differences between the two files you specify by displaying the first filename, followed by the lines that differ between the files, followed by the first line to match in both files. FC then displays the name of the second file, followed by the lines that are different, followed by the first line that matches.

The default for the number of lines to match between the files is 2. (If you want to change this default, specify the number of lines with the */nnnn* switch.)

Example:

```
...
...
**** file1
difference
1st matching line in file1 and file2
next matching line in file1 and file2
**** file2
difference
1st matching line in file1 and file2
next matching line in file1 and file2
...
...
```

If the number of lines in the internal buffer is less than the number of consecutive, differing lines, the program will stop. FC then displays the following message:

```
resynch failed. Files are too different.
```

FC then displays the contents of the internal buffer and returns to the MS-DOS default drive prompt (for example, A>).

Redirecting FC Output to a File

Unless you redirect the FC output to a file, the differences and matches between the two files you specify will be displayed on your screen. You can easily redirect the FC output using the MS-DOS redirection symbols (see Chapter 2, "About Commands").

To compare *file1.txt* and *file2.txt* and then send the FC output to *differ.txt*, type:

```
fc file1.txt file2.txt > differ.txt
```

The differences and matches between *file1.txt* and *file2.txt* will be put into *differ.txt* on the default drive.

Examples

Assume that the text files *pencil.ad* and *pen.ad* exist on your disk and contain the following:

Pencil.ad

Introducing ...
The X-1000 Automatic Pencil Sharpener
From Sharpe Office Supplies
The World Leader in Office Sharpeware

This model is compact, and sharpens
pencils more efficiently than ever before.

Our Motto:
"You oughta be Sharpe too!"

I.R. Sharpe, President

Pen.ad

Introducing ...
The X-2000 Automatic Pen and Pencil Sharpener
From Sharpe Office Supplies
The World Leader in Office Sharpeware

This model is compact, and sharpens pens and
pencils more efficiently than ever before.

Our Motto:
"You oughta be Sharpe too!"

I.R. Sharpe, President

Example 1:

To compare the two files and display the differences on the screen, type:

```
fc pencil.ad pen.ad
```

FC compares *pencil.ad* with *pen.ad* and displays the differences on the screen. All other defaults remain intact. (The defaults are: perform an ASCII comparison with an internal buffer length of 100 lines, treat tabs as spaces to 8-column positions, do not ignore case, do not display line numbers, and do not compress white space.)

The following output would appear on the screen:

```
***** pencil.ad
Introducing ...
The X-1000 Automatic Pencil Sharpener
From Sharpe Office Supplies
***** pen.ad
Introducing ...
The X-2000 Automatic Pen and Pencil Sharpener
From Sharpe Office Supplies
*****
```

```
***** pencil.ad
```

```
This model is compact, and sharpens
pencils more efficiently than ever before.
***** pen.ad
```

```
This model is compact, and sharpens pens and
pencils more efficiently than ever before.
*****
```

Comparing two text files

Printing the file differences**Example 2:**

You can print the differences on the lineprinter using the same two files. In this example, four successive lines must be the same to constitute a match. If four such lines do not match, they are considered different and are displayed.

First, type the following command:

```
fc /4 pencil.ad pen.ad > prn
```

The following output would appear on the lineprinter:

```
***** pencil.ad
Introducing ...
The X-1000 Automatic Pencil Sharpener
From Sharpe Office Supplies
The World Leader in Office Sharpeware

This model is compact, and sharpens
pencils more efficiently than ever before.
***** pen.ad
Introducing ...
The X-2000 Automatic Pen and Pencil Sharpener
From Sharpe Office Supplies
The World Leader in Office Sharpeware

This model is compact, and sharpens pens and
pencils more efficiently than ever before.
*****
```

Example 3:

This example forces a binary comparison and then displays the differences on the screen using the same two files from the previous two examples. First, type the following:

```
fc /b pencil.ad pen.ad
```

The **/b** switch in this example forces the binary comparison. You must type this switch and any others before the filenames in the FC command line. The following display would appear:

00000017	31	32
00000029	63	20
0000002A	69	61
0000002B	6C	6E
0000002C	20	64
0000002D	53	20
0000002E	68	50
0000002F	61	65
00000030	72	6E
00000031	70	63

...

...

...

0000010D	73	49
0000010E	69	2E
0000010F	64	52
00000110	65	2E
00000111	6E	20
00000112	74	53
00000113	20	68
00000114	0D	61
00000115	0A	72
00000116	1A	70

```
fc: pen.ad longer than pencil.ad
```

Forcing a binary file comparison



9 Link: A Linker

In this chapter you'll learn:

- How to create executable files with **link**
- How to use **link** command options
- How **link** creates programs

Introduction

The Microsoft® 8086 Object Linker (**link**) creates executable programs from object files generated by the Microsoft Macro Assembler (MASM) or by compilers for high-level languages, such as C or Pascal. The linker copies the resulting program to an executable (.exe) output file. You can then run the program by typing the file's name on the MS-DOS command line.

To use **link**, you must create one or more object files, then submit these files, along with any required library files, to the linker for processing. **Link** combines code and data in the object files and searches the named libraries to resolve external references to routines and variables. It then copies a relocatable execution image and the relocation information to the executable file. Using the relocation information, MS-DOS can load the executable image at any convenient memory location and then run it. **Link** can process programs that contain up to one megabyte of code and data.

This chapter explains how to use the linker to create executable programs. It also defines each of the options you can use in a **link** command line to control the linking process. Finally, it explains how **link** creates programs.

How to start Link

Starting and Using Link

This section explains three methods for starting and using the linker to create executable programs. These methods let you use **link** by answering a series of prompts, by typing an MS-DOS command line, or by using a response file. The three methods can also be mixed.

Once you start **link**, it will either process the files you supply or prompt you for additional files. Also, note that you can stop the linker at any time by pressing CONTROL-C.

Using Prompts to Specify Link Files

When you type the command, **link**, at the MS-DOS prompt, the linker prompts you for the information it needs. Follow these steps:

- 1 First, type the following command and press the RETURN key:

```
link
```

Link prompts you for the object files you wish to link by displaying the following message:

```
Object Modules [.OBJ]:
```

- 2 Type the name or names of the object files you wish to link. If you do not supply extensions for these files, **link** supplies *.obj* by default. If you have more than one name, make sure you separate each with spaces or plus signs (+). If you have more names than can fit on one line, type a plus sign (+) as the last character on the line and press the RETURN key. **Link** then prompts you for additional object files.

Once you have given all your object filenames, press the RETURN key. The linker displays the following prompt:

```
Run File [filename.EXE]:
```

- 3 Note that in step 2, *filename* is the same as the first *filename* you typed at the Object Modules prompt. Type the name of the executable file you wish to create, and press the RETURN key. If you do not give an extension, **link** supplies *.exe* by default. If you want **link** to supply a default executable filename, just press the RETURN key. The *filename* will then be the same as the first object file, but with the extension *.exe*.

Once you have pressed the RETURN key, **link** displays the following prompt:

List File [NUL.MAP]:

- 4 Type the name of the map file you wish to create, then press the RETURN key. If you do not supply a filename extension, the linker uses *.map* by default. If you don't want a map file, don't type a filename. Just press the RETURN key.

Once you have pressed the RETURN key, **link** displays the following prompt:

Libraries [.LIB]:

- 5 Type the names of any library files containing routines or variables referenced but not defined in your program. If you give more than one name, make sure the names are separated by spaces or plus signs (+). If you don't supply filename extensions, the linker uses *.lib* by default. If you have more names than can fit on one line, type a plus sign (+) as the last character on the line and press the RETURN key. **Link** then prompts you for additional filenames.

After entering all names, press the RETURN key. If you don't want to search any libraries, don't type any names; just press the RETURN key.

Link now creates the executable file.

When entering filenames, you must supply a pathname for any file that is not in the current drive and directory. You can use **link** options by typing them after the filename at any prompt. If the linker cannot find an object file, it displays a message and waits for you to change disks, if necessary.

At any prompt, you can type the rest of the filenames by using the command line format described in the next section, "Using a Command Line to Specify Link Files." For example, you can choose the default responses for all remaining prompts by typing a semicolon (;) after any prompt, or you can type commas (,) to indicate several files. (If you type a semicolon at the Object Modules prompt, be sure to supply at least one object filename.) When the linker encounters a semicolon, it immediately chooses the default responses and processes the remaining files without displaying any more prompts.

Using Link with prompts

Example:

The following example links the object modules *moda.obj*, *modb.obj*, *modc.obj*, and *startup.obj*; searches the library file *math.lib* on drive B of the *\lib* directory for routines and data used in the program; and creates an executable file named *moda.exe*, and a map file named *abc.map*. The **/pause** option in the Object Modules prompt line then causes **link** to pause while you change disks, after which the linker creates the executable file (see the section entitled "Pausing to Change Disks," later in this chapter):

```
link

Object Modules [.OBJ]: moda+modb+
Object Modules [.OBJ]: modc+startup/PAUSE
Run File [moda.EXE]:
List File [NUL.MAP]: abc
Libraries [.LIB]: b:\lib\math
```

Using a Command Line to Specify Link Files

You can create an executable program by typing **link**, followed by the names of the files you wish to process. The command line has the following general form:

```
link objectfiles [,executablefile] [,mapfile]
[,libraryfile]]] [options] [;]
```

The variables in this command line are described as follows:

<i>objectfiles</i>	Includes the name or names of object files that you want to link together. The files must have been created using MASM or a high-level-language compiler. The linker requires at least one object file. If you do not supply an extension, link provides the extension <i>.obj</i> .
<i>executablefile</i>	Is an optional placeholder for the name you wish to give the executable file that link will create. If you do not supply an <i>executablefile</i> , link creates a filename by using the name of the first object file in the command line and appending it with an <i>.exe</i> extension.

<i>mapfile</i>	Is the name of the file that receives the map listing. If you do not supply an extension, the linker provides the extension <i>.map</i> . If you specify the <i>/map</i> or <i>/linenumbers</i> option, the linker creates a map file even if you don't specify one in your command line.
<i>libraryfiles</i>	Includes the name or names of the libraries containing routines that you wish to link to create a program. If you do not supply an extension, link supplies the extension <i>.lib</i> .
<i>options</i>	Controls the operation of link . You can use any of the options listed in the section entitled "The Link Options." You can specify options anywhere on the command line.

If you do not specify a drive or directory for a file, **link** assumes the file is on the current drive and directory. You cannot specify the drive or directory for the *objectfile* and expect **link** to supply the same drive and directory for other files. Instead, you must give the location of each file specifically.

If you type the comma after the object file, **link** supplies the default name for the *executablefile* and suppresses the *mapfile* and *libraryfiles*. You can also use a semicolon (;) anywhere after the object file to terminate the command line.

If you do not supply all filenames in the command line and do not end the command line with a semicolon, the linker prompts you for additional files, using the prompts described previously in the section, "Using Prompts to Specify Link Files." If you give more than one object file or library file, you must separate the names by spaces or plus signs (+).

If you do not specify a drive or directory for a file, **link** assumes the file is on the current drive and directory. So, you cannot specify the drive or directory for the *objectfile* and expect **link** to supply the same drive and directory for other files. Instead, you must give the location of each file specifically.

Note When linking modules produced by a high-level-language compiler that supports overlays, you must specify overlay modules by putting them in parentheses. Since MASM has no overlay manager, you can specify overlays only for object files linked with the run-time library of a language compiler that supports overlays.

For example, you can use overlays with modules compiled with Microsoft FORTRAN versions 3.2 and later, Microsoft Pascal versions 3.2 and later, and Microsoft C versions 3.0 and later. See your language compiler's manual for details on specifying overlays.

Examples:

Using Link with command line options

The first example below uses the object file *file.obj* to create the executable file *file.exe*. **Link** searches the *file.lib* library for routines and variables used within the program. It also creates a file called *file.map*, which contains a list of the program's segments and groups:

```
link file.obj,file.exe,file.map,routine.lib
```

The first example is equivalent to the following line:

```
link file,,,routine
```

The second example uses the two object files, *startup.obj* and *file.obj*, on the current drive to create an executable file named *file.exe* on drive B. **Link** creates a map file on the *\map* directory of the current drive, but does not search any libraries:

```
link startup+file,b:file,\map\file;
```

The final example links the object modules *moda.obj*, *modb.obj*, *modc.obj*, and *startup.obj*:

```
link moda modb modc startup/PAUSE,,abc,b:\lib\math
```

The linker searches through the library file *math.lib* in the *\lib* directory on drive B for routines and data used in the program. It then creates an executable file named *moda.exe*, and a map file named *abc.map*.

The **pause** option in the command line causes the linker to pause and ask you to change disks before it creates the executable file. (This option is described in more detail in the section "Pausing to Change Disks" later in this chapter.)

Using a Response File to Specify Link Files

You can create a program by listing, in a response file, the names of all the files to be processed, and by giving the name of the response file on the **link** command line. The simplest way to use a response file is with a command line of the following form:

link @filename

You can also specify a response file at any prompt, or at any position in a command line. The input from the response file is treated exactly as though you had typed it at the **link** prompts or in a command line. However, any RETURN/LINEFEED combinations in the file are treated the same as if you had pressed the RETURN key in response to a prompt, or typed a comma in a command line.

When you specify a response file, remember that the filename must be the name of the response file, and that you must precede it by an "at" sign (@). If the file is in another directory or on another disk drive, you must provide a pathname.

You can name the response file anything you like. The file content has the following general form:

```
objectfiles
[executablefile]
[mapfile]
[libraryfiles]
```

You can omit any elements that have already been provided at prompts or with a partial command line.

You must place each group of filenames on a separate line. If you have more names than can fit on one line, you can simply continue the names on the next line by typing a plus sign (+) as the last character in the current line and pressing the RETURN key. If you do not supply a filename for a group, you must leave an empty line. You can give options on any line.

You can place a semicolon (;) on any line in the response file. When **link** encounters the semicolon, it automatically supplies default filenames for all files you have not yet named in the response file. The remainder of the response file is ignored.

When you create a program with a response file, the linker displays each response from your response file on the screen in the form of prompts. If the response file does not contain names for required files, **link** prompts you for the missing names and waits for you to enter responses.

The format of a response file

Note A response file should end with either a semicolon (;) or a RETURN/LINEFEED combination. If you fail to provide a final RETURN/LINEFEED in the file, the linker will display the last line of the response file and wait for you to press the RETURN key.

Example:

Using Link with a response file

The following response file tells the linker to link the four object modules, *moda*, *modb*, *modc*, and *startup*. Then, before producing the executable file *moda.exe*, it tells **link** to pause to let you swap disks. Finally, the linker creates a map file *abc.map* and searches the *math.lib* library in the *\lib* directory of drive B:

```
moda modb modc startup /PAUSE
abc
b:\lib\math
```

The following procedure combines all three methods of supplying filenames. Assume you have a response file called *library* that contains one line:

```
lib1+lib2+lib3+lib4
```

Now start **link** with a partial command line:

```
link object1 object2
```

Link takes *object1.obj* and *object2.obj* as its object files, and prompts for the next file:

```
Run File [object1.EXE]: exec
List File [NUL.MAP]:
Libraries [.LIB]: @library
```

You include the name *exec* so that the linker will name the executable file *exec.exe*. You then press the RETURN key to indicate that no map file is desired, and you enter *@library* so that the linker will read in the response file containing the four library filenames.

Using Search Paths with Libraries

You can direct **link** to search directories and disk drives for the libraries you have named in a command by either specifying one or more search paths with the library names, or by assigning the search paths to the environment variable LIB before you invoke **link**. Environment variables are explained under the **set** command in Chapter 3, "MS-DOS Commands."

A search path is the path of a directory or drive name. You type search paths along with library names on the **link** command line or in response to the Libraries prompt. You can also specify up to 16 search paths and assign them to the LIB environment variable by using the MS-DOS **set** command. In the latter case, you must separate the search paths by semicolons (;).

If you include a drive or directory name in the filename for a library in the **link** command line, the linker searches there only. If you don't give a drive or directory name, **link** searches for library files in the following order:

- 1 First, the linker searches the current drive and directory.
- 2 If the library is not found and one or more search paths have been given in the command line, the linker searches the specified search paths in the order in which you gave them.
- 3 If the library is still not found and you have set a search path by using the LIB environment variable, the linker searches there.
- 4 If the library is still not found, **link** prints an error message.

Searching for library files

Examples:

In the first example, the linker searches only the `\altlib` directory on drive A to find the `math.lib` library. To find `common.lib`, it will search the current directory on the current drive, the current directory on drive B, and finally the `\lib` directory on drive D:

```
link file,,file,A:\altlib\math.lib+common+B:+D:\lib\
```

In the second example, **link** searches the current directory, the `\lib` directory on drive C, and the `\system\lib` directory on drive U to find the libraries, `math.lib` and `common.lib`:

```
set LIB=C:\lib;U:\system\lib
link file,,file.map,math+common
```

Using Link with library files

The Map File

The map file lists the names, load addresses, and lengths of all segments in a program. It also lists the names and load addresses of any groups in the program, the program start address, and messages about any errors it may have encountered. If the **/map** option is used in the **link** command line, the map file lists the names and load addresses of all public symbols.

Segment information has the general form shown in this example:

Start	Stop	Length	Name	Class
00000H	0172CH	0172DH	TEXT	CODE
01730H	01E19H	006EAH	DATA	DATA

The Start and Stop columns show the 20-bit addresses (in hexadecimal) of the first and last byte in each segment. These addresses are relative to the beginning of the load module, which is assumed to be address 0000H. The operating system chooses its own starting address once the program is actually loaded. The Length column gives the length of the segment in bytes; the Name column gives the name of the segment; and the Class column gives the segment's class name.

Group information has the following general form:

Origin	Group
0000:0	IGROUP
0173:0	DGROUP

In this example, IGROUP is the name of the code (instruction) group and DGROUP is the name of the data group.

At the end of the listing file, the linker gives you the address of the program entry point.

If you specify the **/map** option in the **link** command line, the linker adds a public-symbol list to the map file. The symbols are presented twice: once in alphabetical order, then in the order of their load addresses. The list has the general form shown in the following example:

Address	Publics by Name
0000:1567	BRK
0000:1696	CHMOD
0000:01DB	CHKSTK
0000:131C	CLEARERR
0173:0035	FAC

Address	Publics by Value
0000:01DB	CHKSTK
0000:131C	CLEARERR
0000:1567	BRK
0000:1696	CHMOD
0000:0035	FAC

The addresses of the public symbols are in segment:offset format. They show the location of the symbol relative to the beginning of the load module, which is assumed to be at address 0000:0000.

When the **/high** and **/dsallocate** options are used and the program's code and data combined do not exceed 64K bytes, the map file may show symbols that have unusually large segment addresses. These addresses indicate a symbol whose location is below the actual start of the program code and data.

For example, the following symbol entry shows that **TEMPLATE** is located below the start of the program:

```
FFF0:0A20          TEMPLATE
```

Note that the 20-bit address of **TEMPLATE** is 00920H.

The Temporary Disk File — Vm.tmp

Link normally uses available memory for the link session. If it runs out of available memory, it creates a temporary disk file named *vm.tmp* in the current working directory. When the linker creates this file, it displays the following message:

```
VM.TMP has been created.
Do not change diskette in drive x:
```

After this message appears, you must not remove the disk from the drive specified by *x* until the link session ends. The **/pause** option cannot be used if a temporary file is created. After **link** has created the executable file, it deletes the temporary file automatically.

Note Do not use the *vm.tmp* filename for your own files, since when the linker creates the temporary file, it destroys any previous file that has the same name.

Using Link options

The Link Options

The linker options specify and control the tasks that **link** performs. All options begin with the linker-option character, which is a slash (/). You can use the following options anywhere on a **link** command line:

Option name	Action
/help	Shows the list of options
/pause	Pauses during linking
/exepack	Packs executable files
/map	Creates a public-symbol map
/linenumbers	Copies line numbers to a map file
/noignorecase	Preserves case sensitivity in names
/nodefaultlibrarysearch	Overrides default libraries
/stack	Sets maximum allocation space
/high	Sets a high load address for a program
/dsallocate	Allocates a data group
/nogroupassociation	Sets a group association override
/overlayinterrupt	Sets an overlay interrupt
/segments	Sets a maximum number of segments
/dosseg	Specifies MS-DOS segment ordering

You can abbreviate an option name as long as your abbreviation contains enough letters to distinguish the specified option from other options. Minimum abbreviations are listed for each option.

Many of the **link** options set values in the MS-DOS program header. You will understand these options better if you understand how the header is organized. The program header is described in the *MS-DOS Programmer's Reference* and in some reference books on MS-DOS.



Viewing the Options List

Syntax:

/help

The **/help** option causes **link** to write a list of the available options to the screen. If you ever need a reminder of the available options, you may find this list convenient. You should not give a filename when using the **/help** option.

Minimum abbreviation: **/he**

Example:

```
link /help
```

The /help option

Pausing to Change Disks

Syntax:

/pause

The **/pause** option causes **link** to pause before writing the executable file to disk so that you can swap disks before the linker writes the executable (.exe) file to disk.

If you specify the **/pause** switch, the linker displays the following message before creating the run file:

```
About to generate .EXE file
Change diskette in drive x: and press <ENTER>
```

Note that *x* is the proper drive name. This message appears after the linker has read data from the object files and library files, and after it has written data to the map file, if you specified one. **Link** resumes processing when you press the RETURN key, and after it writes the executable file to disk, it displays the following message:

```
Please replace original diskette
in drive letter and press <ENTER>
```

Minimum abbreviation: **/p**

The /pause option

Note Do not remove the disk used for the *vm.tmp* file, if such a file has been created. If the temporary disk message appears when you have specified the **/pause** option, you should press CONTROL-C to terminate the **link** session. Rearrange your files so that the temporary file and the executable file can be written to the same disk, then try again.

Example:

The following command causes the linker to pause just before creating the executable file *file.exe*. After creating this file, **link** pauses again to let you replace the original disk:

```
link file/pause,file,,\lib\math
```

Packing Executable Files**Syntax:**

The /exepack option**/exepack**

The **/exepack** option directs **link** to remove sequences of repeated bytes (typically nulls) and optimize the load-time relocation table before creating the executable file. Executable files linked with the **/exepack** option may be smaller, and, thus, load faster than files linked without the option. However, the Microsoft Symbolic Debug Utility (**symdeb**) cannot be used with packed files.

The **/exepack** option does not always save a significant amount of disk space (in some cases it may even increase file size). Programs that have a large number of load-time relocations (about 500 or more) and long streams of repeated characters will usually be shorter if packed. If you are not sure if your program meets these conditions, try linking it both ways and compare the results.

Minimum abbreviation: **/e**.

Example:

This example creates a packed version of the file *program.exe*:

```
link program /e;
```

Producing a Public-Symbol Map**Syntax:**

The /map option**/map**

The **/map** option causes **link** to produce a listing of all public symbols declared in your program. This list is copied to the map file that **link** creates. For a complete description of the listing-file format, see the section, "The Map File," earlier in this chapter. The **/map** option is required if you want to use **symdeb** for symbolic debugging.

Minimum abbreviation: **/m**

Note If you do not specify a map file in a **link** command, you can use the **/map** option to force the linker to create one. **Link** gives the forced map file the same filename as the first object file specified in the command. It also adds the default extension *.map*.

Example:

The following command creates a map of all public symbols in the file *file.obj*:

```
link file, ,/map;
```

Copying Line Numbers to the Map File

Syntax:

/linenumbers

The **/linenumbers** option directs the linker to copy the starting address of each program source line to a map file. The starting address is actually the address of the first instruction that corresponds to the source line. You can use the **mapsym** program to copy line-number data to a symbol file, which can then be used by **symdeb**.

The linker copies the line-number data only if you give a map-file name in the **link** command line, and only if the given object file has line-number information. Line numbering is available in some high-level-language compilers, including Microsoft FORTRAN and Pascal, versions 3.0 and later, and Microsoft C, versions 2.0 and later.

MASM does not copy line-number information to the object file. If an object file has no line-number information, the linker ignores the **/linenumbers** option.

Minimum abbreviation: **/li**

The **/linenumbers** option

Note If you do not specify a map file in a **link** command, you can still use the **/linenumbers** option to force the linker to create one. Just place the option at or before the List File prompt. **Link** gives the forced map file the same filename as the first object file that you specified in the command, and gives it the default extension *.map*.

Example:

This example causes the line-number information in the object file *file.obj* to be copied to the map file *file.map*:

```
link file/linenumbers,,em+slibfp
```

Preserving Lowercase**Syntax:**

/noignorecase

The **/noignorecase** option directs **link** to treat uppercase and lowercase letters in symbol names as distinct letters. Normally, **link** considers uppercase and lowercase letters to be identical, treating the words "TWO", "two", and "Two" as the same symbol. When you use the **/noignorecase** option, however, the linker treats "TWO", "two", and "Two" as different symbols.

Typically, you use the **/noignorecase** option with object files created by high-level-language compilers. Some compilers treat uppercase and lowercase letters as distinct letters and assume the linker does the same.

If you are linking modules created with MASM to modules created with a case-sensitive language such as C, make sure public symbols have the same sensitivity in both modules. For example, you could make all variables in C distinctive by spelling, regardless of case, and then link without the **/noignorecase** option. Another alternative would be to use the **/ML** or **/MX** option to make public variables in MASM case-sensitive. Then link with the **/noignorecase** option.

Minimum abbreviation: **/noi**

Example:

The following command causes the linker to treat uppercase and lowercase letters in symbol names as distinct letters. The object file *file.obj* is linked with routines from the standard C language library *\Slbc.lib* located in the *\lib* directory. The C language expects uppercase and lowercase letters to be treated distinctly:

```
link file1+file2/noi,,,em+mllibfp
```

The /noignorecase option

Ignoring Default Libraries

Syntax:

/nodefaultlibrarysearch

The **/nodefaultlibrarysearch** option directs the linker to ignore any library names it may find in an object file. A high-level-language compiler may add a library name to an object file to ensure that a default set of libraries is linked with the program. Using this option overrides these default libraries and lets you explicitly name the libraries you want by including them on the **link** command line.

Minimum abbreviation: **/nod**

Example:

The following example links the object files, *startup.obj* and *file.obj*, with routines from the libraries, *em*, *slibfp*, and *slibc*. Any default libraries that may have been named in *startup.obj* or *file.obj* are ignored:

```
link startup+file/nod,,,em+slibfp+slibc
```

Setting the Stack Size

Syntax:

/stack:size

The **/stack** option sets the program stack to the number of bytes given by size. The linker usually calculates a program's stack size automatically, basing it on the size of any stack segments given in the object files. If you do use the **/stack** option, the linker uses the value you type in place of any value it may have calculated.

The size can be any positive integer in the range from 1 to 65535. This value can be a decimal, octal, or hexadecimal number. Octal numbers must begin with a zero, and hexadecimal numbers must begin with a leading zero followed by a lowercase x. For example, 0x1B.

By using the **exemod** utility, you can also change the stack size after linking.

Minimum abbreviation: **/st**

The /nodefault-librarysearch option

The /stack option

Examples:

The first example sets the stack size to 512 bytes:

```
link file/stack:512,,;
```

The second example sets the stack size to 255 (FFH) bytes:

```
link moda+modb,run/st:0xFF,ab,\lib\start;
```

The final example sets the stack size to 24 (30 octal) bytes:

```
link startup+file/st:030,,;
```

Setting the Maximum Allocation Space**Syntax:**

/cparmaxalloc:*number*

The /cparmaxalloc option

The **/cparmaxalloc** option sets the maximum number of 16-byte paragraphs needed by a program when it is loaded into memory. The operating system uses this number when allocating space for a program prior to loading it.

Link normally sets the maximum number of paragraphs to 65535. Since this represents all addressable memory, the operating system always denies the default settings and allocates the largest contiguous block of memory it can find. If you use the **/cparmaxalloc** option, the operating system allocates no more space than is given by this option. This means any additional space in memory is free for other programs.

The *number* can be any integer in the range from 1 to 65535. It must be a decimal, octal, or hexadecimal number. Octal numbers must begin with a zero, and hexadecimal values must begin with a leading zero followed by a lowercase x. For example, 0x2B.

If *number* is less than the minimum number of paragraphs needed by the program, **link** ignores your request and sets the maximum value equal to the minimum needed. The minimum number of paragraphs needed by a program is never less than the number of paragraphs of code and data in the program.

Minimum abbreviation: **/c**

Examples:

The first example sets the maximum allocation to 15 paragraphs:

```
link file/c:15,,;
```

The second example sets the maximum allocation to 255 (FFH) paragraphs:

```
link moda+modb,run/cparmaxalloc:0xff,ab;
```

The final example sets the maximum allocation to 24 (30 octal) paragraphs:

```
link startup+file,/c:030,;
```

Setting a High Start Address**Syntax:**

/high

The /high option

The **/high** option sets a program's starting address to the highest possible address in free memory. If you don't use the **/high** option, **link** sets the program's starting address as low as possible in memory.

Minimum abbreviation: **/h**

Example:

This example sets the starting address of the program in *file.exe* to the highest possible address in free memory:

```
link startup+file/high,,;
```

Allocating a Data Group**Syntax:**

/dsallocate

The /dsallocate option

The **/dsallocate** option directs the linker to reverse its normal processing when assigning addresses to items belonging to the group named DGROUP. Normally, **link** assigns the offset 0000H to the lowest byte in a group. If you use **/dsallocate**, **link** assigns the offset FFFFH to the highest byte in the group. The result is data that appear to be loaded as high as possible in the memory segment containing DGROUP.

Typically, you use the **/dsallocate** option with the **/high** option to take advantage of unused memory before the start of the program. The linker assumes that all free bytes in DGROUP occupy the memory preceding the program. To use the group, you must set a segment register to the start address of DGROUP.

Minimum abbreviation: **/d**

Example:

The following example directs the linker to place the program as high in memory as possible, then adjust the offsets of all data items in DGROUP so that they are loaded as high as possible within the group:

```
link startup+file/high/dsallocate,,,em+mlibfp
```

Removing Groups from a Program

Syntax:

/nogroupassociation

The **/nogroupassociation** option directs **link** to ignore group associations when assigning addresses to data and code items.

Minimum abbreviation: **/nog**

Note This option exists strictly for compatibility with older versions of FORTRAN and Pascal (Microsoft versions 3.13 or earlier, or any IBM version prior to 2.0). You should never use the **/nogroupassociation** option except to link with object files produced by those compilers, or with the run-time libraries that accompany the old compilers.

Setting the Overlay Interrupt

Syntax:

/overlayinterrupt:number

The **/overlayinterrupt** option sets the interrupt number of the overlay loading routine to *number*. This option overrides the normal overlay interrupt number (03FH).

Number can be any integer value in the range from 0 to 255. It must be a decimal, octal, or hexadecimal number. Octal numbers must have a leading zero, and hexadecimal numbers must start with a leading zero followed by a lowercase x. For example, 0x3B.

**The /nogroup-
association option**

**The /overlay-
interrupt option**

MASM does not have an overlay manager. Therefore, you can use this option only if you are linking with a run-time module from a language compiler that supports overlays. Check your compiler documentation, since you may not be able to use this option with some compilers.

Minimum abbreviation: `/o`

Note You should not use interrupt numbers that conflict with the standard MS-DOS interrupts.

Examples:

The first example sets the overlay interrupt number to 255:

```
link file/o:255,,,87+slibfp
```

The second example sets the overlay interrupt number to 255 (FFH):

```
link moda+modb, run/overlay:0xff,ab.map,em+mlibfp
```

The final example sets the overlay interrupt number to 255 (377 octal):

```
link startup+file,/o:0377,,em+mlibfp
```

Setting the Maximum Number of Segments

Syntax:

`/segments:number`

The `/segments` option directs the linker to process no more than *number* segments per program. If it encounters more than the given limit, the linker displays an error message, and stops linking. You use this option to override the default limit of 128 segments.

If you do not use `/segments`, the linker allocates enough memory space to process up to 128 segments. If your program has more than 128 segments, you will need to set the segment limit higher to increase the number of segments that **link** can process. If you get the following **link** error message, you should set the segment limit lower:

```
Segment limit set too high
```

The `/segments` option

The *number* can be any integer value in the range from 1 to 1024. It must be a decimal, octal, or hexadecimal number. Octal numbers must have a leading zero, and hexadecimal numbers must start with a leading zero followed by a lowercase x. For example, 0x4B.

Minimum abbreviation: `/se`

Examples:

The first example sets the segment limit to 192:

```
link file/se:192,,;
```

The second example sets the segment limit to 255 (FFH).

```
link moda+modb,run/segments:0xff,ab,em+mlibfp;
```

Using DOS Segment Order

Syntax:

`/dosseg`

The `/dosseg` option causes **link** to arrange all segments in the executable file according to the MS-DOS segment-ordering convention. This convention has the following rules:

- All segments having the class name CODE are placed at the beginning of the executable file.
- Any other segments that do not belong to the group, DGROUP, are placed immediately after the CODE segments.
- All segments belonging to DGROUP are placed at the end of the file.

If you do not use the `/dosseg` option, see the section, "Order of Segments," later in this chapter, for an explanation of the normal segment order.

Minimum abbreviation: `/do`

Example:

The following command causes the linker to create an executable file, named *file.exe*, whose segments are arranged according to the MS-DOS segment-ordering convention. The segments in the object files *start.obj* and *test.obj*, and any segments copied from the libraries *math.lib* and *common.lib*, are arranged according to the same segment-ordering convention as above.

```
link start+test/dosseg,,math+common
```

The `/dosseg` option

How Link Works

Link creates an executable file by concatenating a program's code and data segments according to the instructions in the original source files. These concatenated segments form an *executable image* that is copied directly into memory when you run the program. The order and manner in which the linker copies segments to the executable file defines the order and manner in which it loads the segments into memory.

You can tell the linker how to link a program's segments by using a **SEGMENT** directive to supply segment attributes, or by using the **GROUP** directive to form segment groups. These directives define group associations, classes, and align and combine types that define the order and relative starting addresses of all segments in a program. This information works in addition to any information you supply through command line options.

The following sections explain the process that **link** uses to concatenate segments and resolve references to items in memory.

Alignment of Segments

The linker uses a segment's align type to set the starting address for the segment. The align types are *byte*, *word*, *para*, and *page*. These types correspond to starting addresses at byte, word, paragraph, and page boundaries, representing addresses that are multiples of 1, 2, 16, and 256, respectively. The default align type is *para*.

When the linker encounters a segment, it checks the align type before copying the segment to the executable file. If the align type is *word*, *para*, or *page*, the linker checks the executable image to see if the last byte copied ends at an appropriate boundary. If it doesn't, **link** pads the image with extra null bytes.

Frame Number

The linker computes a starting address for each segment in a program. The starting address is based on a segment's align type and on the size of the segments already copied to the executable file. The address consists of an offset and a canonical frame number, which specifies the address of the first paragraph in memory that contains one or more bytes of the segment. A frame number is always a multiple of 16 (a paragraph address), and the offset is the number of bytes from the start of the paragraph to the first byte in the segment. For *byte* and *word* align types, the offset may be nonzero, but the offset is always zero for *para* and *page* align types.

How link aligns segments

Segment starting addresses

The frame number of a segment can be obtained from a **link** file. The frame number is the first five hexadecimal digits of the start address specified for the segment.

Order of Segments

Link copies segments to the executable file in the same order that it encounters them in the object files. The linker maintains this order throughout the program unless it encounters two or more segments with the same class name. Segments with identical class names belong to the same class type, and are copied to the executable file as contiguous blocks.



The *Microsoft Macro Assembler Reference Manual* includes a more detailed discussion of segment loading order and methods of controlling loading order by assigning class types.

Combined Segments

Link uses combine types to determine whether two or more segments sharing the same name should be combined into a single large segment. The combine types are *public*, *stack*, *common*, *memory*, *at*, and *private*. Combine types are also described in the *Microsoft Macro Assembler Reference Manual*.

Public combine types

If a segment has a *public* combine type, the linker automatically combines it with any other segments that have the same name and belong to the same class. When **link** combines segments, it ensures that the segments are contiguous and that all addresses in the segments can be accessed using an offset from the same frame address. The result is the same as if the segment were defined as a whole in the source file.

The linker preserves each segment's align type. This means that even though the segments belong to a single, large segment, the code and data in the segments retain their original align type. If the combined segments exceed 64K bytes, **link** displays an error message.

Stack combine types

If a segment has a *stack* combine type, the linker carries out the same combine operation as for public segments. The only difference is that stack segments cause **link** to copy an initial stack-pointer value to the executable file. This stack-pointer value is the offset to the end of the first stack segment (or combined stack segment) that the linker encounters.

If you use the *stack* type for stack segments, you do not need to give instructions to load the segment into the SS register.

If a segment has a *common* combine type, the linker combines it automatically with any other segments that have the same name and belong to the same class. When **link** combines common segments, however, it places the start of each segment at the same address, creating a series of overlapping segments. The result is a single segment no larger than the largest of the combined segments.

The linker treats segments with *memory* combine types exactly like segments with *public* combine types. The Microsoft Macro Assembler (MASM), provides combine type memory for compatibility with linkers that support a separate combine type for memory segments.

A segment has a *private* combine type only if no explicit combine type is defined for it in the source file. **Link** does not combine private segments.

Groups

Groups permit noncontiguous segments that do not belong to the same class to be addressable relative to the same frame address. When **link** encounters a group, it adjusts all memory references to items in the group so that they are relative to the same frame address.

Segments in a group do not have to be contiguous and do not have to belong to the same class. Nor do they have to have the same combine type. The only requirement is that all segments in the group fit within 64K bytes.

Groups do not affect the order in which the segments are loaded. Unless you use class names and enter object files in the right order, there is no guarantee that the segments will be contiguous. In fact, the linker may place segments that do not belong to the group in the same 64K bytes of memory. Although **link** does not explicitly check whether all segments in a group fit within this 64K of memory, the linker is likely to encounter a *fixup-overflow* error if this requirement is not met.

Groups, and how to define them, are discussed further in the *Microsoft Macro Assembler Reference Manual*.

Fixups

Once the starting address of each segment in a program is known, and all segment combinations and groups have been established, the linker can fix up any unresolved references to labels and variables. To fix up unresolved references, the linker computes an appropriate offset and segment address and replaces the temporary values, generated by the assembler, with the new values.

Common combine types

Memory combine types

Private combine types



Link carries out fixups for four different references:

- Short
- Near self-relative
- Near segment-relative
- Long

The size of the value to be computed depends on the type of reference. If **link** discovers an error in the anticipated size of a reference, it displays a *fixup-overflow* message. This error can happen, for example, if a program attempts, by using a 16-bit offset, to reach an instruction in a segment that has a different frame address. The error can also occur if the segments in a group do not fit within a single 64K-byte block of memory.

Short references

A short reference occurs in JMP instructions that attempt to pass control to labeled instructions in the same segment or group. The target instruction must be no more than 128 bytes from the point of reference. The linker computes a signed, 8-bit number for this reference and displays an error message if the target instruction belongs to a different segment or group (that is, if it has a different frame address), or if the target is more than 128 bytes distant (in either direction).

Near self-relative references

A near self-relative reference occurs in instructions which access data relative to the same segment or group. The linker computes a 16-bit offset for this reference and displays an error message if the data are not in the same segment or group.

Near segment-relative references

A near segment-relative reference occurs in instructions that attempt to access data in a specified segment or group, or that is relative to a specified segment register. **Link** computes a 16-bit offset for this reference and displays an error message if either of the following conditions exists: the offset of the target within the specified frame is greater than 64K bytes or less than 0, or the beginning of the canonical frame of the target is not addressable.

Long references

A long reference occurs in CALL instructions that attempt to access an instruction in another segment or group. **Link** computes a 16-bit frame address and 16-bit offset for this reference and displays an error message if either of the following conditions exists: the computed offset is greater than 64K bytes or less than 0, or the beginning of the canonical frame of the target is not addressable.

10 Debug

In this chapter you'll learn:

- How to start the **debug** utility
- How to use the **debug** commands and parameters

Introduction

The **debug** utility is a debugging program that provides a controlled testing environment for binary and executable object files. Note that **edlin**, the MS-DOS line editor, is used to alter source files; **debug** is **edlin**'s counterpart for binary files.

Debug eliminates the need to reassemble a program to see if a problem has been fixed by a minor change. It allows you to alter the contents of a file or the contents of a CPU register, and then immediately reexecute a program to check the validity of the changes made.

All **debug** commands may be aborted at any time by pressing CONTROL-C. The CONTROL-S key sequence suspends the display, so that you can read it before the output scrolls away. Pressing any key other than CONTROL-C or CONTROL-S restarts the display. All these commands are consistent with the control character functions available at the MS-DOS command level.

Starting Debug

How to Start Debug

Debug may be started two ways. By the first method, you type all commands in response to the **debug** prompt (a hyphen). By the second method, you type all commands on the line used to start **debug**.

Method 1: **debug**

Method 2: **debug [filename [arglist]]**

Method 1

Method 1: Debug

To start **debug** using method 1, simply type the following:

```
debug
```

Debug responds with the hyphen (-) prompt, signaling that it is ready to accept your commands. Since you didn't specify a filename, you can use other commands to work on current memory, disk sectors, or disk files.

Warnings

- When **debug** (version 2.0) is started, it sets up a program header at offset 0 in the program work area. In previous versions of **debug**, you could overwrite this header. You can still overwrite the default header if you don't give a *filename* to **debug**. If you are debugging a *.com* or *.exe* file, however, do not tamper with the program header below address 5CH, or **debug** will terminate.
- Do not restart a program after the following message is displayed:

```
Program terminated normally
```

You must reload the program with the **N** (name) and **L** (load) commands for it to run properly.

Method 2: Command Line

Method 2

To start **debug** using a command line, you must use the following syntax:

debug [*filename* [*arglist*]]

For example, if you specify a *filename*, the following would be a typical command to start **debug**:

```
debug file.exe
```

Debug would then load *file.exe* into memory starting at 100 (hexadecimal) in the lowest available segment. The **BX:CX** registers are loaded with the number of bytes placed into memory.

If you do include a *filename*, you might also specify an *arglist*. An *arglist* is a list of filename parameters and switches that are to be passed to the program *filename*. So when *filename* is loaded into memory, it is loaded as if it had been started with a command of the form, **debug filename arglist**.

Here, *filename* is the file to be debugged, and *arglist* is the rest of the command line used when **debug** calls and loads *filename* into memory.

Debug Command Information

Each **debug** command consists of a single letter followed by one or more parameters. Additionally, the control characters and special editing functions described in Chapter 5, "MS-DOS Editing and Function Keys," apply to **debug** as well.

If a syntax error occurs in a **debug** command, **debug** reprints the command line and indicates the error with a caret (^) and the word "Error" as in the following example:

```
dcs:100 cs:110
  ^ Error
```

Note that when typing commands and parameters you may use any combination of uppercase and lowercase letters.

The **debug** commands are listed below. Following this list, the commands and their parameters are described in greater detail.

Debug command	Function
A [<i>address</i>]	Assemble
C <i>range address</i>	Compare
D [<i>range</i>]	Dump
E <i>address</i> [<i>list</i>]	Enter
F <i>range list</i>	Fill
G [= <i>address</i> [<i>address...</i>]]	Go
H <i>value value</i>	Hex
I <i>value</i>	Input
L [<i>address</i> [<i>drive:record record</i>]]	Load
M <i>range address</i>	Move
N <i>filename</i> [<i>filename</i>]	Name
O <i>value byte</i>	Output
Q	Quit
R [<i>register-name</i>]	Register
S <i>range list</i>	Search
T [= <i>address</i>] [<i>value</i>]	Trace
U [<i>range</i>]	Unassemble
W [<i>address</i> [<i>drive:record record</i>]]	Write

How to use the Debug command parameters

Debug Command Parameters

All **debug** commands accept parameters, except the **Q** (quit) command. Parameters may be separated by delimiters (spaces or commas), but a delimiter is required only between two consecutive hexadecimal values. Thus, the following commands are equivalent:

```
dc s:100 110
d cs:100 110
d ,cs:100,110
```

Parameter	Definition
<i>drive:</i>	A one-digit hexadecimal value that indicates which drive a file will be loaded from or written to. The valid values are 0–3, where 0 = A:, 1 = B:, 2 = C:, 3 = D:.
<i>byte</i>	A two-digit hexadecimal value placed in or read from an address or register.
<i>record</i>	1-digit to 3-digit hexadecimal value that indicates the logical record number on the disk and the number of disk sectors to be written or loaded. Logical records correspond to sectors; however, since they represent the entire disk space, their numbering differs.
<i>value</i>	A hexadecimal value of up to four digits that specifies a port number or the number of times a command should repeat its functions.
<i>address</i>	A two-part designation containing either an alphabetic segment register or a four-digit segment address plus an offset value. You may omit the segment name or segment address, in which case the default segment DS is used for all commands except G, L, T, U, and W, for which the default segment is CS. All numeric values are hexadecimal.

Following is an example *address*:

```
CS:0100
04BA:0100
```

Note that the colon is required between the segment name (whether numeric or alphabetic) and the offset value.

<i>range</i>	Contains two addresses: for example, <i>address address</i> ; or one address, an L, and a value: for example, <i>address L value</i> where <i>value</i> is the number of lines on which the command should operate (L80 is assumed). The second type of <i>range</i> cannot be used if another hexadecimal value follows, since the hexadecimal value would be interpreted as the second <i>address</i> of the <i>range</i> .
--------------	---

Here are some example ranges:

```
CS:100 110
CS:100 L 10
CS:100
```

The following example, however, is illegal:

```
CS:100 CS:110
      b Error
```

The limit for *range* is 10000 (hexadecimal). To specify a *value* of 10000 with only four digits, type 0000 (or 0).

list A series of *byte* values or *strings*. *List* must be the last parameter on the command line.

Following is an example *list*:

```
fcs:100 42 45 52 54 41
```

string Any number of characters enclosed in quotation marks. The quotation marks may be either single (' ') or double (" "). If the delimiter marks must appear within a *string*, you must use the double quotation marks.

For example, the following strings are legal:

```
"This 'string' is okay."
"This ""string"" is okay."
```

However, this string is illegal:

```
"This "string" is not okay."
```

Note that the double quotation marks are not necessary in the following strings:

```
"This 'string' is not necessary."
'This ""string"" is not necessary.'
```

The ASCII values of the characters in the string are used as a *list* of byte values.

Assemble

(A)ssemble

Purpose:

Assembles 8086/8087/8088 mnemonics directly into memory.

Syntax:

A[address]

Comments:

If a syntax error is found, **debug** responds with the following message, then redisplay the current assembly address:

^Error

All numeric values are hexadecimal and you must type them as 1–4 characters. Also, you must specify prefix mnemonics *in front of* the opcode to which they refer. You may type them on a separate line, however.

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, and use **MOVSB** to move byte strings.

The assembler will automatically assemble short, near, or far jumps and calls, depending on byte displacement, to the destination address. You may override these jumps and calls by using a **NEAR** or **FAR** prefix, as in the following example:

```
0100:0500 JMP 502 ; a 2-byte short jump
0100:0502 JMP NEAR 505 ; a 3-byte near jump
0100:505 JMP FAR 50A ; a 5-byte far jump
```

You may abbreviate the **NEAR** prefix to **NE**, but the **FAR** prefix cannot be abbreviated.

Debug cannot tell whether some operands refer to a word memory location or to a byte memory location. In this case, the data type must be explicitly stated with the prefix, **WORD PTR** or **BYTE PTR**. Acceptable abbreviations are **WO** and **BY**. For example:

```
NEG BYTE PTR [128]
DEC WO [SI]
```

Debug also cannot tell whether an operand refers to a memory location or to an immediate operand. So, it uses the common convention that operands enclosed in square brackets refer to memory. For example:

```
MOV     AX,21           ; Load AX with 21H
MOV     AX,[21]         ; Load AX with the
                        ; contents
                        ; of memory location 21H
```

Two popular pseudo-instructions are available with the **A** (assemble) 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:

```
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"
```

The **A** command supports all forms of register indirect commands. For example:

```
ADD     BX,34[BX+2].[SI-1]
POP     [BP+DI]
PUSH    [SI]
```

All opcode synonyms are also supported, as in the next example:

```
LOOPZ   100
LOOPE   100

JA      200
JNBE    200
```

For 8087 opcodes, the **WAIT** or **FWAIT** prefixes must be explicitly specified, as in these last examples:

```
FWAIT FADD ST,ST(3) ; This line assembles
                    ; an FWAIT prefix
LD TBYTE PTR [BX]   ; This line does not
```

Compare

(C)ompare

Purpose:

Compares the portion of memory specified by *range* to a portion of the same size beginning at the specified *address*.

Syntax:

Crange address

Comments:

If the two areas of memory are identical, there is no display, and **debug** returns with the MS-DOS prompt. If there *are* differences, they are displayed in this format:

address1 byte1 byte2 address2

Example:

The following commands have the same effect:

C100,1FF 300

or

C100L100 300

Each command compares the block of memory from 100 to 1FFH with the block of memory from 300 to 3FFH.

**Using the Compare
command**

(D)ump

Dump**Purpose:**

Displays the contents of the specified *range* of memory.

Syntax:

D[*range*]

Comments:

If you specify a *range* of addresses with the **D** (dump) command, the contents of the *range* are displayed. If you don't use parameters with the **D** command, 128 bytes are displayed at the first address (**DS:100**) after the address displayed by the previous **D** command.

The dump is displayed in two portions: a hexadecimal dump (each byte is shown in hexadecimal value) and an ASCII dump (the bytes are shown in ASCII characters). Nonprinting characters are denoted by a period (.) in the ASCII portion of the display. Each display line shows 16 bytes, with a hyphen between the eighth and ninth bytes. At times, displays are split in this chapter to fit them on the page. Each displayed line begins on a 16-byte boundary.

Examples:

If you type the command:

```
dc5:100 110
```

debug displays the dump in the following format:

```
04BA:0100 42 45 52 54 41 ... 4E 44 TOM SAWYER
```

If you simply type the **D** command, the display is formatted as described above. Each line of the display begins with an address, incremented by 16 from the address on the previous line.

Each subsequent **D** (typed without parameters) displays the bytes immediately following those last displayed.

**Using the Dump
command**

If you type the following command, the display is formatted as described above, but 20H bytes are displayed:

```
DCS:100 L 20
```

If you then type the following command, the display is formatted as described above, but all the bytes in the range of lines from 100H to 115H in the CS segment are displayed:

```
DCS:100 115
```

(E)nter

Enter**Purpose:**

Enters byte values into memory at the specified *address*.

Syntax:

Eaddress[*list*]

Comments:

If, when using the **E** (enter) command, you type the optional *list* of values, the byte values are replaced automatically. (If an error occurs, no byte values are changed.)

If you type the *address* without the optional *list*, **debug** displays the *address* and its contents, repeats the *address* on the next line, and then waits for your input. At this point, the **E** (enter) command waits for you to perform one of the following actions:

- Replace a byte value with a value you type. Simply type the value after the current value. If the one you type is not a legal hexadecimal value or if it contains more than two digits, the illegal or extra character is not echoed.
- Press the SPACEBAR to advance to the next byte. To change the value, simply type the new value as described in the previous action. If, when you press the SPACEBAR, you move beyond an 8-byte boundary, **debug** starts a new display line with the address displayed at the beginning.
- Type a hyphen (-) to return to the preceding byte. If you decide to change a byte behind the current position, typing the hyphen returns the current position to the previous byte. When you type the hyphen, a new line is started with its address and byte value displayed.
- Press the RETURN key to terminate the **E** command. The RETURN key may be pressed at any byte position.

Examples:

Suppose you type the following command:

```
ECS:100
```

Now suppose that **debug** displays the following:

```
04BA:0100 EB._
```

To change this value to, say, 41, type the number 41 at the cursor, as shown:

```
04BA:0100 EB.41_
```

To step through the subsequent bytes, you would press the SPACEBAR until you saw the following:

```
04BA:0100 EB.41 10. 00. BC._
```

To change **BC** to the number 42, for instance, you would type the number at the cursor, as follows:

```
04BA:0100 EB.41 10. 00. BC.42_
```

Notice that the value 10 should be 6F. To correct this value, you would type the hyphen as many times as needed to return to byte 0101 (value 10), then replace 10 with 6F:

```
04BA:0100 EB.41 10. 00. BC.42-
04BA:0102 00.-_
04BA:0101 10.6F_
```

Pressing the RETURN key ends the **E** command and returns you to the **debug** command level.

**Using the Enter
command**

(F)ill

Fill**Purpose:**

Fills the addresses in the specified *range* with the values in the specified *list*.

Syntax:

Range list

Comments:

If the *range* contains more bytes than the number of values in the *list*, the *list* will be used repeatedly until all bytes in the *range* are filled.

If the *list* contains more values than the number of bytes in the *range*, the extra values in the *list* are ignored. If any of the memory in the *range* is not valid (bad or nonexistent), the error will occur in all succeeding locations.

Example:

Suppose you type the following command:

```
F04BA:100 L 100 42 45 52 54 41
```

Debug would now fill memory locations 04BA:100 through 04BA:1FF with the bytes specified. The five values would then be repeated until all the 100H bytes were filled.

**Using the Fill
command**

Go

(G)o

Purpose:

Executes the program currently in memory.

Syntax:

G[= *address* [*addresses*]]

Comments:

If you type the G (go) command by itself, the program currently in memory executes as if it had run outside **debug**.

If you set =*address*, execution of the G command begins at the address specified. The equal sign (=) is required so that **debug** can distinguish the start =*address* from the breakpoint *addresses*.

With the other optional *addresses* set, execution stops at the first *address* encountered, regardless of that address' position in the list of addresses that halt execution or program branching. When program execution reaches a breakpoint, the registers, flags, and decoded instruction are displayed for the last instruction executed. (The result is the same as if you had typed the R (register) command for the breakpoint address.)

You may set up to ten breakpoints, but only at addresses containing the first byte of an 8086 opcode. If you set more than ten breakpoints, **debug** returns the BP error message.

The user stack pointer must be valid and must have 6 bytes available for this command. The G command uses an IRET instruction to cause a jump to the program under test. The user stack pointer is set, and the user flags, Code Segment register, and Instruction Pointer are pushed on the user stack. (If the user stack is not valid or is too small, the operating system may crash.) An interrupt code (0CCH) is placed at the specified breakpoint address(es).

When **debug** encounters an instruction with the breakpoint code, it restores all breakpoint addresses to their original instructions. If you don't halt execution at one of the breakpoints, the interrupt codes are not replaced with the original instructions.

Example:

Suppose you type the following command:

GCS:7550

Using the Go
command

The program currently in memory would execute up to the address 7550 in the CS segment. **Debug** would then display registers and flags, after which the **G** command would terminate.

After **debug** has encountered a breakpoint, if you type the **G** command again the program runs as if you had typed the filename at the MS-DOS command level. The only difference is that program execution begins at the instruction after the breakpoint, rather than at the usual start address.

Hex

(H)ex

Purpose:

Performs hexadecimal arithmetic on the two specified parameters.

Syntax:

Hvalue value

Comments:

First, **debug** adds the two parameters, then subtracts the second parameter from the first. The results of these actions are displayed on one line — first the sum, then the difference.

Example:

Suppose you type the following command:

H19F 10A

In response, **debug** would perform the calculations and then display the following result:

02A9 0095

**Using the Hex
command**

(I)input

Input**Purpose:**

Inputs and displays one byte from the port specified by *value*.

Syntax:

Ivalue

Comment:

A 16-bit port address is allowed.

Example:

Suppose you type the following command:

I2F8

Suppose also that the byte at the port is 42H. **Debug** would input the byte and then display the following:

42

**Using the Input
command**

Load

(L)oad

Purpose:

Loads a file into memory.

Syntax:

L[*address* [*drive: record record*]]

Comments:

Set **BX:CX** to the number of bytes read. The file must have been named either when you started **debug** or with the **N** (name) command. Both the **debug** invocation and the **N** command format a filename properly in the normal format of a File Control Block at **CS:5C**.

If you use the **L** (load) command without any parameters, **debug** loads the file into memory beginning at address **CS:100** and sets **BX:CX** to the number of bytes loaded. If you type the **L** command with an address parameter, loading begins at the memory location specified by the *address*. If you use the **L** command with all parameters included, absolute disk sectors are loaded, instead of a file.

Each *record* is taken from the specified *drive*: (the drive designation is numeric here: 0=A:, 1=B:, 2=C:, etc.). **Debug** begins loading with the first specified *record*, and continues until the number of sectors in the second *record* have been loaded.

Example:

Suppose once you have started **debug** that you type the following commands:

```
-NFILE.COM
```

Now, to load *file.com*, you would simply type the **L** command.

Debug would then load the file and display the **debug** prompt. Suppose now that you want to load only portions of a file or certain records from a disk. To do this, you would type the following:

```
L04BA:100 2 0F 6D
```

Debug would then load 109 (6D hex) records, beginning with logical record number 15, into memory beginning at address **04BA:0100**. Then, once it had loaded the records, **debug** would simply return the hyphen (-) prompt.

Using the Load
command

If the file has an *.exe* extension, it would be relocated to the load address specified in the header of the *.exe* file. The *address* parameter is always ignored for *.exe* files. The header itself is stripped off the *.exe* file before it is loaded into memory. So, the size of an *.exe* file on disk will differ from its size in memory.

If the file was named by the N (name) command, or specified when you started **debug**, as a *.hex* file, then typing the L command with no parameters would cause **debug** to load the file beginning at the address specified in the *.hex* file. If the L command included the option *address*, **debug** would add the *address* specified in the L command to the address found in the *.hex* file to determine the start address at which to load the file.

Move

(M)ove

Purpose:

Moves the block of memory specified by *range* to the location beginning at the specified *address*.

Syntax:

*M**range address*

Comments:

Overlapping moves (i.e., moves where part of the block overlaps some of the current addresses) are always performed without loss of data. Addresses that could be overwritten are moved first. For moves from higher to lower addresses, the sequence of events is to first move the data beginning at the block's lowest address and then work toward the highest. For moves from lower to higher addresses, the sequence is to first move the data beginning at the block's highest address and then work toward the lowest.

Note that if the addresses in the block being moved will not have new data written to them, the data in the block *before the move* will remain. The M (move) command copies the data from one area into another, in the sequence described, and writes over the new addresses. This action is why the sequence of the move is important.

Example:

Suppose you type the following command:

```
MCS:100 110 CS:500
```

In response, **debug** would first move address **CS:110** to **CS:510**, then move **CS:10F** to **CS:50F**, and so on until **CS:100** is moved to **CS:500**. To review the results of the move, you could type the **D** command, using the same *address* as you used with the **M** command.

**Using the Move
command**

(N)ame

Name**Purpose:**

Sets filenames.

Syntax:

N*filename* [*filename*...]

Comments:

The **N** (name) command performs two functions. First, it assigns a filename for a later **L** (load) or **W** (write) command. So, if you start **debug** without naming a file to be debugged, you must type the command **Nfilename** before a file can be loaded. Second, the **N** command assigns filename parameters to the file being debugged. In this case, **N** accepts a list of parameters used by the file being debugged.

Note that these two functions overlap. Consider, for example, the following set of **debug** commands:

```
-N FILE1.EXE  
-L  
-G
```

The **N** command would use these commands to perform the following steps:

- 1 It would first assign the filename *file1.exe* to the file to be used in any later **L** or **W** commands.
- 2 It would also assign the *file1.exe* filename to the first filename parameter used by any program that is later debugged.
- 3 The **L** command would then load *file1.exe* into memory.
- 4 The **G** (go) command would cause *file1.exe* to be run with *file1.exe* as the single filename parameter (that is, *file1.exe* would be run as if *file1.exe* had been typed at the command level).

A more useful chain of commands might look like this:

```
-N FILE1.EXE  
-L  
-N FILE2.DAT FILE3.DAT  
-G
```


In this example, the **N** command sets *file1.exe* as the filename for the subsequent **L** command, which loads *file1.exe* into memory. The **N** command is then used again, this time to specify the parameters to be used by *file1.exe*. Finally, when the **G** command is run, *file1.exe* is executed as if *file1 file2.dat file3.dat* had been typed at the MS-DOS command level.

Note that if you were to execute a **W** command now, then *file1.exe* — the file being debugged — would be saved with the name *file2.dat*. To avoid this kind of result, you should always execute an **N** command before either an **L** or **W** command.

There are four regions of memory that can be affected by the **N** command:

CS:5C	FCB for file 1
CS:6C	FCB for file 2
CS:80	Count of characters
CS:81	All characters typed

The first filename parameter that you specify for the **N** command has a file control block (FCB) set up at **CS:5C**. If you name a second filename parameter, an FCB is set up for this parameter beginning at **CS:6C**. The number of characters typed in the **N** command (exclusive of the first character, **N**) is given at location **CS:80**.

The actual stream of characters given by the **N** command (again, exclusive of the letter **N**) begins at **CS:81**. Note that this stream of characters may contain switches and delimiters that would be legal in any command typed at the MS-DOS command level.

Example:

A typical use of the **N** command is as follows:

```
DEBUG PROG.COM
-NPARAM1 PARAM2/C
-G
-
```

In this case, the **G** command executes the file in memory as if you had typed the following command line:

```
PROG PARAM1 PARAM2/C
```

Testing and debugging therefore reflect a normal run-time environment for *prog.com*.

**Using the Name
command**

(O)utput

Output**Purpose:**

Sends the specified *byte* to the output port specified by *value*.

Syntax:

Ovalue byte

Comment:

A 16-bit port address is allowed.

Example:

Suppose you want to **debug** to output the byte value 4F to output port 2F8. To do this you could simply type the following command:

O2F8 4F

**Using the Output
command**

Quit

(Q)uit

Purpose:

Terminates the **debug** utility.

Syntax:

Q

Comments:

The **Q** (quit) command takes no parameters and exits **debug** without saving the file you're currently working with. You are returned to the MS-DOS command level.

Example:

To end the debugging session, simply type the following and press the RETURN key:

q

Debug terminates, and control returns to the MS-DOS command level.

**Using the Quit
command**

(R)egister

Register**Purpose:**

Displays the contents of one or more CPU registers.

Syntax:

R[*register-name*]

Comments:

If you do not type a *register-name*, the **R** (register) command dumps the register storage area and displays the contents of all registers and flags.

If you do type a *register-name*, the 16-bit value of that register is displayed in hexadecimal, and a colon then appears as a prompt. You can now either type a *value* to change the register, or press the RETURN key if you don't want a change.

Following is a list of the valid *register-names*:

AX	BP	SS	
BX	SI	CS	
CX	DI	IP	(IP and PC both refer
DX	DS	PC	to the Instruction
SP	ES	F	Pointer.)

Any other entry for *register-name* results in a **BR** error message.

If you type **F** as the *register-name*, **debug** displays each flag with a two-character alphabetic code. To change any flag, type the opposite two-letter code. The flags are then either set or cleared.

The flags are listed below with their codes for **SET** and **CLEAR**:

Flag name	Set	Clear
Overflow	OV	NV
Direction	DN Decrement	UP Increment
Interrupt	EI Enabled	DI Disabled
Sign	NG Negative	PL Plus
Zero	ZR	NZ
Auxiliary Carry	AC	NA
Parity	PE Even	PO Odd
Carry	CY	NC

Whenever you type the **RF** command, the flags are displayed (in the order shown in the previous table) in a row at the beginning of a line. At the end of the list of flags, **debug** displays a hyphen (-).

You may enter new flag values in any order as alphabetic pairs. You do not have to leave spaces between these values. To exit the **R** command, press the RETURN key. Any flags for which you did not specify new values remain unchanged.

If you type more than one value for a flag, **debug** returns a **DF** error message. If you enter a flag code other than one of those shown in the table above, **debug** returns a **BF** error message. In both cases, the flags up to the error in the list are changed; those flags at and after the error are not.

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.

Using the Register command

Examples:

If you type the following command, **debug** displays all registers, flags, and the decoded instruction for the current location:

R

If the location is **CS:11A**, for example, the display will look similar to this:

```
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
```

If you then type the following command, **debug** will display these flags:

```
RF
NV UP DI NG NZ AC PE NC - _
```

Now, you could type any valid flag designation, in any order, with or without spaces. For example:

```
NV UP DI NG NZ AC PE NC - PLEICY
```

In response, **debug** would display the **debug** prompt. To see the changes, type either the **R** or **RF** command:

```
RF
NV UP EI PL NZ AC PE CY - _
```

Press the RETURN key to leave the flags this way or to specify different flag values.

Search

(S)earch**Purpose:**

Searches the specified *range* for the specified *list* of bytes.

Syntax:

Strange list

Comments:

The *list* may contain one or more bytes, each separated by a space or comma. If the *list* contains more than one byte, only the first address of the byte string is returned. If the *list* contains only one byte, all addresses of the byte in the *range* are displayed.

Example:

Suppose you type the following command:

```
SCS:100 110 41
```

Debug would display a response similar to this:

```
04BA:0104  
04BA:010D  
-type:
```

**Using the Search
command**

(T)race

Trace**Purpose:**

Executes one instruction and displays the contents of all registers, flags, and the decoded instruction.

Syntax:

T[= *address*] [*value*]

Comments:

If you include the = *address* option in the **T** (trace) command, tracing occurs at the specified = *address*. The *value* option causes **debug** to execute and trace the number of steps specified by *value*.

The **T** command uses the hardware trace mode of the 8086 or 8088 microprocessor. Consequently, you may also trace instructions stored in ROM (Read Only Memory).

Example:

Suppose you type the following command:

T

In response, **debug** would return a display of the registers, flags, and decoded instruction for that one instruction. Assuming, for this example, that the current position is 04BA:011A, **debug** might return the following display:

```
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
```

If you type the following command, **debug** executes sixteen (10 hex) instructions beginning at 011A in the current segment, and then displays all registers and flags for each instruction as it is executed. The display scrolls away until the last instruction is executed, and then stops. Now you can see the register and flag values for the last few instructions performed:

T=011A 10

Remember that if you want to study the registers and flags for any instruction (at any time), you can press **CONTROLS** to stop the display from scrolling.

Using the Trace command

Unassemble

(U)nassemble

Purpose:

Disassembles bytes and displays the source statements that correspond to them, with addresses and byte values.

Syntax:

U[*range*]

Comments:

The display of disassembled code looks like a listing for an assembled file. If you type the U (unassemble) command without parameters, 20 hexadecimal bytes are disassembled at the first address after that displayed by the previous U command. If you type the U command including the *range* parameter, then **debug** disassembles all bytes in *range*. But if you specify *range* only as an *address*, then 20H bytes are disassembled.

Example:

Suppose you type the following command:

U04BA:100 L10

In response, **debug** would disassemble 16 bytes, beginning at address 04BA:0100:

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

Using the
Unassemble
command

Now, suppose you type the following:

```
U04ba:0100 0108
```

The display would now show:

```
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
```

If the bytes in some addresses are altered, the disassembler alters the instruction statements. You can then type the U command for the changed locations, the new instructions viewed, and the disassembled code used to edit the source file.

Write

(W)rite

Purpose:

Writes the file being debugged to a disk file.

Syntax:

W[*address* [*drive: record record*]]

Comments:

If you do not use parameters with the **W** (write) command, **BX:CX** must already be set to the number of bytes to be written; the file is written beginning from **CS:100**. If you type the **W** command with just an *address*, then the file is written beginning at that *address*. If you have used a **G** (go) or **T** (trace) command, you must reset **BX:CX** before using the **W** command without parameters.

Note that if a file is loaded and modified, the name, length, and starting address are all set correctly to save the modified file (as long as the length has not changed).

You must have named the file either with the initial **debug** startup command or with the **N** (name) command (refer to the **N** command earlier in this manual). Both the **debug** startup command and the **N** command properly format a filename in the normal format of a File Control Block at **CS:5C**.

If you include parameters when you use the **W** command, the write begins from the memory address specified; the file is written to the specified *drive*: (the drive name is numeric here — 0 = A:, 1 = B:, 2 = C:, etc.). **Debug** writes the file beginning at the logical record number specified by the first *record*. **Debug** then continues to write the file until the number of sectors specified in the second *record* have been written.

Warning Writing to absolute sectors is *extremely* risky because the process bypasses the file handler.

Examples:

If you type the following command, **debug** will write the file to disk and then display the **debug** prompt:

W

**Using the Write
command**

Two examples are shown below.

W

- _

WCS:100 1 37 2B

Debug writes out the contents of memory to the disk in drive B:, beginning with the address **CS:100**. The data written out starts in the disk logical record number 37H and consists of 2BH records. When the write is complete, **debug** displays the following prompt:

WCS:100 1 37 2B

- _

Debug Error Messages

Debug error codes

During a **debug** session, you may receive any of the following error messages. Each error ends the **debug** command under which it occurred, but does not end **debug** itself.

Error code	Definition
BF	Bad flag You attempted to change a flag, but the characters you typed were not one of the acceptable pairs of flag values. See the R (register) command for the list of acceptable flag entries.
BP	Too many breakpoints You specified more than ten breakpoints as parameters to the G (go) command. Retype the G command using ten or fewer breakpoints.
BR	Bad register While using the R command you typed an invalid register name. See the R command for the list of valid register names.
DF	Double flag You typed two values for one flag. You may specify a flag value only once per RF command.



Appendix A

Instructions for Users with Single Floppy Disk Drive Systems

If you have only one floppy disk drive, you can still type MS-DOS commands just as if you had two disk drives on your system.

Just think of your one-drive system as having *two* drives (drive A and drive B). But instead of A and B representing two physical drives, they represent disks.

Just remember that when you specify drive B, when the drive A disk was used last, MS-DOS prompts you to insert the disk for drive B. For example:

```
A> copy command.com b:
Insert diskette for drive B:
and strike any key when ready
  1 File(s) copied
A>_
```

If you specify drive A when the drive B disk was used last, MS-DOS asks you to change disks again, this time prompting you to insert the drive A disk.

When using a batch file to execute commands, you follow the same procedure. MS-DOS waits for you to insert the appropriate disk and press any key before it continues.

Important The letter displayed in the system prompt represents the default drive where MS-DOS looks to find a file whose name is entered without a drive name; this letter does *not* represent the last disk used.

Assume, for example, that A is the default drive. If the last command performed was **dir b:**, MS-DOS would act as if the drive B disk is still in the drive. The system prompt, however, is still A>, because A is still the default drive. If you type the **dir** command, MS-DOS prompts you for the drive A disk, because it is the default drive and you did not specify another drive in the **dir** command.



Appendix B

How to Configure Your System

What is a Configuration File?

The configuration file *config.sys* is a file that contains certain commands, which MS-DOS checks at startup. Each time you start MS-DOS, it searches the root directory of the drive in which it was started for a file named *config.sys*.

The *config.sys* file allows you to configure your system with a minimum of effort. For example, you can add device drivers to your system by including special commands in your *config.sys* file.

If your MS-DOS disk does not have a *config.sys* file, you can use the MS-DOS line editor, **edlin**, to create one and then save it on the MS-DOS disk in your root directory. If *config.sys* already exists and you want to change it, you can use **edlin** to edit it.

Config.sys Commands

The following commands are used in the *config.sys* file:

break	Sets CONTROL-C check.
buffers	Sets the number of sector buffers.
country	Allows for international time, date, and currency.
device	Installs the device driver in the system.
drivparm	Defines parameters for block devices.
fcbs	Specifies the number of FCBs that can be open concurrently.
files	Sets the number of open files that can access certain MS-DOS system calls.
shell	Begins execution of the shell from a specific file (usually <i>command.com</i>).

These commands are described in detail in the following pages. For a sample *config.sys* file, see the section, "Sample Config.sys File," at the end of this appendix.

Break

Purpose:

Sets CONTROL-C check.

Syntax:

break=[ON]

or

break=[OFF]

Comments:

Depending on the program you are running, you may use CONTROL-C to stop an activity (for example, to stop sorting a file). Normally, MS-DOS checks to see whether you have pressed CONTROL-C only while it is reading from the keyboard or writing to the screen or printer. Therefore, setting **break** to ON extends CONTROL-C checking to other functions, such as disk reading and writing.

Example:

To turn off CONTROL-C checking, put the following line in your *config.sys* file:

```
break=off
```

Break

Setting and canceling CONTROL-C checking

Buffers

Buffers**Purpose:**

Allows you to set the number of disk buffers that MS-DOS allocates in memory at the time you start the system.

Syntax:

buffers=*x*

Comments:

The *x* option is the number of disk buffers, from 2 to 255. A disk buffer is a block of memory that MS-DOS uses to hold data when reading or writing.

The default number of buffers is 2. But for applications such as word processors, a number between 10 and 20 provides the best performance. If you plan to create a lot of subdirectories, you may even want to increase the buffers value to between 20 and 30. Remember, though, that buffers take up space in memory, so you should not increase their number to a value greater than 30.

Example:

To create ten disk buffers, put the following line in your *config.sys* file:

```
buffer s=10
```

**Setting the number
of disk buffers**

Country

Purpose:

Country allows MS-DOS to use international time, date, currency, and case conversions.

Syntax:

country=*x*

Comments:

The *x* option is a number representing a specific country, and is set by the equipment manufacturer. Values for *x* range from 001 to 999. The following table shows the possible values for *x*:

Value	Country
001	United States
031	Belgium
033	France
034	Spain
039	Italy
041	Switzerland
044	United Kingdom
045	Denmark
046	Sweden
047	Norway
049	Germany
061	Australia
358	Finland
972	Israel

Example:

The following example sets the **country** to France (=033) and converts international currency, time, date, and case to French conventions:

```
country=033
```

Country

Setting Country conversions

Device

Device**Purpose:**

Installs the specified device driver on the system list.

Syntax:

device = [*drive*:]*pathname*

Comments:

The standard installable device drivers provided with MS-DOS are *ansi.sys*, *driver.sys*, and *ramdrive.sys*. For more information on these installable device drivers, refer to Appendix C, "Installable Device Drivers."

Example:

If you plan to use the ANSI escape sequences described in Appendix C, you should create a *config.sys* file with the following command:

```
device=ansi.sys
```

This command causes MS-DOS to replace all keyboard input and screen output support with the ANSI escape sequences.

Drivparm

Drivparm

Purpose:

This command allows you to define parameters for block devices when you start MS-DOS, overriding the original MS-DOS device driver settings.

Syntax:

drivparm = /d:*dd* [/c] [/f:*ff*] [/h:*hb*] [/n] [/s:*ss*] [/t:*ttt*]

Comments:

Setting **drivparm** overrides any previous block device driver definitions.

The *dd* parameter for the /d: switch specifies a logical drive number between 0 and 255. This means that drive number 0 = A, 1 = B, 2 = C, etc.

The /c switch specifies that change-line (doorlock) support is required.

The *ff* option on the /f: switch specifies the form factor index, where:

- 0 = 320/360K bytes
- 1 = 1.2M bytes
- 2 = 720K bytes
- 3 = 8" single density
- 4 = 8" double density
- 5 = Hard disk
- 6 = Tape drive
- 7 = Other

The default values for the following parameters depend upon the form factor specified with the /f: switch. If you do not specify the /f: switch, **drivparm** uses a default of 720K bytes.

The *hb* option on the /h: switch specifies the maximum head number. Its value can range from 1 to 99.

The /n switch specifies a nonremovable block device.

The *ss* option for the /s: switch specifies the number of sectors per track. Its value can range from 1 to 99.

The *ttt* option for the /t: switch specifies the number of tracks per side on the block device. Its value can range from 1 to 999.

**Reconfiguring a
tape drive****Example:**

Suppose your computer has an internal tape drive on drive D that is configured at startup to write 20 tracks of 40 sectors per track. If you want to reconfigure this tape drive to write 10 tracks of 99 sectors each, you can put the following line in your *config.sys* file:

```
drivparm=/d:3 /f:6 /h:1 /s:99 /t:10
```

This command line overrides the default device driver settings, and supports a tape drive as drive D (in this case the logical and physical drive numbers are identical). This tape drive has 1 head and supports a tape format of 10 tracks and 99 sectors per track. (This assumes that the device driver for the tape device supports this configuration of tracks and sectors.) So, to create a tape that you can read on another computer, one which can read only this alternate format, you might want to use this method.

FCBS

FCBS

Purpose:

Allows you to determine the number of File Control Blocks (FCBs) that can be open concurrently.

Syntax:

`fcbs = x,y`

Comments:

The *x* option represents the number of files that File Control Blocks can open at any one time. The default value for *x* is 4, but allowed values range from 1 to 255.

If an application tries to open more than *x* files by FCBs, the *y* option specifies the number of files opened by FCBs that MS-DOS cannot close automatically. The first *y* files opened by FCBs are protected from being closed. The default value is 0, but allowed values range from 1 to 255.

Example:

To open up to four files by FCBs and to protect the first two files from being closed, put the following line in your *config.sys* file:

```
fcbs=4,2
```

**Setting the number of
open file control
blocks**

Files

Files**Purpose:**

Sets the number of open files that the MS-DOS system calls can access.

Syntax:

files = *x*

Comments:

The *x* option represents the number of open files that the system calls can access. The default value for *x* is 8, although allowed values range from 8 to 255. The maximum number of files that one program can have open at a time is 20.

MS-DOS system calls 2FH through 60H are compatible with the XENIX operating system.

Example:

To let MS-DOS open 20 files at one time, put the following line in your *config.sys* file:

f i l e s = 2 0

Setting the number of open files

Shell

Purpose:

Begins execution of the shell (top-level command processor) from a file defined by the specified *pathname*.

Syntax:

shell=[*drive:*]*pathname*

Comments:

The *pathname* option specifies the program that MS-DOS uses as a command processor. Instead of reading the standard *command.com*, MS-DOS starts the processor specified in *pathname*.

System programmers who write their own command processors (instead of using the MS-DOS file, *command.com*) should also use **shell**.

Example:

The following command uses the file *\bin\newshell* as the command processor:

```
shell=\bin\newshell
```

Shell

Setting the command processor shell

Sample Config.sys File

A typical configuration file might look like this:

```
buffers=10
files=10
device=\dev\network.sys
break=on
shell=a:\bin\command.com a:\bin /p
```

Note that the **buffers** and **files** commands are both set to 10.

To find the device that is being added to the system, MS-DOS searches for the pathname *\dev\network.sys*. This file is usually supplied on a disk with your device. Make sure that the device file is in the directory that you specify in the **device** command.

Break is turned on in this example.

This file also sets the MS-DOS command processor to the *command.com* file located on the disk in drive A in the *\bin* directory. *a:\bin* tells the command processor where to look for *command.com* if it needs to reread the disk. The */p* switch tells the command processor that it is the first program running on the system, so that it can process the MS-DOS **exit** command.

Appendix C

Installable Device Drivers

Introduction

Ansi.sys, *driver.sys*, and *ramdrive.sys* are three installable device drivers provided with MS-DOS. This appendix explains the details of these drivers. You should read Appendix B, "How to Configure Your System," for more information on how to install them.

Ansi.sys

An ANSI escape sequence is a series of characters (beginning with an escape character or keystroke) that you can use to define functions for MS-DOS. Specifically, you can change graphics functions and affect the movement of the cursor.

This appendix explains the ANSI escape sequences for MS-DOS. For examples of how to use these sequences, turn to the end of this appendix.

Notes:

- *Pn* represents the *numeric parameter*, a decimal number that you specify with ASCII digits.
- *Ps* represents the *selective parameter*, a decimal number that you use to select a subfunction. You may specify more than one subfunction by separating the parameters with semicolons.
- *Pl* represents the *line parameter*, a decimal number that you specify with ASCII digits.
- *Pc* represents the *column parameter*, a decimal number that you specify with ASCII digits.
- MS-DOS uses a default value when you do not specify a value or when you specify zero.

What is an ANSI escape sequence?

Changing the cursor's position**Cursor Functions**

The following escape sequences affect the position of the cursor on the screen. The parameters shown in italics are explained at the beginning of this appendix.

CUP — Cursor Position

ESC [*Pl* ; *Pc* H

HVP — Horizontal & Vertical Position

ESC [*Pl* ; *Pc* F

CUP and HVP move the cursor to the position specified by the parameters. The first parameter specifies the line number; the second, the column number. The default value for *Pl* and *Pc* is 1. When no parameters are specified, the cursor moves to the home position (the upper left-hand corner of the screen).

CUU — Cursor Up

ESC [*Pn* A

This sequence moves the cursor up one line without changing columns. The value of *Pn* sets the number of lines moved. The default value is 1. If the cursor is already on the top line, MS-DOS ignores the CUU sequence.

CUD — Cursor Down

ESC [*Pn* B

This sequence moves the cursor down one line without changing columns. The value of *Pn* sets the number of lines moved. The default value is 1. If the cursor is already on the bottom line, MS-DOS ignores the CUD sequence.

CUF — Cursor Forward

ESC [*Pn* C

The CUF sequence moves the cursor forward one column without changing lines. The value of *Pn* sets the number of columns moved. The default value is 1. If the cursor is already in the far right column, MS-DOS ignores the CUF sequence.

CUB — Cursor BackwardESC [*Pn* D

This escape sequence moves the cursor back one column without changing lines. The value of *Pn* sets the number of columns moved. The default value is 1. If the cursor is already in the far left column, MS-DOS ignores the CUB sequence.

DSR — Device Status Report

ESC [6 n

The console driver outputs an RCP sequence when it receives the DSR escape sequence.

SCP — Save Cursor Position

ESC [s

The console driver saves the current cursor position. This position can be restored by the RCP sequence.

RCP — Restore Cursor Position

ESC [u

This sequence restores the cursor position to the value it had when the console driver received the SCP sequence.

Erase Functions

The following escape sequences affect erase functions.

Affecting Erase Functions**ED — Erase Display**

ESC [2 J

The ED sequence erases the screen. The cursor then goes to the home position.

EL — Erase Line

ESC [K

This sequence erases from the cursor to the end of the line (including the cursor position).

Modes of Operation

Affecting Screen Graphics

The following escape sequences affect screen graphics, but they work only if your monitor supports graphics functions. The parameters shown in *italics* are explained at the beginning of this appendix.

SGR — Set Graphics Rendition

ESC [*Ps* ; ... ; *Ps* m

The SGR escape sequence calls the graphics functions specified by the parameters described in the following list. These functions remain until the next occurrence of an SGR escape sequence.

Parameter	Function
0	All attributes off
1	Bold on
2	Faint on
3	Italic on
5	Blink on
6	Rapid blink on
7	Reverse video on
8	Concealed on
30	Black foreground
31	Red foreground
32	Green foreground
33	Yellow foreground
34	Blue foreground
35	Magenta foreground
36	Cyan foreground
37	White foreground
40	Black background
41	Red background
42	Green background
43	Yellow background
44	Blue background
45	Magenta background
46	Cyan background
47	White background
48	Subscript
49	Superscript

Parameters 30 through 47 meet the ISO 6429 standard.

SM — Set Mode

ESC [= Ps h ESC [= h ESC [= 0 h ESC [? 7 h

The SM escape sequence changes the screen width or type to one of the following:

Parameter	Function
0	40 × 25 black and white
1	40 × 25 color
2	80 × 25 black and white
3	80 × 25 color
4	320 × 200 color
5	320 × 200 black and white
6	640 × 200 black and white
7	Wraps at the end of each line

RM — Reset Mode

ESC [= Ps l ESC [= l ESC [= 0 l ESC [? 7 l

Parameters for RM are the same as for SM (Set Mode), except parameter 7 resets the mode that causes wrapping at the end of each line.

Driver.sys

Driver.sys is an installable device driver that supports external drives.

Note If you want to configure a logical device, use the **drivparm** command described in Appendix B, "How to Configure Your System."

To install *driver.sys*, you must use the following syntax for the command line in your *config.sys* file:

device = *driver.sys* /**d:dd** [/c] [/f:ff] [/h:bb] [/n] [/s:ss] [/t:ttt]

The *dd* parameter on the /d: switch specifies a physical drive number between 0 and 255. Physical drives are numbered differently than logical drives. Floppy drives are numbered starting at 0, and hard drives are numbered starting at 80H.

For example, if you have a computer with two floppy drives, they might be numbered 0 and 1. If you were to add an external floppy drive, its physical drive number would be 02H.

If you have a computer with one floppy and one hard drive, the floppy drive is numbered 00H, but the hard drive is numbered 80H. If you add an external floppy drive, its physical drive number is still 02H because MS-DOS already knows the definitions of physical drives 00H and 01H.

The /c switch specifies that change-line (doorlock) support is required.

The *ff* option on the /f: switch specifies the form factor index, where:

- 0 = 320/360K bytes
- 1 = 1.2M bytes
- 2 = 720K bytes
- 3 = 8" single density
- 4 = 8" double density
- 5 = Hard disk
- 6 = Tape drive
- 7 = Other

The default values for the following parameters depend upon the form factor specified with the /f: switch. If you do not specify the /f: switch, *driver.sys* uses a default of 720K bytes.

Installing the Driver.sys device driver

The Driver.sys switches

The *bb* option on the */h:* switch specifies the maximum head number. Its value can range from 1 to 99.

The */n* switch specifies a non-removable block device.

The *ss* option on the */s:* switch specifies the number of sectors per track. Its value can range from 1 to 99.

The *ttt* option on the */t:* switch specifies the number of tracks per side on the block device. Its value can range from 1 to 999.

For example, if you want to add an external 720K-byte drive to your computer, you would include the following line in the *config.sys* file:

```
device=driver.sys /d:02
```

Installing the Ramdrive.sys device driver

Ramdrive.sys

Ramdrive.sys is a device driver that lets you use a portion of your computer's memory as if it were a disk drive. This memory area is called a RAM disk and is sometimes referred to as a virtual disk.

RAM disks are much faster than disk drives because the information they contain is always loaded into memory. If your computer has extended memory installed (starting at the 1 megabyte boundary), or if you have an extended memory board that meets the Lotus®/Intel®/Microsoft® Expanded Memory Specification, you can use this memory for one or more RAM disks. Otherwise, *ramdrive.sys* locates RAM drives in low memory.

Note This command increases the size of MS-DOS resident in memory.

To install *ramdrive.sys*, include a command of the following form in your *config.sys* file:

```
device = ramdrive.sys [bbbb] [ssss] [dddd] [/e]
```

or

```
device = ramdrive.sys [bbbb] [ssss] [dddd] [/a]
```

The *bbbb* option specifies the disk size in kilobytes. The default size is 64K bytes; the minimum, 16.

The *ssss* option specifies the sector size in bytes. The default value is 128 bytes. The following values are allowed: 128, 256, 512, and 1024 bytes.

The *dddd* option specifies the number of root directory entries. The default value is 64; the minimum, 2; and the maximum, 1024.

Ramdrive.sys adjusts the value of *dddd* to the nearest sector boundary. For example, if you give a value of 25 when the sector size is 512 bytes, the 25 will be rounded up to 32, since 32 is the next multiple of 16 (there are sixteen 32-byte directory entries in 512 bytes).

The */e* switch lets you use extended memory (above 1 megabyte) as a RAM disk if it has been installed. If you use this switch, you cannot use the */a* switch.

The */a* switch lets you use an extended memory board that meets the Lotus/Intel/Microsoft Expanded Memory Specification for a RAM drive, if that board has been installed. If you use this switch, you cannot use the */e* switch.

Note When you reset or turn off the power on your computer, any information stored in RAM disks is lost.



Appendix D

Disk and Device Errors

This appendix describes MS-DOS disk and device error messages. Refer to Appendix E, "MS-DOS Message Directory," for other MS-DOS messages.

If a disk or device error occurs at any time during a command or program, MS-DOS displays an error message in the following format:

type action device
Abort, Retry, Ignore: _

This appendix lists the *type*, *action*, and *device* messages in separate sections.

Type Messages

The *type* message is one of the following:

Bad call format error

- *The length of the request header passed to the device header was incorrect.*

Bad command error

- *A device driver issued an incorrect command to the device specified in the error message.*

Bad unit error

- *Invalid subunit numbers were passed to the device driver.*

Data error

- *MS-DOS could not read the data from the disk properly. This is often due to a defective disk.*
Try choosing R (for Retry) several times, or choose A (for Abort) to end the program. (It's a good idea to make a new copy of the disk, because if it's defective, you may lose information.)

B

D

FCB unavailable**General failure error**

- *An unusual error has occurred. This error usually requires an experienced programmer to fix it.*

Choose *R* (for Retry) or *A* (for Abort).

Invalid disk change error

- *You changed the disk in a drive when you weren't supposed to.*

Put the disk back in the drive and choose *R* (for Retry).

Lock violation error

- *A program tried to access part of a file that another program was using.*

Choose *A* (for Abort), or wait awhile and choose *R* (for Retry).

No paper error

- *The printer is either out of paper or not turned on.*

Non-DOS disk error

- *MS-DOS does not recognize the disk format because the disk is missing information or contains another operating system.*

Try running the **chkdsk** command to correct the problem.

(See Chapter 3, "MS-DOS Commands," for information about **chkdsk**.) If running **chkdsk** does not solve the problem, you should reformat the disk by using the **format** command—even though this will destroy all the files on the disk.

Not ready error

- *The device (usually a drive or printer) specified in the error message is not ready to accept or transmit data.*

This often happens when the disk drive door is open. If this is the problem, close the door and choose *R* (for Retry), or check to see if the printer is on and ready.

Read fault error

- *MS-DOS is unable to read data from the device (usually a disk drive).*

Check to see that the disk is properly inserted in the drive, then choose *R* (for Retry).

Sector not found error

- *This error usually means the disk has a defective spot so that MS-DOS cannot find the requested information on it.*

You should copy all files from the disk onto a good disk and then try to reformat the defective disk.

Seek error

- *MS-DOS is unable to locate the information on the disk.*

Make sure that the disk is properly inserted in the drive, or try a different drive.

Sharing violation

- *A program tried to access a file that another program was currently using.*

Choose A (for Abort), or wait awhile and choose R (for Retry).

Write fault error

- *MS-DOS is unable to write data to the specified device.*

Make sure that the disk is properly inserted in the disk drive. Try R (for Retry). If the error occurs again, choose A (for Abort).

Write protect error

- *You tried to write data on a write-protected disk.*

If the disk has a write-protect tab on it, you must remove the tab before you can write on the disk. (You should consider first why the disk was write-protected.) If the disk doesn't have a write-protect notch, you cannot write on that disk.

Action Messages

The *action* message may be either of the following:

reading
writing

Device Messages

The *device* is the name of the device that has the error. Examples are device PRN (for a printer) or drive A (for a disk drive). MS-DOS waits for you to type one of the following responses:

- | | |
|---|--|
| A | Abort. End the program requesting the disk read or write. |
| I | Ignore. Ignore the bad sector and pretend the error did not occur. This may result in lost data. |
| R | Retry. Repeat the operation. You should use this response when you have corrected the error (for example, with "Not ready" or "Write protect" errors). |

Usually, you will want to recover by typing responses in this order:

R (to try again)

A (to terminate a program and try a new disk)

One other error message might be related to faulty disk reading or writing:

File Allocation Table bad for drive x

This message means that the copy in memory of one of the allocation tables points to nonexistent blocks. Possibly the disk was incorrectly formatted or not formatted before use. If this error persists after you format it, the disk is unusable.

Appendix E

MS-DOS Message Directory

MS-DOS Messages

This section describes MS-DOS messages, their causes, and how to correct them. It includes a listing of the commands in brackets ([]) that cause each message.

For information specifically on device error messages, see Appendix D, "Disk and Device Errors."

Abort edit (Y/N)?

[Edlin]

- *MS-DOS displays this message when you choose the Edlin Q (quit) command. The Q command exits the editing session without saving any editing changes.*

Type Y (for Yes) or N (for No).

Abort, Retry, Ignore?

[MS-DOS]

- *This is a device error. See Appendix D, "Disk and Device Errors."*

Access denied

[Attrib][Find][Print][Replace][Xcopy]

- *You tried to replace a write-protected, read-only, or locked file.*

Add filename? (Y/N)

[Replace]

- *Replace displays this prompt if you specify the /w switch.*

Press Y if you want to add the file to the disk, or N if you do not want to add the file.

Adding filename

[Replace]

- *Replace displays this prompt to let you know that it is adding this file to your disk.*

All files canceled by operator

[Print]

- *MS-DOS displays this message when you specify the /t switch with the Print command.*

Allocation error, size adjusted

[Chkdsk]

- *The size of the file indicated in the directory was not consistent with the amount of data actually allocated to the file. The file was truncated to match the amount of data allocated.*

All specified file(s) are contiguous

[Chkdsk]

- *All files are written sequentially on the disk.*
To correct this error automatically, specify the **chkdsk /f** switch.

APPEND/ASSIGN Conflict

[Append]

- *You cannot use the Append command on an assigned drive.*
Cancel the drive assignment before using the **append** command with this drive again.

Are you sure (Y,N)?

[MS-DOS]

- *MS-DOS displays this message if you try to delete all files in the working directory by using the *.* wildcard.*
Type Y (for Yes) to delete all the files, or N (for No).

Attempted write-protect violation

[Format]

- *The disk you are trying to format is write-protected.*

***** Backing up files to drive x: *****

Diskette Number: n

[Backup]

- *Backup displays this message while backing up files to the specified target drive.*
Be sure to label backup disks with the appropriate backup disk number for use in restoring them later.

Bad call format reading (or writing) drive x:

[MS-DOS]

- *This is a device error. See Appendix D, "Disk and Device Errors."*

Bad command error reading (or writing) drive x:

[MS-DOS]

- *This is a device error. See Appendix D, "Disk and Device Errors."*

Bad command or file name

[MS-DOS]

- *The command cannot find the program you asked it to run.*
Check to see that you typed the command line properly, and that the file or command is on the disk, or in the command path.

Bad or missing Command Interpreter

[MS-DOS]

- *MS-DOS cannot find the command.com file on the disk; either the file is missing from the root directory, or the file is invalid. You also receive this message if command.com has been moved from the directory it was originally in when you started MS-DOS.*

Either restart the system with a disk that contains the *command.com* file, or copy the *command.com* file from your backup MS-DOS master disk onto the disk used to start MS-DOS.

Bad or missing filename

[MS-DOS]

- *You specified a device incorrectly in the config.sys file.*
Check the accuracy of the **device** command in the *config.sys* file.

Bad Partition Table

[Format]

- *This message means that there is no DOS partition on the hard disk.*
You must run **fdisk** to create a DOS partition on your hard disk.

Bad unit error reading drive x:

[MS-DOS]

- *This is a device error. See Appendix D, "Disk and Device Errors."*

BREAK is off (or on)

[MS-DOS]

- *This message tells you the current setting of Break.*

C

**Cannot CHDIR to path -
tree past this point not processed**

[Chkdsk]

- *Chkdsk is checking the structure of the directory and is unable to go to the specified directory. All subdirectories underneath this directory will not be verified.*

To correct this error automatically, specify the **chkdsk /f** switch.

Cannot CHDIR to root

[Chkdsk]

- *Chkdsk is checking the tree structure of the directory and is unable to return to the root directory. Chkdsk is not able to continue checking the remaining subdirectories.*

Try to restart MS-DOS. If this error persists, the disk is unusable.

Cannot CHKDSK a Network drive

[Chkdsk]

- *You cannot check drives that are redirected over the network.*

Cannot CHKDSK a SUBSTed or ASSIGNed drive

[Chkdsk]

- *You cannot check drives that have been substituted or assigned.*

Cannot COPY from (or to) a reserved device

[Xcopy]

- *You cannot copy files from or to a device.*

Cannot create Subdirectory BACKUP on drive x:

[Backup]

- *The disk may be write-protected, full, or the backup subdirectory may already exist and be read-only.*

Use another disk as a target disk.

**Cannot DISKCOMP to or from
an ASSIGNed or SUBSTed drive**

[Diskcomp]

- *One of the drives that you specified is a drive that you created using the Assign or Subst command.*

**Cannot DISKCOMP to or from
a network drive**

[Diskcomp]

- *You cannot compare disks on drives that have been redirected over the network.*

Cannot DISKCOPY to or from an ASSIGNED or SUBSTed drive

[Diskcopy]

- *One of the specified drives was created using the Assign or Subst command.*

Cannot DISKCOPY to or from a network drive

[Diskcopy]

- *You cannot copy disks to or from drives that have been redirected over the network.*

Cannot do binary reads from a device

[Copy]

- *The copy cannot be done in binary mode when you are copying from a device.*

You should either not use the /b switch, or you should use the /a switch to specify an ASCII copy.

Cannot edit .BAK file--rename file

[Edlin]

- *You attempted to edit a file that had a filename extension of .bak (a backup copy created by Edlin).*

If you must edit a file that has an extension of .bak, you must either rename or copy the file and give it a different extension.

Cannot format an ASSIGNED or SUBSTed drive

[Format]

- *You attempted to format a drive currently mapped to another drive by the Assign or Subst command.*

Run **assign** or **subst** again and clear all drive assignments.

Cannot FORMAT a Network drive

[Format]

- *You cannot format drives that are redirected over the network.*

Cannot JOIN a Network drive

[Join]

- *You cannot join drives that are redirected over the network.*

Cannot LABEL a Network drive

[Label]

- *You cannot label a drive that is shared on a network server station.*

Cannot LABEL a SUBSTed or ASSIGNED drive

[Label]

- *You cannot label a drive if it has been substituted with the Subst command or assigned with the Assign command.*

Check the command line to be sure you specified a valid filename.

Cannot perform a cyclic copy

[Xcopy]

- *When you are using the /s switch, you may not specify a target that is a subdirectory of the source.*

Cannot recover . entry, processing continued

[Chkdsk]

- *The "." entry (working directory) is defective and cannot be recovered.*

Cannot recover .. entry,

Entry has a bad attribute (or link or size)

[Chkdsk]

- *The ".." entry (parent directory) is defective and cannot be recovered.*

If you have specified the /f switch, **chkdsk** tries to correct the error automatically.

Cannot RECOVER a Network drive

[Recover]

- *You cannot recover files on drives that are redirected over the network.*

Cannot SUBST a Network drive

[Subst]

- *You cannot substitute drives that are redirected over the network.*

Cannot SYS to a Network drive

[Sys]

- *You cannot transfer the system files to drives that are redirected over the network.*

For more information about the **net print** command, see the *Microsoft Networks User's Guide*.



Cannot use PRINT - Use NET PRINT

[Print]

- *You must use the Net Print command to print files.*

CHDIR .. failed, trying alternate method

[Chkdsk]

- *When checking the tree structure, Chkdsk was not able to return to a parent directory. It will try to return to that directory by starting over at the root and searching again.*

Compare another diskette (Y/N)?

[Diskcomp]

- *Diskcomp displays this message when it has completed its comparison of the disks.*

Press Y (for Yes) if you want to compare more disks.

Compare error on disk**side s, track t**

[Diskcomp]

- *Diskcomp found a difference on the disk in the specified drive, side s, track t.*

Compare OK

[Diskcomp]

- *Diskcomp displays this message if the disks are identical.*

Compare process ended

[Diskcomp]

- *Diskcomp displays this message if a fatal error occurred during the comparison.*

Comparing t tracks**n sectors per track, s side(s)**

[Diskcomp]

- *This message confirms the format of the disks that you are comparing.*

COM port does not exist

[Mode]

- *You have specified an invalid COM port.*

Contains n non-contiguous blocks.

[Chkdsk]

- *The disk contains fragmented files.*

If you want to copy this disk, you should use the **copy** or **xcopy** command instead of the **diskcopy** command. The new copy will then store the new files sequentially.

Content of destination lost before copy

[Copy]

- *The source file that you specified in the Copy command was overwritten before the copy process completed.*

Refer to the **copy** command for the proper syntax.

Convert lost chains to files (Y/N)?

[Chkdsk]

- *Chkdsk displays this message if it finds information on the disk that isn't allocated properly in the disk's File Allocation Table.*

If you type *Y* (for Yes) in response to this prompt, **chkdsk** recovers the lost blocks it found when checking the disk. **Chkdsk** then creates a proper directory entry and a file for each lost chain with a filename of the form: *filennnn.chk*. If you type *N* (for No), **chkdsk** frees the lost blocks so that they can be reallocated and does not recover any data that was in those lost blocks.

Copy another diskette (Y/N)?

[Diskcopy]

- *The Diskcopy command has completed processing.*

Type *Y* (for Yes) if you want to copy another disk, or *N* (for No) if you don't.

**Copying *t* tracks
n Sectors/Track, *s* Sides**

[Diskcopy]

- *Diskcopy displays this message during copying.*

Copy process ended

[Diskcopy]

- *Diskcopy could not copy the entire disk.*

Use the **copy** or **xcopy** command to copy specific files onto the disk.

Copyright 1981,82,83,84,85,86 Microsoft Corp.

[MS-DOS]

- *This message appears on most MS-DOS utility and command banners.*

Corrections will not be written to disk

[Chkdsk]

- *There are errors on the disk, but Chkdsk will not correct them because you did not specify the /f switch.*

You must specify the **chkdsk** switch to correct disk errors.

Current date is *mm-dd-yy*

[Date]

- *The Date command displays this message.*

Enter the correct date and press RETURN.

Current time is hh:mm:ss.hh

[Time]

- *The Time command displays this message.*

Enter the correct time and press RETURN.

Data error reading drive x:

[MS-DOS]

- *This is a device error. See Appendix D, "Disk and Device Errors."*

Delete current volume label (Y/N)?

[Label]

- *If a current volume label exists, Label displays this message in response to the prompt to enter the new volume label for this disk.*

If you want to delete the volume label, press Y (for Yes); otherwise, press N (for No).

DEVICE Support Not Present

[Diskcomp][Diskcopy]

- *The disk drive does not support MS-DOS 4.0 device control.*

Directory is joined

[Chkdsk]

- *Chkdsk does not process directories that are joined.*

Use the **join /d** command to "unjoin" the directories, and then run **chkdsk** again.

Directory is totally empty,**no . or ..**

[Chkdsk]

- *The specified directory does not contain references to working and parent directories.*

Delete the specified directory and recreate it.

Directory not empty

[Join]

- *You can only join onto an empty directory.*

Disk error reading (or writing) drive x:

[MS-DOS]

- *This is a device error. See Appendix D, "Disk and Device Errors."*

Disk error reading (or writing) FAT

[Chkdsk]

- *One of your File Allocation Tables has a defective sector in it. MS-DOS automatically uses the other FAT.*

You should copy all your files onto another disk. To correct this error automatically, you simply specify the **chkdsk /f** switch.

Diskette bad or incompatible

[Diskcopy]

- *The source disk is not formatted, or was formatted incorrectly. You cannot copy it.*

Disk full. Edits lost

[Edlin]

- *Edlin was not able to save your file due to lack of disk space.*
- You should always make sure that there is enough room on the default disk to save your file before you give the **edlin E** (end) command. You should also make sure that the default disk is not write-protected.

Disk unsuitable for system disk

[Format]

- *The Format program detected a bad track on the disk where system files should reside.*
- You should use this disk to store data only.

**Does name specify a file name
or directory name on the target
(F = file D = directory)?**

[Xcopy]

- *Xcopy displays this prompt if the target directory does not exist.*
- Type *F* if the name specifies a file, or *D* if the target specifies a directory that does not currently exist.

(.)(..) Does not exist

[Chkdsk]

- *This is an informational message from Chkdsk, indicating that either the "." or ".." directory entry is invalid.*

Do not specify filename(s)**Command format: DISKCOMP d: d:[/1][/8]**

[Diskcomp]

- *You specified an incorrect switch or gave a filename in addition to a drive name.*

Do not specify filename(s)**Command format: DISKCOPY d: d:[/1]**

[Diskcopy]

- *You specified an incorrect switch or gave a filename in addition to a drive name.*

DOS 2.0 or later required

[Attrib][Backup][FC][Graphics][Join][Mode][Restore][Subst]

- *You cannot use these utilities with 1.xx versions of MS-DOS.*

Do you see the leftmost 0? (Y/N)

[Mode]

- *Mode displays this message to help you align the test pattern on your screen.*

Type *Y* (for Yes) if you can see the leftmost 0 in the test pattern, or type *N* (for No) if you want to shift the display to the right.

Do you see the rightmost 9? (Y/N)

[Mode]

- *Mode displays this message to help you align the test pattern on your screen.*

Type *Y* (for Yes) if you can see the rightmost 9 in the test pattern, or type *N* (for No) if you want to shift the display to the left.

Drive letter must be specified

[Format]

- *You did not specify the drive letter for the drive that you want to format.*

You must specify the name of the drive that you want to format.

Drive x: not ready

Make sure a diskette is inserted into the drive and the door is closed

[Diskcomp][Diskcopy]

- *The drive is empty, or you did not close the door of the disk drive.*

Drive types or diskette types not compatible

[Diskcomp][Diskcopy]

- *You must have the same size and type of disks to run these commands. For example, you cannot copy from a single-sided disk to a double-sided disk, or compare a high-density disk with a low-density disk.*

You should use FC if you want to compare the files on the disks. If you want to copy the disk, you can use **copy** or **xcopy**, or reformat the target disk so that it's the same type as the source disk, or use a disk of the same type.

Duplicate file name

[Rename]

- *You tried to rename a file to a filename that already exists, or the name you specified could not be found.*

ECHO is off (or on)

[MS-DOS]

- *This message tells you the current status of Echo.*

End of input file

[Edlin]

- *The entire file was read into memory. If the file was read in sections, this message indicates that the last section of the file is in memory.*

Enter current Volume Label for drive x:

[Format]

- *Format asks you to enter the current volume label for verification before it formats the hard disk in the specified drive.*
If you do not know what the volume label is, press CONTROL-C to abort this command, and give the **vol** command for the specified drive. Then give the **format** command again.

Enter new date:

[Date]

- *You must respond to this prompt when you start MS-DOS, or when you use the Date command.*
Enter the date in a *mm-dd-yy* format, or press the RETURN key to accept the current date.

Enter new time:

[Time]

- *You must respond to this prompt when you start MS-DOS.*
Enter the time in the *hh:mm* format, or press the RETURN key to accept the current time.

Entry error

[Edlin]

- *The last command you typed contained a syntax error.*
Retype the command with the correct syntax and press the RETURN key.

Entry has a bad attribute (or link or size)

[Chkdsk]

- *This message may be preceded by one or two periods that show which subdirectory is invalid.*

If you have specified the /f switch, **chkdsk** tries to correct the error automatically.

Error in .EXE file

[MS-DOS]

- *The .exe file you have asked MS-DOS to load has an invalid internal format.*

You cannot run this program. Check to make sure that you are using the correct version of MS-DOS.

Error reading/writing partition table

[Format]

- *Format could not read or write the partition table.*

You should run **fdisk** on the disk and then try formatting it again.

Errors found, F parameter not specified**Corrections will not be written to disk**

[Chkdsk]

- *Chkdsk found errors on the disk. If you have not specified the /f switch, Chkdsk continues printing messages but will not correct the errors.*

You should run **chkdsk** with the /f switch if you want to correct the problems encountered by the **chkdsk** command.

Errors on list device indicate that it may be off-line. Please check it.

[Print]

- *Your printer is not turned on.*

**Error trying to open backup log file
Continuing without making log entries.**

[Backup]

- *You specified the Backup /L switch, but Backup could not create the backup log file.*

Error writing to device

[MS-DOS]

- *You tried to send too much data to a device, so MS-DOS was unable to write the data to that device.*

EXEC failure

[MS-DOS]

- *MS-DOS either found an error when reading a command, or the Files command in the config.sys file is set too low.*

Increase the value of the **files** command in the *config.sys* file, and restart MS-DOS.

FCB unavailable reading (or writing) drive x:

[MS-DOS]

- *This is a device error. See Appendix D, "Disk and Device Errors."*

fc: cannot open filename - No such file or directory

[FC]

- *One of the files that you specified doesn't exist.*

Check the directory for the correct filename.

fc: filename longer than filename

[FC]

- *After reaching the end of one of the files in a file comparison, the other file still has uncompared data left.*

fc: incompatible switches

[FC]

- *You have specified switches that are not compatible. (For example, /b and /L.)*

You should not combine binary and ASCII comparison switches.

fc: no differences encountered

[FC]

- *The files are the same.*

fc: out of memory

[FC]

- *You do not have enough memory to perform the comparison.*

File allocation table bad

[MS-DOS]

- *The disk may be defective.*

Run **chkdsk /f** to check the disk.

File allocation table bad drive x:

[Chkdsk]

- *This message means that the disk was not formatted or was formatted improperly. It could also mean that an operating system other than MS-DOS is on the disk.*

Run **chkdsk /f** to check the disk. If this message is displayed again, you must reformat the disk.

File filename canceled by operator

[Print]

- *MS-DOS displays this message when you specify the /t switch with the Print command.*

File cannot be converted

[Exe2bin]

- *The input file is not in the correct format.*

File cannot be copied onto itself

[Copy][Replace][Xcopy]

- *The source filename you specified is the same as the target filename.*

File creation error

[MS-DOS][Edlin][Restore][Xcopy]

- *You tried to add a new filename or replace a file that already exists in the directory, or there was not enough space for the file. If the file already exists, it is a read-only file and cannot be replaced. This error message may also occur if the root directory is full, out of files, or if the filename is the same as a volume or directory, or a hidden (or system) file.*

File is READ-ONLY

[Edlin]

- *The file is designated read-only, so you may not change it.*

File name must be specified

[Edlin]

- *You did not specify a filename when you started Edlin.*
You should type the **edlin** command followed by a filename.

File not found

[Chkdsk][Edlin][FC][Find][Print][Recover][Rename][Xcopy]

- *MS-DOS could not find the file that you specified, or you tried to rename a file with a name already in the directory.*
Check to see that you entered the filename correctly.

File not in PRINT queue

[Print]

- *The file that you specified was not in the print queue, so you cannot remove it from the queue.*
Check to see that you entered the filename correctly.

**Files cannot be added to this diskette
Unless the PACK (/P) switch is used
Set the switch (Y/N)?**

[Backup]

- *The target disk does not have enough room for any of the files on the source disk without dividing them across disks.*

If you do not want to divide a file across disks, type *N* (for No).

If your files are larger than will fit on one floppy disk, you must type *Y* (for Yes).

***** Files were backed up
at time on date *****

[Restore]

- *This is an information message only.*

FIND: Access denied

[Find]

- *You cannot access the file.*

Make sure that the disk is not write-protected, read-only, or locked.

FIND: File not found

[Find]

- *MS-DOS could not find the file that you specified.*

Make sure you have typed the filename correctly.

FIND: Invalid number of parameters

[Find]

- *You specified either too many or too few options in the command line.*

FIND: Invalid Parameter

[Find]

- *One of the switches you specified is wrong.*

FIND: Read error in filename

[Find]

- *The Find command could not read the specified file.*

FIND: Syntax error

[Find]

- *Check to make sure that you have typed the command correctly.*

First cluster number is invalid, entry truncated

[Chkdsk]

- *The file directory entry contains an invalid pointer to the data area. If you specified the /f switch, the file is truncated to a zero-length file.*

FIRST diskette bad or incompatible

[Diskcomp]

- *Diskcomp cannot recognize the format on the source disk. You should run **chkdsk** to help you identify the problem.*

Fixups needed - base segment hex:

[Exe2bin]

- *The source (.exe) file contained information indicating that a load segment is required for the file. You must specify the absolute segment address where the finished module is to be located.*

For cannot be nested

[MS-DOS]

- *You cannot nest For commands in a batch file.*

Format another (Y/N)?

[Format]

- *Format displays this message when it has finished formatting a disk.*

Type *Y* (for Yes) if you want to format another disk, or type *N* (for No) if you don't. If you accidentally type *Y*, you can abort the format process by typing CONTROL-C in response to the message "Strike any key."

Format complete

[Format]

- *Format displays this message when it has finished formatting the disk in the specified drive.*

Format failure

[Format]

- *MS-DOS could not format the disk. This message is usually displayed with an explanation as to why the command failed.*

Format not supported on drive x:

[Format]

- *You cannot use Format to format this drive. You may have specified device parameters that your computer cannot support.*

Formatting while copying

[Diskcopy]

- *Diskcopy displays this message if the target disk has never been formatted.*

General failure reading (or writing) drive x:

[MS-DOS]

- *This is a device error. See Appendix D, "Disk and Device Errors."*

Graftabl needs DOS version 2.0 or later

[Graftabl]

- *You cannot use Graftabl with 1.xx versions of MS-DOS.*

Graphics characters already loaded

[Graftabl]

- *The Graftabl command displays this message if you have already loaded the table of graphics characters into memory.*

Graphics characters loaded

[Graftabl]

- *The Graftabl command displays this message after it loads the table of graphics characters into memory.*

Has invalid cluster, file truncated

[Chkdsk]

- *The file directory entry contains an invalid pointer to the data area. If you specified the /f switch, the file is truncated to a zero-length file.*

Head: hhh Cylinder: ccc

[Format]

- *Format displays the head and cylinder number of the track currently being formatted.*

Illegal device name

[Mode]

- *Your computer does not recognize this device name.*

Incompatible system size

[Sys]

- *The system files occupy more space on the source disk than is available on the target disk.*

You cannot use the sys command to transfer the system files to this disk.

Incorrect DOS Version

[Append][Attrib][Backup][Chkdsk][Diskcomp][Diskcopy]
 [Edlin][FC][Find][Format][Graphics][Join][Keybxx]
 [Label][Mode][More][Print][Recover][Replace][Restore]
 [Share][Sort][Subst][Sys][Tree][Xcopy]

- *Some MS-DOS utilities will not run on older versions of the operating system, and many are written to run only on the exact version of MS-DOS that they were created for.*

You must use the correct version of MS-DOS to run this command.

Incorrect number of parameters

[Join][Subst]

- *You specified either too many or too few options in the command line.*

Incorrect parameter

[Assign][Share]

- *One of the options you specified is wrong.*

Infinite retry on parallel printer timeout

[Mode]

- *Your printer is probably off-line or not ready.*

If the printer appears to be ready, you may have to press the CONTROL-ALT-DEL keys to reset the computer.

Insert backup diskette *n* into drive *x*:

[Backup][Restore]

- *This message prompts you for the *n*th backup disk.*

Put the next disk into the specified drive. Be sure to label each backup disk in the appropriate order for use when restoring the files.

**Insert destination disk in drive *x*:
and strike any key when ready**

[Sys]

- *This message appears when you are using Sys to transfer the operating system with a single disk drive.*

You should insert a disk in the appropriate drive and press any character or number key to begin processing.

**Insert diskette for drive *x*:
and strike any key when ready**

[MS-DOS]

- *This message appears when MS-DOS is copying and formatting.*

You should insert a disk in the appropriate drive and press any character or number key to begin processing.

**Insert diskette with batch file
and press any key when ready**

[MS-DOS]

- *The disk containing your batch file is not in the drive you originally specified.*

Reinsert the disk that contains the batch file in the appropriate drive.

**Insert DOS diskette in drive x:
and strike ENTER when ready**

[Format]

- *You typed the Format /s command, but the disk in the default drive does not contain MS-DOS system files.*

Insert a disk with the files *io.sys* and *msdos.sys* in the drive specified and press any key.

Insert FIRST diskette into drive x:

[Diskcomp]

- *This message prompts you for the first disk that you want to compare.*

Insert last backup diskette in drive x:

Strike any key when ready

[Backup]

- *This message prompts you for the final backup disk.*

After you have put the final backup disk into the drive specified, press any alphanumeric key to continue the backup process.

Insert restore target diskette into drive x:

[Restore]

- *Restore displays this prompt if you are restoring files to a floppy.*

Put the target disk into the specified drive.

Insert SECOND diskette into drive x:

[Diskcomp]

- *This message prompts you for the disk that you want to compare with the first disk.*

Insert source disk

[Backup]

- *This message prompts you to put the source disk into the drive.*

Insert SOURCE diskette into drive x:

[Diskcopy]

- *This message prompts you to put the disk to be copied into the specified drive.*

**Insert system diskette in drive x:
and strike any key when ready**

[Sys]

- *Sys needs a disk from which to read the io.sys and msdos.sys files.*

Insert a system disk into the specified drive and press any character or number key to start the system copy process.

Insert TARGET diskette into drive x:

[Diskcopy]

- *Diskcopy displays this message to prompt you to place the target disk into the specified drive. If your computer has one floppy drive, this message prompts you to put the proper disk into the drive.*

Insufficient disk space

[MS-DOS][Replace][Sort][Xcopy]

- *The disk is full and does not contain enough room to perform the specified operation.*

Insufficient memory

[Backup][Chkdsk][Diskcomp][Diskcopy][Edlin][Replace]

[Restore][Sort][Xcopy]

- *There is not enough memory in your computer to perform the specified operation.*

Before retrying this operation, you must free memory by deleting files. In **edlin**, you may be able to free memory by typing a **W** (write) command followed by an **A** (append) command.

Insufficient memory for system transfer

[Format]

- *Your memory configuration is insufficient to transfer the MS-DOS system files io.sys and msdos.sys with the Format /s switch.*

Insufficient room in root directory.**Erase files in root and repeat CHKDSK**

[Chkdsk]

- *Chkdsk always recovers lost files into the root directory. In this case, your root directory is full.*

Delete some files in your root directory, or move them to another directory to make room for the lost files.

Intermediate file error during pipe

[MS-DOS]

- *The pipe operation uses temporary files on the disk that are deleted automatically once the piping process is complete. An error has occurred in one of these files.*

Make sure that there is enough room on the disk for the temporary file and that the disk is not write-protected, and try the command again.

Internal error

[FC][Mode][Share]

- *This message indicates an error in the utility.*

Invalid argument

[Backup][FC][Restore]

- *You have specified an invalid argument.*

Refer to Chapter 3, "MS-DOS Commands," for the correct syntax of the command, and try again.

Invalid baud rate specified

[Mode]

- *You have specified an incorrect baud rate. Valid choices are 110, 150, 300, 600, 1200, 2400, 4800, and 9600.*

You must specify at least the first two digits of the baud rate.

Invalid characters in volume label

[Format][Label]

- *The volume label should only contain up to 11 alphanumeric characters.*

Invalid COMMAND.COM**Insert COMMAND.COM disk in default drive and strike any key when ready**

[MS-DOS]

- *The program you have just run used up almost all of memory. MS-DOS must now reload the command.com file from disk. However, either MS-DOS cannot find command.com on the disk, or the copy found is the incorrect version.*

Insert a disk that contains a copy of *command.com* into the default drive (it must be the same version with which you started MS-DOS).

Invalid country code

[MS-DOS]

- *In your config.sys file you have specified a country number that is not in the table of files configured in this version of MS-DOS. Country codes must be in the range 1–99 and are set by your computer manufacturer.*

Invalid current directory

[Chkdsk]

- *Your disk has an invalid directory on it.*

You may be able to recover some of the files on this disk by copying them with the **copy** command. Otherwise, you must replace the disk.

Invalid date

[Date][Xcopy]

- *You specified an invalid date in response to the date prompt.*

Enter a valid date. Refer to Chapter 3, "MS-DOS Commands," for the proper syntax of the **date** command.

Invalid Date/Time

[Backup]

- *You specified an invalid date with one of the Backup command switches.*

Refer to Chapter 3, "MS-DOS Commands," for the proper syntax of the **backup** command.

Invalid device

[MS-DOS]

- *The device specified was not AUX, CON, NUL, or PRN.*

Invalid device parameters from device driver

[Format]

- *Format displays this message when the number of hidden sectors is not evenly divisible by the number of sectors per track (that is, the partition does not start on a track boundary). This might happen if you tried to format a hard disk that previously had been formatted with MS-DOS 2.x without first running Fdisk, or if you have set the device driver parameters incorrectly.*

Check the **config.sys** file for incorrect **device** or **drivparm** commands.

Invalid directory

[MS-DOS]

- *The directory you specified either does not exist or is invalid.*

Check to see that you entered the directory name correctly.

Invalid disk change reading (or writing) drive x:

[MS-DOS]

- *This is a device error. See Appendix D, "Disk and Device Errors."*

Invalid drive in search path

[MS-DOS]

- *The drive does not exist.*

Invalid drive or filename

[Edlin][Recover]

- *You did not type a valid drive name or filename. Enter a valid drive name or filename.*

Invalid drive specification

[Backup][Chkdsk][Diskcomp][Diskcopy][Format][Label]

[Print][Replace][Restore][Sys][Tree][Xcopy]

- *The drive is incorrect or does not exist.*

Enter a valid drive name.

Invalid environment size specified

[Command]

- *You gave an invalid number of bytes with the /e switch.*

You must specify a number between 128 and 32768 (bytes).

Invalid number of parameters

[Attrib][Backup][FC][Find][Recover][Restore][Xcopy]

- *Either you did not specify an option or string, or you specified the wrong number of options in the command line.*

Invalid parameter(s)

[Backup][Chkdsk][Diskcomp][Diskcopy][Edlin][Find]

[Format][Join][Mode][Print][Replace][Restore][Sort]

[Subst][Sys][Tree][Xcopy]

- *One of the switches you specified is wrong or does not exist.*

Refer to Chapter 3, "MS-DOS Commands," to make sure you are using the correct switches.

Invalid path, not directory, or directory not empty

[MS-DOS]

- *You are unable to remove the directory requested for one of the specified reasons.*

Invalid path (or file not found)

[Attrib][Backup][Copy][Restore][Tree][Xcopy]

- *You have entered a pathname or filename that does not exist.*

Enter a valid pathname or filename with the command.

Invalid sub-directory entry

[Chkdsk]

- *The subdirectory that you specified either does not exist or is invalid.*

Check to see that you typed the subdirectory name correctly.

Invalid time

[Time]

- *You specified an invalid time.*

Refer to Chapter 3, "MS-DOS Commands," for the correct syntax, and try the command again.

Invalid Volume ID

[Format]

- *Format displays this message if you enter a volume label that doesn't match the label on the hard disk you want to format. It then quits the format process.*

Use the **vol** command to find out what the volume label for the hard disk is, then try the command again.

Label not found

[MS-DOS]

- *Your batch file contains a Goto command to a nonexistent label.*

Last backup diskette not inserted**Insert last backup diskette in drive x:****Strike any key when ready**

[Backup]

- *This message prompts you for the final backup disk.*

After you have put the final backup disk into the drive specified, press any alphanumeric key to continue the backup process.

***** Last file not backed up *****

[Backup]

- *Backup could not back up the last file on the disk. This message may occur if there is no more room on the target disk. It may also occur if there was an error in the source file, or on the target disk.*

You may have to back up this file separately to another disk.

Line too long

[Edlin]

- *During an Edlin R (replace) command, the string given as the replacement caused the line to expand beyond the limit of 253 characters.*

You should divide the long line into two lines and retry the **R** command.

List output is not assigned to a device

[Print]

- *When you first type the Print command, MS-DOS asks you what device you want to specify as a printer. This message appears if Print is set up for a device that does not exist.*

Lock violation reading (or writing) drive x:

[MS-DOS]

- *This is a device error. See Appendix D, "Disk and Device Errors."*

x lost cluster(s) found in y chains**Convert lost chains to files (Y/N)?**

[Chkdsk]

- *Chkdsk displays this message if it finds information on the disk that isn't allocated properly in the disk's File Allocation Table.*

If you type *Y* in response to this prompt, **chkdsk** recovers the lost blocks it found when checking the disk. **Chkdsk** then creates a proper directory entry and a file for each lost chain with the filename of the form: *filennnn.chk*. If you did not specify the */f* switch, **chkdsk** displays: "x bytes would be freed." If you type *N*, **chkdsk** frees the lost blocks so that they can be reallocated and does not recover any data that was in those lost blocks. If you did not specify the */f* switch, **chkdsk** does nothing.

LPT#: not redirected

[Mode]

- *Mode could not redirect the parallel printer port.*
Check to see whether you have specified the proper options.

LPT#: redirected to COM#:

[Mode]

- *Output on the parallel printer port will now be sent to this asynchronous communications port.*

LPT#: set for 80

[Mode]

- *The parallel printer port has been set for 80 columns.*

LPT#: set for 132

[Mode]

- *The parallel printer port has been set for 132 columns.*

Memory allocation error. Cannot load MS-DOS, system halted

[MS-DOS]

■ *Restart MS-DOS.*

If this error persists, make a new copy of the MS-DOS disk from your backup copy of the system disk.

--More--

[More]

■ *Press the SPACEBAR to view more of the file or directory.*

MORE: Incorrect DOS version

[More]

■ *The More command does not run on MS-DOS versions before 2.0.*

Must specify destination line number

[Edlin]

■ *You did not specify the destination line number for an Edlin C (copy) or M (move) command.*

Retype the command with a destination line number.

Must specify ON or OFF

[MS-DOS]

■ *The command requires either an ON or an OFF argument.*

Name of list device [PRN]:

[Print]

■ *This prompt appears the first time that Print is run and the /d switch is not specified.*

You can specify the name of any valid device, which then becomes the **print** output device. If you press the RETURN key, MS-DOS uses the default list device PRN.

New file

[Edlin]

■ *Edlin prints this message if it does not find a file with the name you specified.*

If you are creating a new file, ignore this message. If you do not intend to create a new file, check to see whether you have correctly typed the name of the file that you wish to edit.

No appended directories

[Append]

■ *You did not specify a path with the Append command.*

No COM: ports

[Mode]

- *Your computer does not have an asynchronous communications (serial) port.*

No files added (or replaced)

[Replace]

- *The Replace command did not add or replace any files.*

No files found filename

[Replace]

- *Replace could not find matching source or target files.*

No free file handles.

Cannot start COMMAND.COM, exiting

[MS-DOS]

- *Restart MS-DOS.*

If this message persists, increase the **files** command value in the *config.sys* file.

Non-DOS disk error reading (or writing) drive x:

[MS-DOS]

- *This is a device error. See Appendix D, "Disk and Device Errors."*

Non-system disk or disk error

Replace and strike any key when ready

[Format][Sys]

- *Replace the disk with the proper disk and press any alphanumeric key to continue.*

No paper error writing device dev

[MS-DOS]

- *This is a device error. See Appendix D, "Disk and Device Errors."*

No path

[Path]

- *You typed Path and pressed the RETURN key to find out what your search path is, but you didn't set a command search path.*

No room for system on destination disk

[Sys]

- *There is not enough room for the system files on the target disk.*

Delete some files to make room for the system files or use another disk. You may need to reformat the disk to put the system on it.

No room in directory for file

[Edlin]

- *You tried to create or save a file to the root directory, but it is either full, or you specified an invalid disk drive or filename.*

Check the command line that you used to start **edlin** for an invalid filename or disk drive entry. If your command contains no invalid entries, you should run the **chkdsk** program for the specified disk drive. If the status report shows that the disk directory is full, and if there is still enough memory left on the disk, you may be able to create the file in a subdirectory. (This is because subdirectories are not limited in size as is the root directory.) Otherwise, remove the disk and replace it with another formatted disk.

No room in root directory

[Label]

- *There is not enough room in the root directory for a volume label.*

Delete or move some of the files from the root directory to make room for the volume label.

No space left on device

[Backup][FC][Restore]

- *You cannot back up or restore any more files, and you cannot send any more output from a file comparison to your disk because the target disk is now full.*

You should probably delete some of the files on the disk to make more room.

No sub-directories exist

[Tree]

- *You have specified the /s switch, but the directory does not contain subdirectories.*

No such file or directory

[Backup][FC][Restore]

- *One or more of the files or directories that you specified does not exist.*

***** Not able to back up (or restore) file *****

[Backup]

- *This message may occur if there was an error in the source file or on the target disk.*

Use the **chkdsk** command on the source disk to see if you can determine the problem.

Not a graphics printer file

[Graphics]

- *The file you are printing does not contain graphics.*

Not enough memory

[Join][Share][Subst]

- *There is not enough memory for MS-DOS to run the command.*

Not enough room to merge the entire file

[Edlin]

- *There was not enough room in memory to hold the file during an Edlin T (transfer) command.*

You must free some memory by writing some files to a disk or by deleting some files before transferring this file.

Not found

[Edlin]

- *You specified an Edlin S (search) or R (replace) command that was unable to find a further occurrence of the specified search or replace string.*

Not ready error reading (or writing) drive x:

[MS-DOS]

- *This is a device error. See Appendix D, "Disk and Device Errors."*

O.K.?

[Edlin]

- *This prompt occurs during Edlin S (search) or R (replace) command processing.*

If you press any key except Y (for Yes) or the RETURN key, the search or replace process continues.

Out of environment space

[Command][MS-DOS]

- *There is not enough room in the program environment to accept more data.*

To increase the size of the existing environment, use the /e switch with the **command** command or remove some of the existing environment variables by using the **set** command.

Parameters not compatible

[Format][Replace]

- *You have specified switches that cannot be used together.*

Parameters not compatible with fixed disk

[Format]

- *You have used a switch that is not compatible with the specified drive.*

Parameters not supported

[MS-DOS][Format]

- *You have specified parameters that MS-DOS does not support.*

Parameters not supported by Drive

[Format]

- *Format displays this message when the device driver for this drive does not support Generic IOCTL function requests.*

Path(name) too long

[Print][Replace][Xcopy]

- *The pathname you specified was too long.*

You may have to change directories to use this command with files in deep subdirectories.

Path not found

[Chkdsk][Replace][Subst][Xcopy]

- *You specified an invalid pathname.*

Press any key to begin adding (replacing) file(s)

[Replace]

- *When you specify the /w switch, Replace displays this message to prompt you to start replacing files.*

Press any key to begin formatting x:

[Format]

- *This prompt is issued before you format a disk.*

Press any key to begin the format process. Or, if you wish to end this command, press CONTROL-C.

Press any key to begin recovery of the**n file(s) on drive x:**

[Recover]

- *This prompt is issued before you recover a disk or file.*

Press any key to begin the recovery. Recovered files are named *filennnn.rec*. If you wish to end this command, press CONTROL-C.

Press any key when ready . . .

[Diskcomp][Diskcopy]

- *This prompt gives you time to insert the appropriate disks before copying them.*

When you have inserted the disks into the appropriate drives, press any key to begin the **diskcopy** process. Or, if you wish to end this command, press CONTROL-C.

Printer error

[Mode]

- *The printer is off, or is not ready.*

Printer lines per inch set

[Mode]

- *Mode has set the number of lines per inch for the printer.*

PRINT queue is empty

[Print]

- *There are no files waiting to be printed.*

PRINT queue is full

[Print]

- *There is only room for 10 files in the list of files waiting to be printed.*

You can make room for more by using the **print /q** switch.
The limit is 32 files.

Probable non-DOS disk

Continue (Y/N)?

[Chkdsk]

- *The disk you are using is not recognized by this version of MS-DOS. The disk was either created by another system with a format that is not supported on this version of MS-DOS, or it is not an MS-DOS disk.*

Do not continue processing if **chkdsk** returns this message for a floppy disk. If this message returns for a hard disk, the information describing the characteristics of the disk to MS-DOS has been destroyed. In this case, you may continue **chkdsk** processing by typing Y (for Yes). This error may mean that the File Allocation Table (FAT) is bad and that the disk is unusable.

Processing cannot continue

[Chkdsk]

- *There is not enough memory in your machine to run Chkdsk for this disk.*

You must obtain more memory to run **chkdsk**.

Program too big to fit in memory

[MS-DOS]

- *You need more memory to run your application. It is possible that some programs you have run are still using some memory.*

You may try to restart MS-DOS; however, if you still receive this message, you still need more memory.

Read error in filename

[Edlin][Find]

- *MS-DOS could not read the entire file.*

Read fault error reading drive x:

[MS-DOS]

- *This is a device error. See Appendix D, "Disk and Device Errors."*

Reading source file(s)...

[Xcopy]

- *Xcopy is now reading the source files that you specified.*

Reinsert diskette for drive x:

[Format]

- *Reinsert the disk being formatted in the indicated drive.*

Replace filename? (Y/N)

[Replace]

- *Replace displays this prompt if you specify the /w switch. Press Y if you want to replace the existing file, or N if you do not want to replace the file.*

Replacing filename

[Replace]

- *Replace displays this prompt to let you know that it is replacing this file that exists on your disk.*

Requested Screen Shift out of range

[Mode]

- *You cannot shift the display any farther.*

Resident part of PRINT installed

[Print]

- *This is the first message that MS-DOS displays when you issue the Print command. It means that to process the Print command with other processes, available memory has been reduced by several thousand bytes.*

Resident portion of MODE loaded

[Mode]

- *Part of the Mode program is now resident in memory, and available memory has been reduced by several thousand bytes.*

Restore file sequence error

[Restore]

- *You have restored files in the wrong order.*

You must insert the backup disks in the same order that they were backed up.

***** Restoring files from drive x: *****

Diskette: *n*

[Restore]

- *This message is displayed during the restore process.*

Resynch failed. Files are too different

[FC]

- *FC compares what can be loaded into memory. If no lines match in the portion of the files in the buffer space, FC displays this message.*

SECOND diskette bad or incompatible

[Diskcomp]

- *The second disk does not contain the same format as the first disk, or Diskcomp does not recognize the format of the second disk.*

You should run **chkdsk** to help you identify the problem.

Sector not found error reading (or writing) drive x:

[MS-DOS]

- *This is a device error. See Appendix D, "Disk and Device Errors."*

Sector size too large in file filename

[MS-DOS]

- *The specified device driver loaded by config.sys uses a sector size larger than that of any other device driver on the system.*
- You cannot run this device driver.

Seek error reading (or writing) drive x:

[MS-DOS]

- *This is a device error. See Appendix D, "Disk and Device Errors."*

SHARE already installed

[Share]

- *Share can only be installed once.*

Sharing violation reading drive x:

[MS-DOS]

- *This is a device error. See Appendix D, "Disk and Device Errors."*

SORT: Incorrect DOS version

[Sort]

- *Sort does not run on MS-DOS versions before 2.0.*

SORT: Insufficient disk space

[Sort]

- *The disk is full.*

SORT: Insufficient memory

[Sort]

- *There is not enough memory to run the Sort program.*

Source and target drives are the same

[Backup][Restore]

- *You specified the same drive for the source and target disks.*

Source disk is Non-removable

[Backup]

- *This is an informational message indicating that the source disk is a hard disk.*

Source does not contain backup files

[Restore]

- *You are attempting to restore files from a disk that does not contain backup files.*

Source is a floppy (or hard) disk

[Restore]

- *This is an informational message only.*

Source path required

[Replace]

- *You did not specify a source path for the Replace command.*

**Specified drive does not exist,
or is non-removable**

[Diskcomp][Diskcopy]

- *You cannot compare or copy hard disks with this command.*
You must specify the name of a valid floppy drive.

Specified MS-DOS search directory bad

[MS-DOS]

- *The Shell command in the config.sys file is incorrect.*
Make sure that the *command.com* file exists and that MS-DOS can find it.

Strike a key when ready...

[MS-DOS]

- *This prompt occurs during command processing and is always accompanied by another message. This message is also displayed if you have inserted a Pause command in a batch file. Usually, MS-DOS asks you to insert disks into appropriate drives before this prompt.*

To begin command processing, press any character, any number key, the SPACEBAR, or the RETURN key.

Syntax error

[Attrib][Find][MS-DOS]

- *You have entered a command incorrectly.*
Check to make sure you have typed the command correctly. Remember to enclose the **find** command string in double quotation marks.

System transferred

[Format][Sys]

- *The system files were transferred during Format or Sys command processing.*

Target cannot be used for backup

[Backup]

- *Either the target disk has an unrecognizable format, or it is bad.*
Do not use the disk, or try to format the disk with the **format** command, or run **chkdsk** on it to determine the problem.

Target disk is Non-removable

[Backup]

- *This is an informational message that the target disk is a hard disk.*

Target diskette is write protected

[Diskcopy]

- *The target disk either has a write-protect tab on it, or it does not have a write-protect notch.*
If you want to destroy any existing information on the disk, remove the write-protect tab and give the command again. If the disk does not have a write-protect notch, you cannot use it as a target disk.

Target diskette may be unusable

[Diskcopy]

- *Either the target disk has an unrecognizable format, or it is bad.*

Try to format the disk with the **format** command, or run **chkdsk** on it to determine the problem.

Target is a floppy (or hard) disk

[Backup]

- *This is an informational message only.*

Target is full

[Restore]

- *There is no more room on the target disk for restored files.*

You must delete some of the files on the disk to make room for these files, or use another disk.

Target is Non-Removable

[Restore]

- *This is an informational message only.*

Terminate batch job (Y/N)?

[MS-DOS]

- *If you press CONTROL-C while in batch mode, MS-DOS asks you whether or not you wish to end batch processing.*

Press Y (for Yes) to end processing, or N (for No) to continue.

The last file was not restored

[Restore]

- *There was not enough room on the target disk for the file, or the last file was bad.*

Use the **chkdsk** command to determine the problem.

Too many files open

[Edlin][Label]

- *MS-DOS could not open the .bak file or write the volume label due to the lack of available system file handles.*

Increase the value of the **files** command in the *config.sys* file.

Too many open files

[Backup][FC][Restore][Xcopy]

- *MS-DOS could not open the files that you want to compare due to the lack of available system file handles.*

Increase the value of the **files** command in the *config.sys* file.

Track 0 bad - disk unusable

[Format]

- *The Format command can accommodate defective sectors on the disk, except for those near the beginning.*
You must use another disk.

Unable to create directory

[Mkdir][Xcopy]

- *MS-DOS could not create the directory you specified.*
Check to see that there is not a name conflict. You may have a file with the same name, or the disk may be full.

Unable to erase

[Backup]

- *Backup could not erase the files on the target disk.*
Check to see that the files on the backup disk are not read-only, and that the disk is not write-protected.

Unable to shift Screen

[Mode]

- *Mode is unable to shift the test pattern on the screen any farther.*

Unexpected DOS Error n

[Replace]

- *An unexpected error n occurred, where n is the MS-DOS error number.*

Unrecognized command in CONFIG.SYS

[MS-DOS]

- *There is an invalid command in your config.sys file.*
Refer to Appendix B, "How to Configure Your System," for a list of valid statements.

Unrecognized printer

[Graphics]

- *You are using an invalid printer.*
Check to see whether you entered the command properly, or refer to Chapter 3, "MS-DOS Commands," to make sure that you have specified a valid printer name.

Unrecognized printer port

[Graphics]

- *The printer device name that you specified was invalid.*
You may need to set the printer port by using the **mode** command.

Unrecoverable error in directory

Convert directory to file (Y/N)?

[Chkdsk]

- *This message is displayed if Chkdsk is unable to correct an error in a directory.*

If you respond *Y* (for Yes) to this prompt, **chkdsk** converts the bad directory into a file. You can then fix the directory or delete it. If you respond *N* (for No) to this prompt, you may not be able to write to or read from the bad directory.

Unrecoverable read (or write) error on drive x:

[MS-DOS]

- *This is a device error. See Appendix D, "Disk and Device Errors."*

usage: **fc** [/a] [/b] [/c] [/l] [/lb n] [/w]

[t] [/n] [/NNNN] file1 file2

[FC]

- *One of the switches that you have specified is invalid.*

VERIFY is off (or on)

[MS-DOS]

- *This message tells you the current setting of the Verify command.*

Volume in drive x: has no label

[Dir][Label][Vol]

- *This is an informational message displayed in response to the Dir, Label, or Vol command.*

Volume in drive x: is filename

[Dir][Label][Vol]

- *This is an informational message displayed in response to the Dir, Label, or Vol command.*

Volume label (11 characters, ENTER for none)?

[Format][Label]

- *This message is displayed when you specify the Label command, or the /v switch in the Format command.*

Type a volume label, or press the RETURN key to indicate that you do not want a volume label for this disk. It's a good idea, though, to specify a volume label to help you identify your disks.

W

**WARNING, ALL DATA ON NON-REMOVABLE DISK
DRIVE x: WILL BE LOST!
Proceed with Format (Y/N)?**

[Format]

- *This message appears when you try to format a hard disk that already contains data.*

If you press *Y* (for Yes) the data on the disk will be erased.

If you do not want the files on your hard disk erased, press *N* (for No). Copy the files to a floppy disk and repeat the **format** command.

Warning - directory full

[Recover]

- *The root directory is too full for Recover processing.*

Delete some files in the root directory to free space for the recovered files, and try the command again.

**Warning! Diskette is out of sequence
Replace diskette or continue if okay
Strike any key when ready**

[Restore]

- *You should restore the diskettes in the order that you backed them up.*

**Warning! File filename
is a hidden (or read-only) file
Replace the file (Y/N)?**

[Restore]

- *This message prompts you as to whether you want to replace a hidden or read-only file.*

Type *Y* (for Yes) if you want to restore the hidden or read-only file from the backup disk. Type *N* (for No) if you do not want to restore this file.

**Warning! File filename
was changed after it was backed up
Replace the file (Y/N)?**

[Restore]

- *This message prompts you as to whether you want to replace a backup file that has been changed.*

Type *Y* (for Yes) if you want to restore this file, or *N* (for No) if you do not.

**Warning! Files in the target drive
\\BACKUP (or root) directory will be erased**

[Backup]

- *Backup found files in the target drive, and you did not specify the /a switch to append files.*

Warning! No files were found to back up

[Backup]

- *Backup did not find any files to back up on the disk you specified.*

Warning! No files were found to restore

[Restore]

- *Restore did not find the file that you wanted to restore from the backup disk.*

Warning: Read error in EXE file

[Exe2bin]

- *The amount read was less than the size of the header. This is a warning message only.*

Write fault error writing drive x:

[MS-DOS]

- *This is a device error. See Appendix D, "Disk and Device Errors."*

Write protect error writing drive x:

[MS-DOS]

- *This is a device error. See Appendix D, "Disk and Device Errors."*



Appendix F

Configuring Your Hard Disk (Fdisk)

Introduction

Hard disks can be divided into one to four separate sections, called partitions. Partitions separate your hard disk into individual areas, and each partition may contain a different operating system.

To prepare your hard disk for the MS-DOS operating system, you must create a partition for MS-DOS, called a DOS partition. You can create a DOS partition on your hard disk by using a menu-driven utility called **fdisk**. You must use **fdisk** if you want to:

- prepare your hard disk for MS-DOS
- create a DOS partition
- change the active partition
- delete a DOS partition
- display partition information
- choose the next hard disk on your computer

If you want to use your entire hard disk for MS-DOS, you will use the **fdisk** program only once to create the DOS partition.

Note Many computer stores configure hard disk computers for MS-DOS, so you may not need to use **fdisk**. They may also format your hard disk to start MS-DOS when you turn the power on. To find out whether this has been done, try to start MS-DOS from your hard disk, or run **fdisk** and try to create a DOS partition. **Fdisk** displays a message if your hard disk already has a DOS partition.

Starting Fdisk

How to Start Fdisk

To start **fdisk**, you simply type the following command and press the RETURN key:

```
fdisk
```

In response, **fdisk** displays its main menu on your screen:

```
Fixed Disk Setup Program Version 0.01  
(C)Copyright Microsoft, 1985, 1986.
```

FDISK Options

Choose one of the following:

1. Create DOS Partition
2. Change Active Partition
3. Delete DOS Partition
4. Display Partition Data
5. Select Next Fixed Disk Drive

Enter choice:[1]

Press ESC to return to DOS

If your computer has only one hard disk, choice 5 will not appear on this menu.

The following sections contain a description of each of these options, and show the screens that they display. To return to MS-DOS from the main menu, just press the ESC key. You can also use the ESC key to return to the main menu from any of the other **fdisk** screens.

Each **fdisk** screen displays a default value. To choose the default value, press RETURN. To choose another value, just type the value you want, and then press the RETURN key.

Choice 1 — Creating a DOS Partition

If you choose the first option on the main menu, and if your hard disk already contains a DOS partition, **fdisk** displays a screen showing partition status information for your hard disk. For example, your screen might look something like this:

Create DOS Partition

Partition	Status	Type	Start	End	Size
1	A	DOS	0	304	305

Total disk space is 305 cylinders

Fixed disk already has a DOS partition

Press ESC to return to FDISK Options

If your hard disk does not contain a DOS partition, **fdisk** displays the following screen:

Create DOS partition

Current Fixed Disk Drive: 1

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

If you want to use the entire hard disk for MS-DOS, press the RETURN key to accept the default setting, Y. **Fdisk** then displays the following message:

System will now restart
Insert DOS diskette in drive A:
Press any key when ready . . .

You should insert your MS-DOS disk in drive A and press any key to restart MS-DOS.

Now that you have created your DOS partition, you must format your hard disk so that MS-DOS can use it. If you want to start MS-DOS from your hard disk, remember to use the **format /s** switch. For more information on how to format your hard disk, see the **format** command in Chapter 3, "MS-DOS Commands."

Using Part of Your Hard Disk for MS-DOS

If you want to use only part of your hard disk for MS-DOS, press *N*, then press RETURN. **Fdisk** displays the following message in response:

```
Total fixed disk space is xxxxx cylinders
Maximum available space is xxxxx
cylinders at xxxxx
Enter partition size.....: [xxxx]
```

This message shows the total number of cylinders available for a hard disk partition, and prompts you to enter the size of your new partition. The default for this size is the maximum available space on the hard disk. Press the RETURN key if you want the default; otherwise, type the size (in cylinders) that you want for the partition, and press the RETURN key. **Fdisk** then displays the following message:

```
Enter starting cylinder number...:[xxxx]
```

Note If it finds any defective tracks at the start of the partition, **fdisk** adjusts the partition boundaries to avoid those bad tracks.

The default starting cylinder number starts at the first cylinder large enough to contain the partition. If you want to locate the DOS partition at the default location, press the RETURN key. Otherwise, type the number of the cylinder where you want the DOS partition, and press the RETURN key.

If you want to start MS-DOS from this partition on your hard disk, remember to use the **format /s** switch. For more information on formatting your hard disk, see the **format** command in Chapter 3, "MS-DOS Commands."

Note You can have only one DOS partition on your hard disk at a time. To create a partition for another operating system, you must refer to the documentation for that operating system.

Choice 2 — Changing the Active Partition

If you choose the second option on the main menu, **fdisk** displays a screen showing the status of each partition on your hard disk. The active partition, indicated with a status of *A*, is the partition whose operating system and files you access when you turn on the power or reset your computer. Only one partition is active at a time; the others are non-active, indicated with a status of *N*.

For example, your Change Active Partition screen might look like this:

Change Active Partition

Current Fixed Disk Drive: 1

Partition	Status	Type	Start	End	Size
1	A	DOS	0	119	120
2	N	non-DOS	120	304	185

Total disk space is 305 cylinders

Enter the number of the partition you want to make active.....: [1]

Press ESC to return to FDISK Options

Type the number of the partition that you want to activate, and press the RETURN key. The default setting is the active partition number.

If you are using the entire hard disk as a DOS partition, **fdisk** displays the following message:

Partition 1 is already active

Press Esc to return to FDISK Options

instead of prompting you for the partition that you want to activate. Press the ESC key to return to the main menu.

Choice 3 — Deleting a DOS Partition

If you choose the third option on the main menu, **fdisk** displays a screen showing the status of each partition on your hard disk, and prompts you to delete the DOS partition. When you delete a DOS partition, **fdisk** deletes the partition boundaries and any data that existed in that partition. Once you delete the partition, *you cannot recover that data.*

Note You cannot use **fdisk** to delete a non-DOS partition. Instead, to continue using MS-DOS after you have deleted the DOS partition, you must put an MS-DOS system disk into drive A. If you want to start a different operating system in another partition of your hard disk, you must change the active partition to that number *before* you delete the DOS partition.

Your Delete DOS Partition screen might look like this:

Delete DOS Partition

Current Fixed Disk Drive: 1

Partition	Status	Type	Start	End	Size
1	N	DOS	0	304	305

Total fixed disk space is 305 cylinders

Warning! Data in the DOS partition
will be lost. Do you wish to
continue.....? [N]

Press ESC to return to FDISK Options

If you do not want to delete the DOS partition, press the RETURN key to accept the default value, *N*. To delete the DOS partition, type *Y*, and then press the RETURN key.

Choice 4 — Displaying Partition Data

If you choose the fourth option on the main menu, **fdisk** displays a screen that contains information about each of the partitions on your hard disk.

For example, your Display Partition Information screen might look like this:

Display Partition Information

Partition	Status	Type	Start	End	Size
1	N	DOS	0	304	305

Total fixed disk space is 305 cylinders

Press ESC to return to FDISK Options

This screen gives you information about the partition, including its number, status, and type, its starting and ending cylinder numbers, and its size in cylinders. Press the ESC key to return to the main menu.

Choice 5 — Next Fixed Disk

This option appears on the **fdisk** main menu only if you have more than one hard disk attached to your computer. If you choose this option, **fdisk** changes the current disk drive to the next drive number.

For example, if the current disk drive is number 1, and you choose option 5 on the main menu, **fdisk** changes the current disk drive to number 2. You could then choose any of the **fdisk** options to prepare the second fixed disk for MS-DOS.



Index

\$ (dollar sign) prompt 75
> (greater-than sign) prompt 75
< (less-than sign) prompt 75
| (pipe symbol) prompt 75

Abbreviation

/cparmaxalloc option, Link 198

Debug memory locations

BYTE 213

WORD 213

/dosseg option, Link 202

/dsallocate option, Link 200

/help option, Link 193

/high option, Link 199

/linenumbers option, Link 195

/map option, Link 194

/nodefaultlibrarysearch option, Link 197

/noignorecase option, Link 196

/overlayinterrupt option, Link 201

/pause option, Link 193

/segments option, Link 202

/stack option, Link 197

Aborting a command 123

Action messages

reading 269

writing 269

Adding lines, Edlin 144

Adding spacing to batch files 115

Align type, Link 203

Aligning the screen 69

Allocating a data group, Link 199

Allocating disk space 2, 51

Alphabetical sorting 17, 107

Alphabetizing a file 88

ANSI escape sequence

CUB 259

CUD 258

CUF 258

CUP 258

CUU 258

ANSI escape sequence (*continued*)

definition 257

DSR 259

ED 259

EL 259

ESC[2J 36

HVP 258

modes of operation 260

note 257

RCP 259

RM 262

SCP 259

SGR 260

using in Config.sys file 250

ANSI escape sequence, driver 76

Append command

purpose 24

syntax 24

Append command, Edlin

purpose 144

syntax 144

Appending output 16

Argument 21

ASCII characters 216

ASCII code (escape) prompt 75

ASCII file 40

Assemble command, Debug

abbreviation 213

comments 213

purpose 213

syntax 213

Assign command

purpose 26

syntax 26

Asterisk (*)

as system prompt in Edlin 126

wildcard character 8

Asynchronous communications mode 68

Attrib command

purpose 27

syntax 27

Attrib command (*continued*)

- using wildcards with 27

Attribute, setting 27

Autoexec.bat file 16, 44, 103, 105

Autoexec.bat options

- BASIC** 105

- clearing the screen 105

- Date** command 105

- default directory 105

- default drive 105

- detaching swapper 105

- Path** command 105

- prompt 105

- size of memory partitions 105

- Time** command 105

Background color functions, ANSI 261

Backslash (\) 6

Backup command

- drive option 29

- errorlevel* 83

- purpose 29

- switch

- /a 30

- /d 30

- /L 30

- /m 30

- /s 30

- /t 30

- syntax 29

Backup disk 2

Bad call format error 267

Bad command error 267

Bad sectors 77

Bad unit error 267

.Bak extension 127, 162

Bar, vertical. *See* Pipe symbol (|)

BASIC 105

.Bat extension 14, 101, 106

Batch command

- Echo** 109

- For** 110

- Goto** 112

- If** 113

- Pause** 114

- Rem** 115

- Shift** 116

Batch file

- chain 103

Batch file (*continued*)

- comment 115

- creating 101

- definition 101

- dividing into pieces 114

- label 112

- processing 85

- readability and spacing 115

- remark 115

- replaceable parameters 106

- running 102, 107

- Sorter.bat** 106, 107

- stopping 114

- suspending execution of 114

- using a colon in 112

- using a percent sign in 106, 111

- using redirection symbols in 103

- using temporary files with 108

- using the Shift command with 116

Batch processing 85, 101, 110

Batch processing commands 108

Batch program, halting 114

Baud rate 68

Bin directory 5, 15

Binary file

- copying 40

- displaying 93

Blink function, ANSI 261

Block of text, moving, Edlin 159

Bold function, ANSI 261

Brackets, used for options 21

Break command

- purpose 32

- syntax 32

Break command, **Config.sys** file

- definition 246

- purpose 247

- syntax 247

Buffer 172

Buffer, internal 73

Buffers command, **Config.sys** file

- definition 246

- purpose 248

- syntax 248

Byte 211

Bytes, available 34

Capital letter, use of 156

Case-sensitivity, preserving 196

- Chains, batch file 103
- Changing directories 10
- Changing the name of a file 78
- Chdir command
 - changing the working directory 10
 - purpose 33
 - shorthand notation 11
 - syntax 33
- Chkdsk command
 - checking files for errors 3
 - errors found 3
 - purpose 34
 - status report 3, 34
 - switch
 - /f 34
 - /v 34
 - syntax 34
- Class type, Link 204
- Clearing the screen 36
- Cls command
 - purpose 36
 - syntax 36
- Colon, use of 21
- Colon, using in a batch file 112
- .Com extension 14, 208
- COM1, communications port 68
- COM2, communications port 68
- Combine type, Link
 - at 204
 - common 205
 - memory 205
 - private 205
 - public 204
 - stack 204
- Combining files 41
- Combining segments 204
- Command
 - Append 24
 - Assign 26 *See also* Subst
 - Attrib 27
 - Backup 29, 83
 - batch processing 108
 - Break 32
 - Break, Config.sys file 246, 247
 - Buffers, Config.sys file 246, 248
 - Cd 10
 - Chdir 33
 - Chkdsk
 - checking files for errors 3
 - errors found 3

- Command (*continued*)
- Chkdsk (*continued*)
 - status report 3
- Cls 36
- Command 37
- commands not usable over network
 - Chkdsk 20
 - Diskcomp 20
 - Diskcopy 20
 - Format 20
 - Label 20
 - Recover 20
 - Subst 20
 - Sys 20
- conditional execution 113
- Copy 39
- Country, Config.sys file 246
- Date 44
- Debug commands 210
- default values 20
- Del 9, 14, 45
- Device, Config.sys file 246, 250
- Dir 6, 11, 14, 46
- Diskcomp 48
- Diskcopy 14, 51
- diskcopy.exe 14
- Drivparm, Config.sys file 246
- Echo 109
- Exe2bin 53
- Exit 55
- external 13, 14, 15
- external symbol 19
- FCBS, Config.sys file 246, 253
- Fdisk 56
- Files, Config.sys file 246, 254
- Find 17, 57
- For 110
- Format 3, 14, 58
- format.bat 14
- Goto 112
- Graftabl 61
- Graphics 62
- If 113
- internal 13
- internal symbol 19
- Join 64
- Keybdv 65
- Keybfr 65
- Keybgr 65
- Keybit 65

Command (*continued*)

- Keybsp 65
- Keybuk 65
- Label 66
- list of 22
- Mkdir 10, 67
- Mode 68
- More 17, 71
- Path 15, 72
- Pause 114
- pipeline 18
- piping 17
- Print 73
- Prompt 75
- Recover 77
- Rem 115
- Ren 78
- repeating 117
- Restore 79, 82
- Rmdir 11, 84
- rules for using options
 - brackets 21
 - ellipses 21
 - equal signs 21
 - italics 21
 - semicolons 21
 - spaces and tabs 21
- Set 85
- Share 87
- Shell, Config.sys file 246, 255
- Shift 116
- Sort 17, 88
- Subst 89
- symbol
 - external 19
 - internal 19
 - non-network 20
- syntax 20
- Sys 90
- Time 91
- Tree 92
- Type 14, 93
- Ver 94
- Verify 95
- Vol 96
- Xcopy 97
- Command command
 - purpose 37
 - switch
 - /c 37

Command command (*continued*)

- switch (*continued*)
 - /p 37
 - syntax 37
- Command information, argument 21
- Command line
 - parameters, changing position of 116
 - relation to template 118
- Command lines, Link 184
- Command options, Edlin 142
- Command parameter, Debug
 - address 211
 - byte 211
 - description 210
 - drive: 211
 - list 212
 - range 211
 - record 211
 - string 212
 - value 211
- Command processor, MS-DOS 37, 55
- Command.com 13, 55, 113, 246, 256
- Comment, batch file 115
- Communications port
 - COM1 68
 - COM2 68
- Compare command, Debug
 - abbreviation 215
 - comments 215
 - purpose 215
 - syntax 215
- Comparing 171
- Compilers 195
- Compiler overlays 185, 200
- Computer memory 13
- Computer network 19
- Concatenation 41
- Concealed function, ANSI 261
- Condition parameter 113
- Conditional execution of commands 113
- Config.sys command
 - Break 246-7
 - Buffers 246, 248
 - Country 246, 249
 - Device 246, 250
 - Drivparm 246
 - FCBS 246, 253
 - Files 246, 254
 - Shell 246, 255
- Config.sys file 245

- Config.sys, example file 256
- Configuration File. *See* Config.sys file
- Configuring a hard disk 56
- Configuring your system 245
- Console 102
- Control character functions 122
- Control characters, using in Edlin 141
- Control key sequence
 - stop a command, CONTROL-C 122, 207
 - stop screen scrolling, CONTROL-S 122, 207
- CONTROL-ALT-F1 65
- CONTROL-ALT-F2 65
- CONTROL-BREAK 68
- CONTROL-C 102, 114, 122, 123, 141, 153, 207, 247
- CONTROL-C check 32, 247
- CONTROL-H 123
- CONTROL-J 123
- CONTROL-N 123
- CONTROL-P 123
- CONTROL-S 122, 123, 207
- CONTROL-V 141
- CONTROL-X 123
- CONTROL-Z 40, 102, 105, 106, 140, 163
- Copy command
 - option
 - drive 39
 - pathname 39
 - purpose 39
 - switch
 - /a 40
 - /b 40
 - /v 40
 - syntax 39
 - using wildcards with 41
- Copy command, Edlin
 - purpose 145
 - syntax 145
- Copying
 - disks 51
 - File Allocation Table 3
 - files 39
 - lines, Edlin 145
 - template characters 122
- Country command, Config.sys file
 - changing time format 91
 - definition 246
 - purpose 249
 - syntax 249
- /cparmaxalloc option, abbreviation 198
- /cparmaxalloc option, Link 198
- Creating a directory 10, 67
- CUB-Cursor Backward, ANSI escape sequence 259
- CUD-Cursor Down, ANSI escape sequence 258
- CUF-Cursor Forward, ANSI escape sequence 258
- CUP-Cursor Position, ANSI escape sequence 258
- Cursor
 - function 258
 - in the template 120
 - location, Edlin 129-135, 137
 - movement 257
- CUU-Cursor Up, ANSI escape sequence 258
- Data error 267
- Data group, allocating in Link 199
- Data path, NUL 24
- Data, losing 269
- Databit 68
- Date command
 - purpose 44
 - syntax 44
- Date message 44
- Date prompt 75
- Debug
 - arglist 209
 - command parameters
 - address 211
 - byte 211
 - description 210
 - drive: 211
 - list 212
 - range 211
 - record 211
 - string 212
 - value 211
 - commands 210
 - error indication 209
 - error messages 241
 - hyphen prompt 208
 - illegal ranges 212
 - illegal strings 212
 - memory locations, abbreviating 213
 - mnemonics 213
 - opcodes 214
 - quitting 231
 - starting 208
- Debug command
 - Assemble 213

Debug command (*continued*)

- Compare 215
- Dump 216
- Enter 218
- Fill 220
- Go 221
- Hex 223
- Input 224
- Load 225
- Move 227
- Name 228
- Output 230
- Quit 231
- Register 232
- Search 235
- Trace 236
- Unassemble 237
- Write 239

Debug errors

- BF - Bad flag 241
- BP - Too many breakpoints 241
- BR - Bad register 241
- DF - Double flag 241

Debugging programs 207

Default drive 20, 243

Default drive prompt 75

Default filename extension, Link 182

Default libraries, ignoring 197

Default printer, PRN 73

Default values, commands 20

Del command 9

- purpose 45
- syntax 45

DEL command, Edlin

- purpose 132
- similarity to F1 132
- syntax 132

DEL key 118, 128

Delete command, Edlin

- purpose 147
- syntax 147

Deleting

- a directory 11, 84
- files 12, 45
- lines, Edlin 147
- template characters 122
- text, Edlin 125

Destination file. *See* Target file

Device command, Config.sys file

- definition 246

Device command, Config.sys file (*continued*)

- purpose 250
- syntax 250

Device error

- Bad call format error 267
- Bad command error 267
- Bad unit error 267
- Not ready error 268
- Read fault error 268
- Sector not found error 268
- Seek error 269
- Sharing violation 269

Device error messages 267

Device, operation mode 68

Dir command

- purpose 46
- switch
 - /p 46
 - /w 46
- syntax 46
- using wildcards with 46

Directives, Link 203

Directory

- \bin 15
- changing 10, 33
- creating 10
- deleting 11
- displaying 11, 46
- erasing 11
- external commands, \bin 5
- hierarchical structure 3, 5
- listing of 6
- making 67
- maximum capacity 3
- multilevel 3, 5, 6
- removing 11, 84
- renaming 12
- root 3, 4, 5, 64
- shorthand notation 6
- sorted listing 88
- sorting 18
- subdirectories 3, 5, 11
- system 4
- user 5
- working 4, 5, 10, 15, 24

Disk

- backing up 29
- checking 34
- copying 51
- formatting 58

Disk (continued)

- fragmented 51
- media 29
- sector 77
- target 82, 90
- types 29, 82
- volume label 59, 96
- write-protected 71
- writing to 95
- Disk buffer
 - default number of 248
 - definition 248
- Disk drive 20
 - if you have only one 22, 243
 - source 21
 - target 21
 - virtual 89
- Disk error
 - Abort 269
 - Data error 267
 - FCB unavailable 267
 - General failure error 267
 - Ignore 269
 - Invalid disk change error 268
 - Lock violation error 268
 - No paper error error 268
 - Non-DOS disk error 268
 - Retry 269
 - Write fault error 269
 - Write protect error 269
- Disk error messages 267
- Disk space, allocation of 51
- Disk swapping 193
- Diskcomp command
 - purpose 48
 - syntax 48
- Diskcopy command
 - option
 - source drive 51
 - target drive 51
 - purpose 51
 - syntax 51
- Display modes 68
- Displaying a command, MS-DOS 109
- Displaying a directory 46
- Displaying a file 17, 93
- Displaying lines, Edlin 156
- Dividing a batch file into pieces 114
- .Doc extension 41, 78
- /dosseg option, abbreviation 202

- /dosseg option, Link 202

- Drive
 - source 21
 - target 21
- Drive letter 89, 96
- Drive name 20, 82
- Driver.sys 257
- Drvparm command, Config.sys file 246, 251
- /dsallocate option, abbreviation 200
- /dsallocate option, Link 191, 199
- DSR-Device Status Report, ANSI escape sequence 259
- Dummy parameter. *See* Replaceable parameter
- Dump command, Debug
 - abbreviation 216
 - comments 216
 - purpose 216
 - syntax 216

Echo command, batch

- purpose 109
- syntax 109

- Echoing a command. *See* Displaying a command

- ED-Erase Display, ANSI escape sequence 259

Edit command, Edlin

- purpose 150
- syntax 150

- Editing a file 125

Editing commands

- copy multiple characters 130
- copy one character 129
- copy template 131
- enter insert mode 135
- new template 137
- quit 134
- skip multiple characters 133
- skip one character 132

Editing key

- DEL 119
- description 128
- ESC 119
- F1 119
- F2 119
- F3 119
- F4 119
- F5 119
- F6 119
- INS 119
- using with Edlin 128

Editing keys 117, 128

Editing text, Edlin 150

Edlin

adding lines 144

adjusting lines 125

asterisk (*) prompt 126

.bak extension 127

changing your Config.sys file with 245

command option

line 142

question mark 142

text 143

commands 139

consecutive line numbering 125

copying lines 145

creating a new file 126

current line, asterisk (*) 140

cursor location 129–135, 137

default values 156

deleting lines 125, 147

displaying lines 125, 140

editing an existing file 126

editing text 125, 150

freeing memory 127

insert mode 129, 130, 131, 134, 135, 137

inserting text 125, 153

line numbers 125

listing text 156

loading files into memory 126, 170

memory capacity 170

merging files 169

moving lines 159

naming files 126

omitting options 156

paging through files 161

question mark option 166

quitting 127, 162

renumbering lines 125

replace mode 135

replacing text 163

revising 125

saving a file 127, 152

searching for text 166

starting 126

using commands with 140

writing to a disk 170

Edlin command

Append lines 144

Copy lines 145

Delete lines 147

Edlin command (*continued*)

Edit line 150

End editing 152

Insert text 153

List text 156

Move text 159

Page 161

Quit 162

Replace text 163

Search text 166

Transfer text 169

Write lines 170

EL-Erase Line, ANSI escape sequence 259

Ellipsis, used for options 21

End-of-file mark 40

Ending an edit session, Edlin 152

Enter command, Debug

abbreviation 218

comments 218

purpose 218

syntax 218

Environment strings 86, 103

Environment variable, LIB 189

Environment, definition 85

Equal sign, used for options 21

Erase 45

Erasing

files 12

the screen 259

Error checking

directories 3

disks 3

File Allocation Table 3

files 3

Error indication, Debug 209

Error message

action 267

device 267

disk 267

type 267

Error message, network 19

Error messages

Debug 241

MS-DOS 271

Error, responding to

Abort 270

Ignore 270

Retry 270

ESC key 118, 128

- ESC command, Edlin
 - purpose 134
 - similarity to F5 134
 - syntax 134
- Escape sequence, ANSI 36, 76, 250, 257, 258, 259
- Example Config.sys file 256
- .Exe extension 14
- .Exe file 14
- Exe2bin command
 - purpose 53
 - syntax 53
- /exepack option, Link 194
- Executable file
 - creating 181
 - .exe 14
- Executable file packing, Link 194
- Executable image 203
- Executable program, Link 184
- Executing a .bat file 107
- Exit code 113
- Exit command
 - purpose 55
 - syntax 55
- Extension
 - .bak 127, 162
 - .bat 14, 101, 106
 - .com 14, 208
 - .doc 41, 78
 - .exe 14, 182, 208, 226
 - .lib 183
 - .map 183
 - .obj 182
 - .ref 41
 - .txt 8, 9, 41, 78
 - substitution 41
- External command
 - \bin directory 5
 - definition 14
 - searching for 72
 - symbol 19
- F1 key 118, 128
- F1 command, Edlin
 - purpose 129
 - similarity to DEL 129
 - syntax 129
- F2 key 118, 128
- F2 command, Edlin
 - purpose 130
 - similarity to F4 130
 - syntax 130
- F3 key 118, 128
- F3 command, Edlin
 - purpose 131
 - syntax 131
- F4 key 118, 128
- F4 command, Edlin
 - purpose 133
 - similarity to F2 133
 - syntax 133
- F5 key 118, 128
- F5 command, Edlin
 - purpose 137
 - similarity to ESC 137
 - syntax 137
- F6 key 118
- Faint function, ANSI 261
- FAT. *See* File Allocation Table
- FC 171, 173
- FCB unavailable 267
- FCBS command, Config.sys file
 - definition 246
 - purpose 253
 - syntax 253
- Fdisk command
 - menus 56, 313
 - purpose 56
 - syntax 56
- File
 - alphabetizing 88
 - ASCII 40
 - attribute 27
 - Autoexec.bat 16, 44, 103, 105
 - backup 162
 - Backup.log 30
 - batch 85, 106
 - binary 40
 - Command.com 13
 - concatenating 41
 - Config.sys 91, 245
 - copying 39
 - creating a new file, Edlin 126
 - deleting 45
 - deleting a temporary file 108
 - destination. *See* Target
 - displaying 17, 93
 - editing 125

File (*continued*)

- editing an existing file, Edlin 126
- grouping 3
- hidden 34, 82
- location of 2
- organization 3, 6
- printing 73
- protection 2
- read-only 82
- recovery 77
- renaming 78
- searching for 4
- separation by category 3
- Sorter.bat 106
- sorting 17, 88
- source 39
- system 3
- temporary 71, 106, 108
- text 125
- updating 125

File Allocation Table

- allocating disk space 2
- copying of 2
- definition 2
- error message, drive 270
- locating files 2

File Control Blocks. *See* FCBS

File header, MS-DOS 192

File restoring 82

Filename

- as a command option 20
- Autoexec.bat 16
- Command.com 13
- syntax 20

Filename extension

- .bak 127, 162
- .bat 14
- .com 14, 208
- .doc 41, 78
- .exe 14, 182, 208, 226
- .lib 183
- .map 183
- .obj 182
- .ref 41
- .txt 8, 9
- when to add 15

Files, comparing 171

Files command, Config.sys file

- definition 246
- purpose 254

Files command, Config.sys file (*continued*)

- syntax 254

Fill command, Debug

- abbreviation 220
- comments 220
- purpose 220
- syntax 220

Filter

- definition 17
- Find 17
- More 17, 71
- Sort 17, 88

Find command

- purpose 57
- switch
 - /c 57
 - /n 57
 - /v 57
- syntax 57

Finding a file 4

Fixups, Link 205

Flags 233

Floppy disk drive, if you have only one 243

For command, batch

- purpose 110
- syntax 110

Foreground color functions, ANSI 261

Format command

- purpose 58
- switch
 - /s 59
 - /v 59
- syntax 58

Formatting a disk 58

Fragmented disk 51

Frame number, canonical 203

Freeing memory, Edlin 127

Function key

- DEL 119
- description 128
- ESC 119
- F1 119
- F2 119
- F3 119
- F4 119
- F5 119
- F6 119
- INS 119

Functions, special editing 119

- General failure error 267
- Go command, Debug
 - comments 221
 - purpose 221
 - syntax 221
- Goto command, batch
 - purpose 112
 - syntax 112
- Graftabl command
 - purpose 61
 - syntax 61
- Graphic functions, changing 257
- Graphics command
 - purpose 62
 - syntax 62
- Graphics monitor
 - color disabled (BW) 69
 - color enabled (CO) 69
- Graphics, ANSI escape sequences
 - function 260
 - Background colors 261
 - Blink 261
 - Bold 261
 - Concealed 261
 - Faint 261
 - Foreground colors 261
 - Italic 261
 - Reverse Video 261
 - parameter 260
 - Set Graphics Rendition-SGR 260
- Grouping files 3
- Groups, linking 205
- Halting a command 123
- Hard disk
 - assigning drive letters to 26
 - configuration 56
- /help option, abbreviation 193
- /help option, Link 193
- Hex command, Debug
 - abbreviation 223
 - comments 223
 - purpose 223
 - syntax 223
- Hidden file 34, 90
- Hidden file, restoring 82
- Hierarchical directory, definition 3, 5
- /high option, abbreviation 199
- /high option, Link 191, 199
- High start address, setting 199
- Hint, Autoexec.bat 103
- HVP-Horizontal and Vertical Position, ANSI
 - escape sequence 258
- Hyphen prompt, Debug 208
- If command, batch
 - purpose 113
 - syntax 113
- Illegal ranges, Debug 212
- Illegal strings, Debug 212
- Input 16
- Input command, Debug
 - abbreviation 224
 - comments 224
 - purpose 224
 - syntax 224
- Input, redirecting 17
- INS key 118, 128
- INS command, Edlin
 - purpose 135
 - syntax 135
- Insert command, Edlin
 - purpose 153
 - syntax 153
- Insert mode, Edlin 129 – 131, 134, 135, 137
- Inserting template characters 122
- Inserting text, Edlin 125, 153
- Interactive processing 110
- Internal buffer 73
- Internal command 13
 - definition 13
 - symbol 19
 - syntax. *See* Command syntax
- International
 - case conversion 249
 - currency 249
 - date 249
 - time 249
- Invalid disk change error 268
- Inverse video mode 76
- Io.sys file 90
- Italic function, ANSI 261
- Italics, used for options 21
- Join command
 - purpose 64
 - syntax 64

Joining files 41

Key

- DEL 119
- editing 117
- ESC 119
- F1 119
- F2 119
- F3 119
- F4 119
- F5 119
- F6 119
- INS 119
- LINEFEED 123
- template 117

Key sequence

- CONTROL-ALT-F1 65
- CONTROL-ALT-F2 65
- CONTROL-BREAK 68
- CONTROL-C 123, 141, 153, 247
- CONTROL-H 123
- CONTROL-J 123
- CONTROL-N 123
- CONTROL-P 123
- CONTROL-S 123
- CONTROL-V 141
- CONTROL-X 123
- CONTROL-Z 140, 163

Keyboard

- copying information from 102
- type 65

Keybxx command 65

- purpose 65
- syntax 65

Keys, reassigning 257

Label command 66

- prompts 66
- purpose 66
- syntax 66

Label variable, batch 112

Letters, lowercase 209

Letters, uppercase 209

.Lib extension 183

LIB, environment variable 189

Library file 183, 185

Line editor, Edlin 10, 10, 125

Line mode 262

Line option, Edlin

- description 142
- number 142
- number sign 142
- period 142

Line parameter 257

LINEFEED key 123

/linenumbers option, abbreviation 195

/linenumbers option, Link 185, 195

Lineprinter 73

Lineprinter, output to 123

Link

- align type 203
- canonical frame number 203
- class type 204
- command line 184
- creating an executable file 184
- default extensions 182
- description 181
- directives 203
- frame number, canonical 203
- groups 205
- library files 183, 185
- library search 197
- map file 183, 184, 190, 194, 195
- operation 203
- option
 - /cparmaxalloc 198
 - /dosseg 202
 - /dsallocate 191, 199
 - /exepack 194
 - /help 193
 - /high 191, 199
 - /linenumbers 195
 - /map 185, 190, 194
 - /nodefaultlibrary 197
 - /nodefaultlibrarysearch 197
 - /nogroupassociation 200
 - /noignorecase 196
 - /overlayinterrupt 200
 - /pause 193
 - /segments 201
 - /stack 197
- overlays 185, 200
- placeholder 184
- preserving case-sensitivity 196
- prompts 182
- references
 - long 206
 - near segment-relative 206

Link (continued)

- references (*continued*)
 - near self-relative 206
 - short 206
- response file 187
- search paths 189
- starting 182
- temporary file 191
- using the comma with 183
- using the semicolon with 183
- variable
 - executable file 184
 - libraryfiles 185
 - mapfile 185
 - objectfiles 184

Link options 192**Linking, described 181, 203****List command, Edlin**

- purpose 156
- syntax 156

Listing a directory 6**Listing lines, Edlin 156****Load command, Debug**

- abbreviation 225
- comments 225
- purpose 225
- syntax 225

Locating a file 4**Locating files 2****Lock violation error 268****Long reference, Link 206****Lowercase letters 209****Lowercase letters, preserving 196****Making a directory 10, 67****.Map extension 183****Map file**

- addresses 190
- creating 183, 195
- format 184
- including line numbers 194
- including public symbols 190
- segments 190

/map option, abbreviation 194**/map option, Link 185, 190, 194****Media, disk**

- floppy 29
- hard 29

Memory

- allocating disk buffers 248
- computer 13

Menu, BASIC 105**Merging files, Edlin 169****Message directory 271****Message, placeholder 109****Misspelling, template 121****Mistakes, correcting 117****Mkdir command**

- purpose 67
- syntax 67
- working directory 67

Mnemonics, Debug 213**Mode**

- asynchronous communications 68
- display 68
- inverse video 76
- operation 68
- parallel printer 68

Mode command

- asynchronous communications mode
 - default settings 69
 - options 68
- display modes, options 69
- parallel printer mode
 - default settings 68
 - options 68
- purpose 68
- syntax 68

Modes of operation 260**Monochrome display adapter 69****More command**

- purpose 71
- syntax 71

Move command, Debug

- abbreviation 227
- comments 227
- purpose 227
- syntax 227

Move command, Edlin

- purpose 159
- syntax 159

Moving lines, Edlin 159**MS-DOS**

- argument 21
- command processor 37, 55
- exiting to from Edlin 162
- file header 192
- function keys 117

MS-DOS (*continued*)

- line editor, Edlin 101
- special editing keys 117
- special prompts 75
- switches 21

MS-DOS command

- Append 24
- Assign 26
- Attrib 27
- Backup 29, 83
- Break 32
- Chdir (cd) 10, 33
- Chkdsk 3, 34
- Cls 36
- Command 37
- Copy 14, 39
- Date 44
- Del 14, 45
- Dir 6, 11, 14, 46
- Diskcomp 48
- Diskcopy 14, 51
- Echo 109
- Exe2bin 53
- Exit 55
- external 13, 14, 15
- Fdisk 56
- Find 17, 57
- For 110
- Format 3, 14, 58
- Goto 112
- Graftabl 61
- Graphics 62
- If 113
- internal 13
- Join 64
- Keybxx 65
- Label 66
- Mkdir 10, 67
- Mode 68
- More 17, 71
- Path 15, 72
- Pause 114
- Print 73
- Prompt 75
- Recover 77
- Rem 115
- Ren 78
- Replace 79
- Restore 82
- Rmdir 11, 84

MS-DOS command (*continued*)

- Set 85
- Set, used with Link 189
- Share 87
- Shift 106, 116
- Sort 17, 88
- Subst 89
- syntax 20
- Sys 90
- Time 91
- Tree 92
- Type 14, 93
- Ver 94
- Verify 95
- Vol 96
- Xcopy 97

MS-DOS commands, listing of 22

MS-DOS prompt

- allowed characters 75
- command 75
- default drive letter 243
- default setting 75
- special prompts 75

MS-DOS version number, displaying 94

Msdos.sys file 90

Multilevel directory

- description 3
- Mkdir command 67
- removing a directory 84
- using wildcards with 8

Name command, Debug

- abbreviation 228
- comments 228
- purpose 228
- syntax 228

Near segment-relative reference, Link 206

Near self-relative reference, Link 206

Network 19

Network, symbol 20

No paper error 268

/nodefaultlibrary option, abbreviation 197

/nodefaultlibrarysearch option, Link 197

/nogroupassociation option, Link 200

/noignorecase option, abbreviation 196

/noignorecase option, Link 196

Non-DOS disk error 268

NOT parameter, If command 113

Not ready error 268

- NUL data path 24, 72
- Numeric parameter 257
- Obj extension 182
- Object file, Link 182
- Object Linker. *See* Link
- Omitting options, Edlin 156
- One-drive system 243
- Opcodes, Debug 214
- Operating system, exiting to from Edlin 162
- Operating system, XENIX 254
- Operation modes 68
- Options, command 21
- Order of segments, Link 204
- Organizing files 3, 6
- Output
 - appending 16
 - redirecting 16, 175
 - sending to lineprinter 123
 - stopping 123
- Output command, Debug
 - abbreviation 230
 - comments 230
 - purpose 230
 - syntax 230
- /overlay interrupt option, abbreviation 201
- /overlay interrupt option, Link 200
- Overlays, Link 185, 200
- Packing executable files 194
- Page command, Edlin
 - purpose 161
 - syntax 161
- Paging through files, Edlin 161
- Parallel 68
- Parameter
 - changing position of 116
 - condition 113
 - definition 106
 - line 257
 - NOT, If command 113
 - numeric 257
 - replaceable 106, 108
 - selective 257
 - shifting 116
- Parameter, replaceable 85
- Parent directory
 - Chdir command 33
 - Parent directory (*continued*)
 - parent directory 7
 - shorthand notation 7
- Parity 68
- Path
 - definition 21
 - NUL 72
 - specifying a working path 15
 - syntax 21
 - using with internal command 14
- Path command 15
 - purpose 72
 - syntax 72
- Pathname
 - as a command option 20
 - definition 6
 - maximum length 7
 - Restore option 82
 - searching 7
 - syntax 6, 20
 - using with Chdir command 10
 - using with internal commands 14
- Pause command, batch
 - purpose 114
 - syntax 114
- /pause option, abbreviation 193
- /pause option, Link 193
- Percent sign, used in a batch file 106
- Permission
 - read-only 27
 - write 27
- Pipe 16
- Pipe symbol (!) 17
- Pipe, Sort command 88
- Pipeline, definition 18
- Piping and redirecting
 - definition 16
 - input 17
 - output 17
- Piping commands 17
- Placeholders 106
- Port, asynchronous communications
 - COM1 68
 - COM2 68
- Preserving lowercase characters, Link 196
- Print command
 - purpose 73
 - switch
 - /b 73
 - /c 73

Print command (*continued*)switch (*continued*)

/d 73

/p 73

/q 73

/t 73

syntax 73

Print queue

cancel mode 73

deleting files from 73

print mode 73

Print queue, number of files allowed in 73

Printer, serial 69

PRN, default printer 73

Processing, batch files 85, 101, 110

Processing, interactive 110

Processor, command 255

Program

Autoexec.bat 16

batch 101

Command.com 13

Diskcopy.exe 14

Format.exe 14

Program, executable 184

Prompt

> (greater-than sign) 75

\$ (dollar sign) 75

< (less-than sign) 75

ASCII code (escape) 75

asterisk (*), Edlin 126

date 75

default drive 75

default drive letter 243

RETURN-LINEFEED 75

space 75

time 75, 91

version number 75

working directory, default drive 75

| (pipe symbol) 75

Prompt command

purpose 75

syntax 75

Prompt, MS-DOS 75

Protecting files 2

Public-symbol 190

Public-symbol map 194

Punctuation, use of 21

Question mark (?), wildcard character 8

Question mark option, Edlin 142, 166

Queue, print 73

Quit command, Debug

abbreviation 231

comments 231

purpose 231

syntax 231

Quit command, Edlin

purpose 162

syntax 162

Quotation marks, use of 21, 57

Ramdrive.sys 257

Rapid Blink function, ANSI 261

RCP-Restore Cursor Position, ANSI escape

sequence 259

Read fault error 268

Read-only file, restoring 82

Read-only permission 27

Recover command

purpose 77

syntax 77

Recovering a file 77

Redirecting 175

Redirecting and piping

input 17

output 17

Redirecting parallel printer output 68

Redirection and piping symbols 103

Redirection symbol

greater-than sign (>) 16

less-than sign (<) 17

two greater-than signs (>>) 16

.Ref extension 41

Register command, Debug

abbreviation 232

comments 232

purpose 232

syntax 232

Rem command, batch

purpose 115

syntax 115

Remark, batch file 115

Removing

a directory 11, 84

a virtual drive 89

files 12, 45

Removing groups from a program, Link 200

- Ren command
 - purpose 78
 - syntax 78
 - using wildcards with 78
- Renaming a directory 12
- Renaming a file 78
- Replace command
 - purpose 79
 - syntax 79
- Replace command, Edlin
 - purpose 163
 - syntax 163
- Replace mode, Edlin 135
- Replaceable parameters 85, 106, 108
- Replacing text, Edlin 163
- Responding to errors
 - Abort 270
 - Ignore 270
 - Retry 270
- Response file, Link 187
- Restore command
 - purpose 82
 - switch
 - /a 82
 - /b 82
 - /e 82
 - /L 82
 - /m 82
 - /n 82
 - /p 82
 - /s 82
 - syntax 82
- Restoring
 - a file 82
 - a subdirectory 82
- RETURN-LINEFEED prompt 75
- Reverse sorting (Z to A) 88
- Reverse Video function, ANSI 261
- Revising text, Edlin 125
- RM-Reset Mode, ANSI escape sequence 262
- Rmdir command
 - purpose 84
 - shorthand notation 12
 - syntax 84
- Root directory
 - Autoexec.bat 103
 - Config.sys file 245
 - definition 3
 - joining with 64
 - making subdirectories 67
- Root directory (*continued*)
 - shorthand notation 6
 - subdirectories of 5
- Running a batch file 107
- Sample Config.sys file 256
- Saving a file, Edlin 127
- SCP-Save Cursor Position, ANSI escape sequence 259
- Screen
 - aligning, Mode 69
 - changing 257
 - erasing 259
 - graphics 260
 - width, changing 262
- Search command, Debug
 - abbreviation 235
 - comments 235
 - purpose 235
 - syntax 235
- Search command, Edlin
 - purpose 166
 - syntax 166
- Search path 72
- Search paths, Link 189
- Searching a directory, Path 72
- Searching for a file 4
- Searching for text 17
- Searching for text, Edlin 166
- Sector 77
- Sector not found error 268
- Seek error 269
- Segment alignment, Link 203
- Segment number, setting maximum 201
- Segment order, MS-DOS convention 202
- /segments option, abbreviation 202
- /segments option, Link 201
- Maximum memory allocation, controlling in Link 198
- Selective parameter 257
- Semicolon, used for options 21
- Separating files 3
- Separator
 - equal sign 112
 - semicolon 112
- Separators in comments
 - comma 115
 - space 115
 - tab 115

- Sequences of commands 101
- Serial printer 69
- Set command
 - purpose 85
 - syntax 85
- Set command, used with Link 189
- Setting a working path 16
- SGR-Set Graphics Rendition, ANSI escape
 - sequence 260
- Share command
 - purpose 87
 - syntax 87
- Sharing violation 269
- Shell
 - definition 255
 - starting 255
- Shell command, Config.sys file
 - definition 246
 - purpose 255
 - syntax 255
- Shift command 106
- Shift command, batch
 - purpose 116
 - syntax 116
- Shifting parameters 116
- Short reference, Link 206
- Shortcuts
 - asterisk (*) 8
 - Chdir command (Cd) 11, 33
 - directory
 - "." 84
 - ".." 84
 - parent directory 7
 - question mark (?) 8
 - Rmdir command (Rd) 84
 - wildcard characters 8
 - working directory 7
- Shorthand notation. *See* Shortcuts
- Sign
 - greater-than (>) 16
 - less-than (<) 16
- Single-drive system 243
- Skipping over template characters 122
- Sort command 17
 - filter 88
 - pipe 88
 - purpose 88
 - switch
 - /+ *n* 88
 - /r 88
- Sort command (*continued*)
 - syntax 88
- Sorter.bat file 106-107
- Sorting
 - a directory 18
 - a file 17
 - alphabetically 107
 - files 88
 - in reverse order (Z to A) 88
- Source drive 21
- Source file 39
- Space prompt 75
- Spaces and tabs, used for options 21
- Special editing key
 - DEL 119
 - description 128
 - ESC 119
 - F1 119
 - F2 119
 - F3 119
 - F4 119
 - F5 119
 - F6 119
 - INS 119
- Special editing keys, using with Edlin 125
- Specifying a working path 15
- /stack option, abbreviation 197
- /stack option, Link 197
- Stack size, controlling with Link 197
- Starting
 - Edlin 126
 - FC 172
- Stopbit 69
- Stopping a command 32, 123
- Stopping Edlin 127
- String
 - definition 21
 - environment 85, 103
 - finding 57
 - invalid separators
 - comma 113
 - equal sign 113
 - semicolon 113
 - space 113
 - replacing, Edlin 163
 - searching, Edlin 166
 - separating with CONTROL-Z 163
- String, Debug 212
- Structure
 - directory 4

Structure (continued)

- file 4
 - hierarchical 4
 - multilevel 4
- Subdirectory 3, 5, 11
- Subdirectory, restoring 82
- Subfunction 257
- Subscript function, ANSI 261
- Subst command
 - purpose 89
 - syntax 89
- Superscript function, ANSI 261
- Switch
 - definition 21
 - syntax 21
- Symbol
 - greater-than sign (>) 16
 - less-than sign (<) 16
 - redirection 103
- Symbol-map file, format 190
- Synonym, Del/Eraser 45
- Syntax
 - command 20
 - path 21
 - pathname 6, 20
 - switch 21*command. See Command syntax*
- Sys command
 - purpose 90
 - syntax 90
- System calls, XENIX-compatible 254
- System file
 - Io.sys 90
 - Msdos.sys 90
- System files, transferring 90
- System prompt 243
- System, file 3

Target disk 82, 90

Target drive 21

Target file 39

Template

- advantages of 117
- automatic overwriting 120
- dealing with misspellings 121
- deleting contents of 137, 138
- relation to command line 118
- using the DEL key with 132
- using the ESC key with 134

Template (continued)

- using the F1 key with 129
- using the F2 key with 130
- using the F4 key with 133
- using the F5 key with 137
- using the INS key with 135

Template character

- copying 122
- deleting 122
- inserting 122
- skipping over 122

Temporary file

- deleting 108
- purpose 108
- used in batch processing 106
- used with the more command 71

Temporary file, Link 191**Terminating a batch program 114****Text editor, Edlin 10****Text file, creating with Edlin 125****Text option, Edlin 143****Text, moving, Edlin 159****Text, searching for 17****Time command**

- purpose 91
- syntax 91

Time prompt 75, 91**Trace command, Debug**

- abbreviation 236
- comments 236
- purpose 236
- syntax 236

Transfer command, Edlin

- purpose 169
- syntax 169

Transferring system files 90**Transferring text, Edlin 169****Transmission rate 68****Tree command**

- purpose 92
- syntax 92

Two-letter code, Keybxx 65**.Txt extension 8, 9, 41, 78****Type command 14**

- purpose 93
- syntax 93

Type messages 267**Typical Config.sys file 256**

Unassemble command, Debug
 abbreviation 237
 comments 237
 purpose 237
 syntax 237

Underline, cursor 120

Updating the system 90

Uppercase letter, use of 156

Uppercase letters 196, 209

User directory 5

Variable, replaceable 85

Ver command

purpose 94

syntax 94

Verify command

purpose 95

syntax 95

Version number prompt 75

Version number, displaying 94

Vertical bar. *See* Pipe symbol (|)

Virtual drive

creating 89

definition 89

removing 89

Vol command

purpose 96

syntax 96

Volume label

allowed characters 66

creating 66

deleting 66

displaying 96

length 59, 66

Wildcard character

asterisk (*) 8

destructive uses of 9

question mark (?) 8

shortcuts 8

special uses 9

using the asterisk (*) 8

using the question mark (?) 8

using with Del 12

Working directory 5

definition 4

displaying 10, 33

external command directory 15

Working directory (*continued*)

Mkdir command 67

program directory 15

prompt, default drive 75

searching for data files in 24

shorthand notation 7

using filenames 7

using pathnames 7

Working path 15

Write command, Debug

abbreviation 239

comments 239

purpose 239

syntax 239

Write command, Edlin

purpose 170

syntax 170

Write fault error 269

Write-protect error 269

Write-protect notch 2, 269

Write-protect tab 269

Write-protected disk 71

Write-protection 2

Writing to

a disk 95

a disk, Edlin 170

Xcopy command

purpose 97

syntax 97

XENIX operating system 254

XENIX-compatible system calls 254





Commodore Business Machines, Inc.
1200 Wilson Drive
West Chester, PA 19380

Commodore Business Machines, Limited
3470 Pharmacy Avenue
Agincourt, Ontario M1W 3G3

Commodore Business Machines Pty Limited
67 Mars Road
Lane Cove, New South Wales 2066 Australia