Microsoft® Operating System/2

User's Reference

Microsoft Corporation

Document Number 510830002-200-O01-1087

Contents

Welcome

This is a resource manual for Microsoft® Operating System/2 (MS® OS/2). It is unlikely that you will ever sit down to read it cover to cover. Rather, you will probably pick it up when you are looking for a specific piece of information. The *Microsoft Operating System/2 User's Reference* was designed with that in mind.

This manual is written for the person who has some experience using a personal computer and includes sections that will help you to learn about some of the more advanced features of MS OS/2. It assumes that you know a little about whan an operating system is and does.

Many of the MS OS/2 commands discussed in this manual are similar to MS-DOS® commands. If you have never used a computer with an operating system such as MS-DOS, you should read through the *Microsoft Operating System/2 Beginning User's Guide* before you go on to this manual.

Other manuals that may help you

These other manuals will also be of interest to you:

- The *Microsoft Operating System/2 Setup Guide* explains how to set up MS OS/2 on a personal computer system.
- The *Microsoft Operating System/2 Beginning User's Guide* introduces you to the basics of using a personal computer with MS OS/2.

In addition, three manuals discuss topics of interest to programmers:

- The *Microsoft Operating System/2 Programmer's Guide* introduces programmers to MS OS/2.
- The *Microsoft Operating System/2 Programmer's Reference* describes all MS OS/2 system calls.
- The *Microsoft Operating System/2 Device Drivers Guide* describes how to use device drivers with MS OS/2 and explains device-driver commands.

Notational Conventions

Elements of text

Throughout this manual, the following conventions are used to distinguish elements of text:

Text	Use
bold	Distinguishes commands, options, switches, and literal portions of syntax that must appear exactly as shown.
italic	Distinguishes variables, and placeholders that represent the type of text to be entered by the user.
`monospace`	Distinguishes sample command lines, program code and examples, and sample sessions.
CAPITALS	Distinguish filenames, directory names, programming languages, and acronyms.
SMALL CAPS	Distinguish keys and key combinations.

A plus sign (+) used between two keynames indicates that those keys must be pressed at the same time. For example, "Press CONTROL+C" means that you should press and hold down the CONTROL key while you press and release the C key. Then release the CONTROL key.

How to Use This Manual

About this manual

The following is a quick overview of the topics covered in this manual:

Turn to	To find out about
Chapter 1	MS OS/2 features, including memory management and operating modes
Chapter 2	Using the program selector; running multiple programs at the same time
Chapter 3	Creating and using batch files
Chapter 4	Using function keys for editing
Chapter 5	Using Edlin, the real-mode line editor
Chapter 6	Advanced techniques for using MS OS/2 commands; using redirection and grouping symbols with MS OS/2 commands
Chapter 7	MS OS/2 commands; batch commands
Appendix	MS OS/2 command exit codes
Glossary	Terms used in the MS OS/2 user's manuals

1 What is MS OS/2?

Before you begin to use Microsoft Operating System/2 you should know a little about some of its features. MS OS/2 is much like its predecessor, MS-DOS 3.*x*, but is an even more versatile and powerful disk operating system.

In this chapter, you will learn the following:

- What MS OS/2 is and what it does
- Some of the unique features of the MS OS/2 operating system

Introducing MS OS/2

The MS OS/2 operating system acts as a translator between you and your computer. The programs in this operating system allow you to communicate with your computer, your disk drives, and your printer.

MS OS/2 also helps you manage programs and data. Once you have loaded MS OS/2 into your computer's memory, you can compose letters and reports, run a variety of programs, and use programming languages such as Microsoft BASIC.

MS OS/2 has a number of unique features, including the following:

- Multitasking
- Dual modes—real mode and protected mode
- An enhanced memory manager
- On-line help

Multitasking

Running several tasks at once

MS OS/2 lets you run several tasks at the same time. A task is something your computer does when you give it a command by typing at the keyboard. Running an application program is an example of a task. Most microcomputers perform tasks sequentially. This means that they do one task at a time, and finish one task before starting the next.

With MS OS/2, though, your computer doesn't have to finish one task before starting another. It can perform more than one task at a time. This ability is called multitasking.

For example, you could run a word-processing program and a spreadsheet program on your computer at the same time. MS OS/2 even lets you jump back and forth between programs without having to end either one. So, while you are plotting a graph of last year's expenses with the spreadsheet, you can write a memo to accompany it with the word processor.

Real and Protected Modes

Dual Modes

MS OS/2 has two unique environments in which you can run programs. Each environment, called a mode, is like an operating system of its own.

The first MS OS/2 environment is called real mode. This environment looks and acts just like MS-DOS 3.*x*. Almost any application program that you can run on DOS 3.*x* also runs in MS OS/2 real mode. You can tell that you are in real mode when you see the real-mode prompt. The default real-mode system prompt is the current drive letter followed by the greater-than (>) symbol, for example:

```
C>
```

The second MS OS/2 environment is called protected mode. In some ways, it seems like MS-DOS 3.*x*, but it is actually much more powerful. For example, in protected mode, you can run more than one program at the same time. This is possible because the protected mode "protects" one program from being interfered with by another. The default protected-mode system prompt is the current drive letter and path enclosed within brackets, for example:

```
[C:\]
```

Command processors

Each of the two MS OS/2 modes has its own command processor. When you type a command, it is the command processor that translates the command into machine language for your computer. The real-mode command processor is called COMMAND.COM. The protected-mode command processor is called CMD.EXE. Both of these files are on your program disk.

When you enter real mode, MS OS/2 uses the COMMAND.COM command processor to interpret your commands. When you enter protected mode, MS OS/2 uses CMD.EXE.

Most MS OS/2 commands work in both protected and real modes. However, certain MS OS/2 commands act differently in protected mode than they do in real mode. For example, in real mode, the **del** command allows you to delete only one file at a time. But in protected mode, you can delete multiple files by listing more than one filename with a single **del** command.

For more information about how specific MS OS/2 commands work in real and protected modes, see Chapter 7, "MS OS/2 Command Reference."

Enhanced Memory Manager

MS OS/2 is designed to run on personal computers that have an Intel 80286 or 80386 processor. The 80286 processor can access twice as much memory as its predecessor, the Intel 8086 processor—up to 16 megabytes. MS OS/2 takes advantage of this ability by using a memory manager that keeps track of everything that goes on in memory.

The MS OS/2 memory manager

The MS OS/2 memory manager knows which programs you are running and the memory requirements of each. It will also let you run more programs than will fit into memory. This is called storage overcommitment. In fact, even if your application and its data are larger than the memory available, MS OS/2 may let you run that application. It can move programs in and out of memory to disk so that all of your programs can continue to run.

On-line Help

Using Helpmsg

Another feature of MS OS/2 is a special on-line help program called Helpmsg. This program provides information about the messages that MS OS/2 displays on your screen.

Some messages appear if you make a mistake while using a command. Other MS OS/2 messages prompt you for information. Each message that MS OS/2 displays is preceded by a unique identification number. This number begins with three letters (such as "SYS") followed by four numbers.

To find out what a particular message means, and what action you need to take, type a command in this format:

helpmsg *messageid*

Messageid is the unique identification number that MS OS/2 displayed. In response to this command, MS OS/2 displays a help message that explains the original message. If applicable, the help message also suggests what steps you should take in response.

For example, suppose you want to rename a file called NEWHOST.TLK to OLDHOST.TLK. You would type this command line:

```
ren newhost.tlk oldhost.tlk
```

But if a file called OLDHOST.TLK already exists, MS OS/2 displays this message:

```
SYS1083: A duplicate filename exists,
or the file was not found.
```

If you want to find out what this message means and what to do about it, you would type this command line:

```
helpmsg sys1083
```

MS OS/2 displays a help message like the following to explain the original message more fully:

```
Explanation: The target filename specified
             with the rename command already
             exists or the source file could
             not be found.
Action: Check the source and target
             filenames used and retry the
             command.
```

In the next chapter

In Chapter 2, you will learn how to start MS OS/2 and how to use the MS OS/2 program selector to run one or more programs.

2 Running Programs

With Microsoft Operating System/2 you can run more than one program at a time. MS OS/2 uses the program selector to create work environments for each program you want to run. These work environments, called sessions, are listed on the program-selector screen. You can use this list of sessions to see what programs are running and to move from one session to another.

In this chapter, you will learn the following:

- How to start MS OS/2
- What the program selector is and what it does
- How to start a session with the program selector
- How to switch between programs
- How to end programs and sessions

Starting MS OS/2

Two ways to start MS OS/2

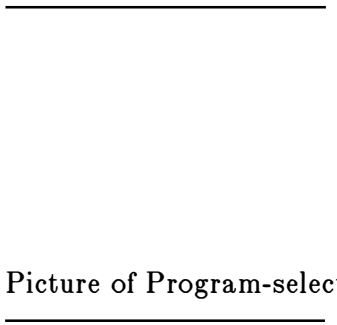
Before you learn how to work with the MS OS/2 program selector, you will need to learn how to start MS OS/2. Assuming that you have installed MS OS/2 on your hard disk, there are two ways to start MS OS/2:

- The first way is simply to turn on your computer. Be sure to turn on your screen as well. MS OS/2 starts automatically.
- If your computer is already on, you can press CONTROL+ALT+DELETE. Make sure no programs are currently running when you use this method. If programs are running when you press this key combination, you may lose some data.

If you have not yet installed MS OS/2 on your hard disk, see the *Microsoft Operating System/2 Setup Guide* for information on how to do so.

The Program-Selector Screen

When you first start MS OS/2, the program-selector screen is displayed automatically:



Picture of Program-selector Screen

This screen contains four sections:

- **Switch to a Running Program** This section of the program-selector screen lists all of the sessions that are currently running. It also lists the real-mode session (MS-DOS Command Prompt) whether you have started it or not. This is because you may run only one real-mode session.
- **Start a Program** This section contains a list of protected-mode programs or commands that you may want to start. Except for the first item, you will create this list yourself using the Update option. The first item in the list (MS OS/2 Command Prompt) is given automatically. If you select this item, you can start a new protected-mode session. You may run multiple protected-mode sessions with MS OS/2.
- **Update** This option is listed in the upper-left corner of the program-selector screen. You can use this option to create, change, or delete items in the Start a Program list.
- **Help** This option is listed in the upper-right corner of the program-selector screen. The on-line help program gives you information about tasks associated with the program selector.

Note To run DOS 3.*x*-compatible programs, you must include the following command line in your CONFIG.SYS file:

```
protectonly=no
```

For more information about the CONFIG.SYS file, see the *Microsoft Operating System/2 Setup Guide.*

Selecting an option

You can choose items from the program-selector screen by using either the mouse or the keyboard. With the mouse, you can choose an item from the list in the Switch to a Running Program or Start a Program section by pointing to the desired item with the mouse cursor and pressing the left mouse button twice. You can select the Update option or the Help option by pointing to either of these items on the program-selector screen and pressing the left mouse button twice.

You can also use the keyboard to make your selection. Use the DIRECTION keys to select items from the two lists (Start a Program and Switch to a Running Program). Press the F1 key to select the Help option. Press the F10 key to select the Update option.

Useful keys

You will also find the following keys useful for editing and moving around the program-selector screen:

Key	Action
CONTROL+LEFT	Moves the cursor to the beginning of a field.
CONTROL+RIGHT	Moves the cursor to the end of a field.
HOME	Moves the cursor to the first position of a field that is visible on the screen (some fields may be scrolled).
END	Moves the cursor to the last position of a field that is visible on the screen.
INSERT	Enters or exits insert mode (toggles).
DELETE	Deletes the character marked by the cursor.
CONTROL+HOME	Moves the cursor to the beginning of data in the first field of the Update screen.
CONTROL+END	Moves the cursor to end of data in the last field of the Update screen.
ALT+F7	Erases characters from the cursor to the end of the field.

Starting a New Program

Every time you use the program selector to start a new program or command, you begin a new session. For example, suppose you want to do some work using Microsoft Multiplan®. When you start Multiplan, the program selector shows that you have one session running. The program selector uses the name of the program to identify the session. Then, suppose you decide to write a memo to your accountant. You can use the program selector to start a word-processing program. When you do this, the program selector will show that you are now running two sessions. The program-selector screen might look something like this:

[program selector w/2 sessions: Plan; Word]

When you finish writing your memo, you can use the program selector to switch back to the exact place you left Multiplan.

Starting a program

There are two ways to start a new program:

- Select a protected-mode program name from the Start a Program list on the program-selector screen.
- Start a new session. Then, from that session, type the command that starts the program.

Starting a New Session

To start a new session to run a program or MS OS/2 command, use the DIRECTION keys to highlight either MS OS/2 Command Prompt in the Start a Program section (for a protected-mode session) or MS-DOS Command Prompt in the Switch to a Running Program section (for the real-mode session). Press the ENTER key to make your selection. Once you do this, the program selector switches you to the new session. If it is a real-mode session, a real-mode prompt like this one will be displayed:

```
C>
```

If it is a protected-mode session, the prompt will look like this:

```
[C:\]
```

MS OS/2 is now ready for you to start an MS OS/2 command or a program. (See the user's documentation for the program to find out how to start it.)

For example, if you want to start a protected-mode program called Big Deal, you would follow these steps:

1. Choose MS OS/2 Command Prompt from the Start a Program section of the program-selector screen. A protected-mode session begins.
2. Type the command that starts the program after the protected-mode prompt ([C:\]):

```
big deal
```

Updating the Start a Program List

The program selector lets you maintain a list of your most-used programs. In addition to saving the name of a program, the program selector stores the path and parameters for each program. You can change the information on the Start a Program list with the Update option.

Using the Update menu

Press the F10 key to see the Update menu. With the Update menu, you can do the following:

- Add a program title to the Start a Program list
- Delete a program title from the Start a Program list
- Change information about a program's path or parameters

Adding a Program

Suppose you have a protected-mode program called Planner that you run every morning. You could run this program by first starting a protected-mode session, and then typing the command that starts your program. An easier way, though, would be to add Planner to the Start a Program list.

Adding a program name

To add a program to the Start a Program list, follow these steps:

1. Press the F10 key to select the Update option from the program-selector screen. The Update menu appears in the upper-left part of the screen.
2. Press the ENTER key to make the highlighted selection bar appear. It is automatically positioned on the first line, Add a Program Title.

3. Press the ENTER key to select Add a Program Title. Another screen appears.
4. In the first line, type the program name that will appear on the Start a Program list.
5. On the next line, type the path of the program and the command that starts the program.
6. On the third line, type any parameters that you want to use each time you run this program. If you want MS OS/2 to prompt you for parameters each time you run the program, type ? in this space.
7. Press the ENTER key to add this information to the Start a Program list.

Changing or Deleting Program Information

You can also use the Update menu to change program information or delete a program from the Start a Program list.

For example, if you added a new graphics board to your computer, you might want to change the parameter line for one or more programs in the Start a Program list to take advantage of the new board.

Changing program information

To change program information, follow these steps:

1. Press the F10 key to select the Update option from the program-selector screen.
2. Press the ENTER key to make the highlighted selection bar appear.
3. Use the DOWN key to move the selection bar to the Change Program Information line, and press the ENTER key to make your selection. The current program information appears in another screen.
4. Use the DIRECTION keys to move the cursor to the entry you want to change and type the desired program title, path, start command, and any parameters over the old information.
5. Press the ENTER key to update the information for that program.

Deleting program information

You can also use the Update menu to delete information about a program and remove the program name from the Start a Program list. This is *not* the same as deleting the program files from your system. The application program's files are not affected by this process. To remove a

program name and the related program information from the Start a Program list, follow these steps:

1. Press the F10 key to select the Update option from the program-selector screen. The Update menu appears in the upper-left part of the screen.
2. Press the ENTER key to make the highlighted selection bar appear.
3. Use the DOWN key to move the selection bar to the Delete a Program Title option and press the ENTER key to make your selection.
4. A final screen asks you to verify your decision before the program name and information is actually removed. Press the ENTER key to delete the program title and program information from the list.

Switching Between Sessions

You can use the program selector to switch between the sessions you are running. The program selector lets you switch between sessions much like a television channel selector lets you switch between channels.

There are two ways to switch between sessions:

- You can use the program-selector screen as a menu to select the session you would like to work in.
- You can switch directly from one session to the next.

Switching with the program selector

To switch between sessions using the program-selector screen, do the following:

1. Press CONTROL+ESCAPE. The program-selector screen appears and lists the names of the programs that are running in the Switch to a Running Program section.

   The program-selector screen should look something like this:

[screen dump: program selector screen]

By choosing MS-DOS Command Prompt from the Switch to a Running Program section, you switch to programs that run in real mode. If MS-DOS Command Prompt is not listed in this section on the program-selector screen, check the CONFIG.SYS file. If your CONFIG.SYS file contains the command line **protectonly=yes**, you cannot run real-mode programs on your computer. You must change it to **protectonly=no** to run real-mode programs.

2. Select the name of the session you would like to work in next by pressing the DIRECTION keys. Use the UP key to move the highlighted selection bar up. Use the DOWN key to move the highlighted selection bar down.
3. Press the ENTER key to make your selection and switch to the selected program.

   When you move into a session, that session becomes the active session.

Switching by pressing ALT+ESCAPE

To switch between existing sessions without returning to the program-selector screen, press ALT+ESCAPE. This action moves you directly from one session to the next. If you have two or more sessions, you can use ALT+ESCAPE to move through the sessions in the order that they are listed in the Switch to a Running Program list, as though you were flipping through the channels on a television.

Using the Help Option

If you would like more information about a function of the program-selector screen, you can press the F1 key to

access the on-line help program. This help program is context sensitive, meaning that it will show you information about the task that you are working on. For example, if you move the cursor to the Select a Program list and then press F1, help information about selecting a program is displayed.

The on-line help information is displayed in small screens that overlay the program-selector screen. In many cases the information available on a subject does not fit in the help screen. If this is the case, the word "More" is displayed in the upper-right corner of the help screen. You can use the HOME, END, PAGE UP, and PAGE DOWN keys to scroll through the help text.

At the bottom of each help screen, options are displayed. These include the following:

- Press the ESCAPE key to remove the help screen. (If several help screens are displayed, you need to press ESCAPE for each help screen.)
- Press the F1 key for a help screen that provides general information.
- Press the F5 key for an index of topics on which you can get help.
- Press the F9 key for a help screen that describes special function keys and other key combinations.

Using the help index

If you press F5 to display the help index, use the DIRECTION keys to highlight a topic of interest. Then, press the SPACEBAR to make your selection. The program selector displays a new help screen about the topic you selected.

Ending a Session

Before you can end a session, you must be sure that all MS OS/2 commands in the session and all programs in the session have ended.

Ending an MS OS/2 Command

MS OS/2 commands are designed to run their course and end by themselves. However, if you want to end an MS OS/2 command before it is completed, you can press CONTROL+C.

For example, if you are sorting a very large file and decide to cancel the **sort** command, you would do the following:

1. Make sure that you are in the correct session.
2. Press CONTROL+C.

The command is cancelled and MS OS/2 displays the appropriate prompt (C> or [C:\]).

Ending a Program

You can run three types of programs:

- Interactive programs
- Non-interactive programs
- Detachable non-interactive programs

Ending interactive programs

Interactive programs require input from you. These programs wait for your instructions before processing. Examples of interactive programs include word-processing applications and programs that ask you to respond to questions as those programs run.

To end an interactive program, follow these steps:

1. Make sure that you are in the session belonging to the program you want to end.

   If you are not, press CONTROL+ESCAPE to move to the program-selector screen. Choose the program that you want to end, and press the ENTER key.
2. Type the command to end the interactive program.

   See the user's documentation for the program to find out how to end that program.

Ending non-interactive programs

Non-interactive programs do not need your input as they run. Examples of non-interactive programs include program compilers and print spoolers. Non-interactive programs must end on their own. Sometimes, you can press CONTROL+C to stop such a program. See the documentation for the program to see if this method will work.

Ending detachable programs

The third type of program you may run is a detachable program. These are non-interactive programs that run in the background, and generally end on their own. You can run detachable programs in protected mode only. Most

programs run in the foreground, meaning that you can see what they are doing as they run.

Detachable programs are run using the **detach** command. When you start a detachable program, MS OS/2 displays a message that shows an assigned process identification (PID) number. You see nothing else related to that program until it ends. When a detached program ends, MS OS/2 lets you know by displaying another message with the same PID number.

For more information about detaching programs, see the description of the **detach** command in Chapter 7, "MS OS/2 Command Reference."

If you are not sure whether a session has a program or command running, look at its name on the program-selector screen. If a protected-mode session has nothing running in it, that session will be named either MS OS/2 Command Prompt or CMD.EXE.

Using Exit to End a Session

Ending a protected-mode session

To end a protected-mode session, do the following:

1. Press CONTROL+ESCAPE to move to the program-selector screen.
2. Select the session that you want to end by moving the highlighted selection bar to the correct name and pressing the ENTER key.
3. Make sure that all programs and commands for this session have ended.
4. Type **exit** and press the ENTER key.

The program-selector screen now shows one less session.

Note You cannot remove the real-mode session, MS-DOS Command Prompt, from the program-selector screen.

Like MS OS/2, the program selector has no "end" or "quit" command. Instead, to end the program-selector program you must make sure that all of your programs and commands have ended, remove your floppy disks, and turn off your computer.

In the next chapter

In this chapter, you have learned how to start MS OS/2 and how to use the program selector to run multiple sessions. The rest of this manual will teach you how to use MS OS/2 commands. In Chapter 3, you will learn about special MS OS/2 commands, called batch commands, that are used primarily in batch files.

3 Using Batch Files

You may often find yourself typing the same sequence of commands to perform a common task. With Microsoft Operating System/2, you can create a special file called a batch file to perform a set of commands. Then, instead of typing several commands, you type only one.

In this chapter you will learn the following:

- How to create and use batch files
- How to use parameters in a batch file

Note You need to read this chapter only if you are creating batch files.

Learning About Batch Files

Instead of typing commands one by one, you can use batch files to group MS OS/2 commands together. When you type the name of your batch file, MS OS/2 performs all of the commands listed in the file as though you were typing each command from the keyboard. This is called batch processing because MS OS/2 performs commands in batches rather than singly. 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 sets of commands.

In real mode, all batch filenames must have an extension of .BAT. In protected mode, batch filenames always have a .CMD extension. You need only type the filename of your batch file, but not the .BAT or .CMD extension, to run it.

For example, if you created a batch file in real mode called GATHER.BAT, you would type the following command to run that batch program:

```
gather
```

Three useful batch files

MS OS/2 uses three useful batch files:

- The AUTOEXEC.BAT file lets you run programs and commands automatically when you start the real-mode session for the first time.
- The STARTUP.CMD file lets you run programs and commands automatically when you start the first protected-mode session after booting MS OS/2.
- The INITENV.CMD file lets you run a set of commands or programs for each protected-mode session as you start each session.

For more information about using these files, see the *Microsoft Operating System/2 Setup Guide.*

Here are a few things you should know before you run a batch process with MS OS/2:

- Each real-mode batch file must have an extension of .BAT.
- Each protected-mode batch file must have an extension of .CMD.

- To execute a batch file, type only its filename and not its extension.
- In protected mode, if you press CONTROL+BREAK or CONTROL+C while the batch file is running, MS OS/2 immediately stops running the batch file, and the system prompt appears on your screen.
- In real mode, if you press CONTROL+BREAK or CONTROL+C while the batch file is running, MS OS/2 displays a message asking if you want to terminate the batch job. If you type **y** (for Yes), the batch job stops running. If you type **n** (for No), the batch job continues.
- If you remove the disk that contains a batch file being run, MS OS/2 prompts you to reinsert the disk so that it can continue processing the file.
- You can specify the name of another batch file as a command in a batch file. This feature lets you call one batch file from another when the first has finished. In protected mode, you can call another batch file (with the .CMD extension) from the middle of the first batch file. When the second protected-mode batch file is completed, you can return to the first batch file to continue. Note that you cannot call real-mode batch files (.BAT) from protected-mode batch files (.CMD), or vice versa.
- You can use the pipe symbol (|) or any of the redirection symbols (<, >, >>) in a batch file. For more information on using these symbols, see Chapter 6, "Using MS OS/2 Commands."
- You can use the at symbol @ at the front of a command line in a batch file to prevent that line from appearing on your screen.
- Commands in the batch file that set the directory or drive affect every subsequent command in the batch file.
- Commands in the batch file that set environment strings (such as the current search path) also affect every subsequent command in the batch file.

Note If you have two or more files with the same filename but different extensions, MS OS/2 will run only one of them. Files with the .COM extension run before .EXE files. If MS OS/2 cannot find a file with either a .COM or .EXE extension, MS OS/2 looks for a batch file with a .BAT or .CMD extension.

For example, suppose that your disk includes the files FORMAT.EXE and FORMAT.BAT in the same directory. If you were to type the external command **format**, MS OS/2 would always run the program FORMAT.EXE first. To run the real-mode batch file FORMAT.BAT, you would have to place it in a different directory and type a path along with the filename to run it.

As an example, to run a file called FORMAT.BAT in a directory named \COMMAND\BATCH you would type the following from the real-mode prompt (C>):

```
\command\batch\format
```

A simpler alternative would be to rename your batch file to FORM to avoid possible confusion. Then, to run your batch program, you would need to type only the batch filename:

```
form
```

Creating a Batch File

You can create a batch file by using one of the following:

- A line editor like Edlin
- The **copy** command
- A word processor that saves files as ASCII text

For more information about creating files with Edlin, see Chapter 5, "Using Edlin."

Creating a batch file with Copy

The examples in this chapter show you how to use the **copy** command to create batch files. To create a batch file using the **copy** command, use the following steps:

1. Type **copy con** followed by the filename and extension of the batch file. Then press the ENTER key. Remember that the extension for batch files in real mode is .BAT. The extension for batch files in protected mode is .CMD.

2. Type the contents of the batch file. Press ENTER to start each new line.
3. After the last line, press CONTROL+Z. Then press the ENTER key to save the batch file.

For example, suppose that you want to create a batch file in protected mode called CHECKNEW.CMD that will format and check a new disk. To create this file, you simply follow these steps:

1. Make sure that you are in a protected-mode session to create a protected-mode batch file. In protected mode, you will see a prompt like this:

```
[C:\]
```

2. Type the following command and press the ENTER key:

```
copy con checknew.cmd
```

   This command copies the information from the console (keyboard) into the file CHECKNEW.CMD.
3. Type the contents of the file exactly as it should appear. If you make a mistake on a line, use the BACKSPACE key to correct it. (If you have already completed a line and pressed ENTER, you cannot go back to edit that line using this method.)

   Type the following lines:

```
rem  This is a file to format and
rem  check new disks.
rem  It is named CHECKNEW.CMD.
pause  Insert new disk in drive B:
format b: /v
chkdsk b:
```

4. After the last line, press CONTROL+Z. Then press ENTER to save the batch file. MS OS/2 displays the message "1 File(s) copied" to show that it created the file.

Now, to execute the file in protected mode, simply type the following command:

```
checknew
```

The result is the same as if the lines in the batch file had been entered from the keyboard as individual commands.

Using Parameters

You have learned that batch files save you time by allowing you to put several commands into a single batch file. For example, you could use the following batch file to sort and display the contents of a file called RERUNS.SHO:

```
type reruns.sho | sort
```

Using parameters in batch files

Now, suppose you want to sort and display the contents of other files. Rather than create several batch files naming different files to sort and display, you can create one file by using a parameter. A parameter is a placeholder that lets you name something else to put in its place. The ten parameters that MS OS/2 uses are *%0–%9*. The following batch file, called NEWSORT.BAT, sorts and displays any file you name:

```
type %1 | sort
```

For example, to sort and display the contents of a file named STAGE.SHO using the NEWSORT.BAT file, you would type the following command:

```
newsort stage.sho
```

To sort and display a file called SCREEN.SHO, you would type this command:

```
newsort screen.sho
```

When you execute the NEWSORT.BAT batch file, MS OS/2 replaces the *%1* placeholder with the parameter values you supply. If you had a file with two parameters, MS OS/2 would replace *%1* with the first parameter value you supply and *%2* with the second.

You can specify up to nine parameters *(%1–%9)* in a batch file. The parameter *%0* is always replaced with the drive name (if specified) and the filename of the batch file itself.

If you want to specify more than nine parameters in a batch file, use the **shift** batch command. For more information about **shift**, see Chapter 7, "MS OS/2 Command Reference."

Note 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 DANCER%.EXE, you must type it as DANCER%%.EXE in the batch file.

Using Named Variables

You can use named variables in protected-mode batch files. Named variables differ from parameters in that you do not specify their values on the command line. Instead, MS OS/2 retrieves the value of each named variable from its environment. A variable name begins and ends with a percent sign and has from one to six letters.

You can use the MS OS/2 **set** command to set the value of a variable before you run your batch file, or you can include the **set** command in your batch file.

For example, suppose that you want to create a batch file called MYDEL.BAT that moves a file you want to delete into a separate directory. You might want to use this method to make sure that you don't accidentally delete files from an important directory.

The MYDEL.BAT file might contain the following lines:

```
echo off
echo Before you use this batch file, you must
echo specify the directory by typing the
echo following command at the MS OS/2 prompt:
echo set deldir=directory
echo Press Control+C to exit if you haven't
echo set deldir or if deldir does not exist.
pause
copy %1 %deldir%
del %1
dir /w %deldir%
echo All done.
```

Typing the following command line sets the directory name on drive C to DELETED:

```
set deldir=c:\deleted
```

Then, to move the file REPORT23.JUN to the DELETED directory, type the following:

```
mydel report23.jun
```

The batch file automatically replaces the *%deldir%* variable with the directory name DELETED.

Note The DELETED directory must exist for the MYDEL.BAT file to work. For more information about creating directories, see the **mkdir** command in Chapter 7, "MS OS/2 Command Reference."

Using Temporary Files

Using temporary files with batch files

There is no limit to the number of commands you can put into a batch file. But if your batch file includes several commands, you may need to use temporary files to hold your work. These work files hold the output of one command for use by the next command in the batch file. When the batch program is completed, the work files are no longer needed, so you should delete them. You can include a command at the end of your batch file to automatically delete these temporary work files.

Note Deleting temporary files is important. You can save space on your disks by removing unnecessary files.

Be careful not to use the same name for all of your temporary work files. 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.

A sample batch file

The following is an example of a batch file called SORTER.BAT that sorts a file containing a specific string of characters:

```
type %2 ¦ find "%1" > %3
type %3 ¦ sort > prn
del %3
```

Each time you run the SORTER batch file, you can tell MS OS/2 the following information:

- The string you want MS OS/2 to search for (parameter1, *%1*)
- The file to search to find that string (parameter2, *%2*)
- The name of a temporary file to use for sorting (parameter3, *%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:

```
Sheppard, Megan          north
Moynihan, Ann            south
Sharpe, Isabel R.        north
Fisher, Pete             east
Conrad, Betty            south
Mack, Mark               north
Shaw, Rick               west
Moss, Melissa            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 parameter values, by typing the following command line and then pressing the ENTER key:

```
sorter north a:customer temp.fil
```

The output on the printer should look like this:

```
Mack, Mark               north
Moss, Melissa            north
Sharpe, Isabel R.        north
Sheppard, Megan          north
```

The following table shows how MS OS/2 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 their parameter values, as follows:

```
type a:customer ¦ find "north" > temp.fil
type temp.fil ¦ sort > prn
del temp.fil
```

For a complete listing of batch commands, see Chapter 7, "MS OS/2 Command Reference."

In the next chapter

In Chapter 4, you will learn how to use MS OS/2 editing keys as a shortcut to typing long commands.

4 MS OS/2 Editing Keys

Microsoft Operating System/2 saves the last command you typed in a storage area so that you can use it again. When you type a command and press the ENTER key, MS OS/2 performs the command and also stores the command in a template. With the special MS OS/2 editing keys, you can use this command template to repeat commands. You can also use the editing keys to make quick revisions to the command in the template.

In this chapter, you will learn the following:

- How to use MS OS/2 editing keys
- What a command template is and how to use it
- How to use control characters to affect commands

Many operating systems handle command input differently than MS OS/2 does. One difference in particular that sets both MS-DOS and MS OS/2 apart from other operating systems is a set of special editing keys that MS-DOS and MS OS/2 use. For instance, with MS OS/2 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 OS/2 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.

When you type a command and press the ENTER key, MS OS/2 automatically sends the command to the command processor (COMMAND.COM or CMD.EXE) for execution. At the same time, MS OS/2 also saves a copy of this command in the template. You can then recall or modify the command by using the MS OS/2 editing keys.

Using MS OS/2 Editing Keys

Special editing functions

The MS OS/2 editing keys provide you with a set of editing tools that can save you time. You can use them to correct typing mistakes, repeat frequently used commands, or create similar command lines.

These editing keys are described briefly in the following list, and more fully in Chapter 5, "Using Edlin":

Key	Function
F1	Copies one character from the template to the command line. You can also press the RIGHT key to perform the F1 function.
F2	Copies the characters in the template up to the character specified and puts these characters on the command line.
F3	Copies all remaining characters in the template to the command line.
DELETE	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.
ESCAPE	Cancels the current input and leaves the template unchanged.
INSERT	Enters/exits insert mode.
F5	Makes the new line the new template.
F6	Puts a CONTROL+Z end-of-file character (1AH) in the new template.
BACKSPACE	Deletes a character from the command line and moves the cursor back one character space in the template. You can also use the LEFT key or CONTROL+H to perform the BACKSPACE function.

Using the MS OS/2 Template

Creating a template

When you type a command on the command line, MS OS/2 copies that command into the template area of memory. You can use MS OS/2 editing keys like F1 and F3 to reuse all or part of the template. Or, you can modify the template to create a new command with MS OS/2 editing keys like INSERT and DELETE.

As an example, suppose you want to see the directory information for a file named INVEST.MNT. To get this information, you would type the following command:

```
dir invest.mnt
```

This command line is saved in the template. If you want to repeat the command, just press two keys: F3 and ENTER. MS OS/2 displays the repeated command and executes it.

Notice that when you press the F3 key, MS OS/2 copies the contents of the template to the command line; pressing the ENTER key 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 to create the command line. Pressing F2 then typing **m** copies all characters from the template to the command line, up to but not including "m." MS OS/2 then displays the following:

```
dir invest._
```

The underline is the cursor. Now type **rpt** to get the following result:

```
dir invest.rpt_
```

Changing the template

To run the command and save the new command line in the template, press the ENTER key.

Now, suppose you want to review the contents of the INVEST.RPT file by using the following command:

```
type invest.rpt
```

To do this, type **type** and then press the following sequence of keys: INSERT, SPACEBAR, F3, ENTER.

Notice as you type that the characters appear directly on the command line, overwriting their corresponding characters in the template; "type" replaces "dir" in the template. But note that the space following "dir" is overwritten by the letter "e" in "type." When you press the INSERT key, this automatic replacement feature is turned off.

To insert a space between "type" and the filename INVEST.RPT, you pressed INSERT and then the SPACEBAR. Finally, to copy the rest of the template to the command line and run the command, you pressed F3 and then the ENTER key.

The command line **type invest.rpt** is then processed by MS OS/2, and the template now looks like this:

```
type invest.rpt
```

Correcting errors in the template

If you had misspelled "type" as "pyte," MS OS/2 would have displayed an error message. Still, instead of retyping the entire command line, you could save the misspelled line before pressing the ENTER key. You do this by pressing the F5 key *before* the ENTER key, creating the following template:

```
pyte invest.rpt
```

Then you can correct this misspelling by using the following key sequence (do not type the commas):

DELETE, DELETE, F1, INSERT **yp**, F3

The following list illustrates how the keys you press affect the command line; the result of the keystrokes is shown in parentheses following the keyname:

Key (Result)	Description
DELETE (_)	Skips over the first template character.
DELETE (_)	Skips over the second template character.
F1 (t_)	Copies the third template character.
INSERT **yp** (typ_)	Inserts two characters, "y" and "p."
F3 (type invest.rpt_)	Copies the rest of the template.

Notice that pressing DELETE does not affect the command line. Instead, it affects the template by deleting the first character.

MS OS/2 Control Characters

In addition to editing keys, MS OS/2 has control characters that help you control the output from a command, or control the contents of the current command line. A control character is produced by pressing and holding down the CONTROL key while you press and release another key. A control character affects the command line in a special way. For example, you press CONTROL+C to stop running the current command, and you press CONTROL+S to suspend the screen output from a command.

The following list contains the MS OS/2 CONTROL+*key* combinations and describes what they do:

Key combination	Function
CONTROL+C	Stops the command that is running.
CONTROL+H	Removes the last character from a command line, and erases that character from the screen.
CONTROL+J	Inserts a physical end-of-line character, 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+P	Sends output to a lineprinter as you type it on the screen. Press CONTROL+P again to cancel the printing.
CONTROL+S	Suspends output display on the screen. Press CONTROL+Q to resume.
CONTROL+X	Cancels the command line you are typing, but does not affect the template used with the editing keys.

Note Some of the control characters listed in this section may not be supported on all hardware.

In the next chapter

In Chapter 5, you will learn how to use MS OS/2 editing keys with the real-mode line editor, Edlin. Chapter 5 also explains how to use the Edlin commands to edit text files in real mode.

5 Using Edlin

Edlin is the Microsoft Operating System/2 real-mode line editor that you can use to create text files and save them on your disks. Edlin also helps you to update existing files by deleting, changing, and inserting lines in files. And, though it isn't a word processor, Edlin does make it easy for you to create and revise files memos, letters, reports, or BASIC programs.

In this chapter you will learn the following:

- How to start Edlin
- How to end Edlin and save edits
- How to use the MS OS/2 special editing keys with Edlin
- How to use Edlin commands

How Edlin Works

Edlin divides the text from a file into lines, with 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.

Starting Edlin

How to start Edlin

To start Edlin, type a command of the following form:

edlin *filename*

The *filename* placeholder is the file you want to edit. If you are creating a new file, *filename* should be the name or pathname of the file you want 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, to create a file called BUDGET.JUN, you would type the following command and then press the ENTER key:

```
edlin budget.jun
```

Once you type the command, Edlin displays the following:

```
New file
*_
```

Note that the Edlin prompt is an asterisk (*).

To begin entering text you must type the **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, which are discussed in more detail later in this chapter.

Note Be sure to press the ENTER key at the end of each line.

Editing an Existing File

How to edit an existing file

Suppose you want to edit an existing file called BUDGET.MAY. To do this you would type the following:

```
edlin budget.may
```

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 a disk into memory. For more information on editing large files, see the sections describing the **w** (write) and **a** (append) commands later in this chapter.

Ending Edlin

How to end Edlin and save your changes

When you finish your editing session and the cursor is at the asterisk (*) prompt, you can save your original file and the updated (new) file by using the **e** (end) command, discussed later in this chapter. Edlin renames your original file with the .BAK extension and saves the updated file with the filename and extension you gave when you started Edlin.

Note You cannot update a file with an extension of .BAK because when you try to save your file, Edlin always saves the original file as a .BAK file, overwriting any existing .BAK file. Thus, any changes you made to the existing .BAK file using Edlin are lost. If you need to update a .BAK file, rename it with another extension (using the MS OS/2 **ren** command discussed in Chapter 7, "MS OS/2 Command Reference"), and start Edlin by using the new filename.

In addition, if a .BAK file already exists and has read-only permission, the updated file will be saved by using the **e** (end) command, but the original file will not be saved in a .BAK file. Edlin detects this condition when you first start to edit a file and displays the following message:

```
WARNING! Backup is Read-only -- backup
will not be made
```

Using MS OS/2 Editing Keys with Edlin

Edlin editing keys

You can use seven of the MS OS/2 editing keys with Edlin. For more information about MS OS/2 editing keys, see Chapter 4, "MS OS/2 Editing Keys." The following list summarizes the function of each key:

Key	Purpose
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.
DELETE	Does not copy (skips over) a character.
F4	Does not copy (skips over) the characters in the template, up to the character specified.
ESCAPE	Clears the current input and leaves the template unchanged.
INSERT	Enters/exits insert mode.
F5	Makes the new line the new template.
BACKSPACE	Deletes a character from the command line and moves the cursor back one character in the template.

The next few pages describe how to use each of the MS OS/2 editing keys with Edlin.

The F1 Key

Copying one character

When you press the F1 key, Edlin copies one character from the template to the current line, and turns the insert mode off. As an example of how to use the F1 key with Edlin, type the following lines:

```
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 (the result is shown below the keyname):

```
F1
2:*S_
```

Each time you press the F1 key one more character appears:

```
F1
2:*Sh_

F1
2:*Sha_

F1
2:*Shar_
```

The F2 Key

Copying multiple characters

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 the insert mode.

For example, if you type **Sharpe Office Supplies** on line 1, press F2, then type **c**, Edlin copies the characters up to the "c" in the word "Office."

```
F2 c
2:*Sharpe Offi_
```

The F3 Key

Copying the template

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. As an example of how to use the F3 key with Edlin, if you type **Sharpe Office Supplies** on line 1 and press F3, Edlin copies the characters in the template (from line 1) to the line with the cursor (shown in line 2):

```
F3
2:*Sharpe Office Supplies_
```

This command automatically turns off the insert mode.

The DELETE Key

Skipping one character

Each time you press the DELETE key, Edlin skips over and does not copy the next character in the template. The action of the DELETE key is similar to the F1 key, except that DELETE skips a character in the template instead of copying it to the current line. Thus, if the template contains "Sharpe Office Supplies" and you press the DELETE key, Edlin skips over the first character, "S." The cursor does not move as Edlin changes the template. So 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:

```
DELETE
F3
2:*harpe Office Supplies_
```

The F4 Key

Skipping multiple characters

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. Using the same template example, "Sharpe Office Supplies," you can press the F4 key then type **c** to skip over all the characters in the template up to the "c" in the word "Office." 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:

```
F4 c
F3
2:*ce Supplies_
```

The ESCAPE Key

Clearing current input

When you press the ESCAPE key, Edlin clears the current line and leaves the template unchanged. ESCAPE also prints a backslash (\), activates the ENTER and LINEFEED keys, and turns off insert mode. The cursor 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 ESCAPE key. For example, create the following lines:

```
1: Sharpe Office Supplies
2:*The World Leader_
```

To cancel the current line, line 2, press ESCAPE. Notice that a backslash appears on line 2 to tell you it has been canceled:

```
ESCAPE
2:*The World Leader\
```

Press the ENTER key to save line 1 or to perform any other editing functions. If you press F3, Edlin copies the original template to the line:

F3
```
2:Sharpe Office Supplies
```

The INSERT Key

Entering insert mode

The INSERT key toggles between insert mode and replace mode. When you start Edlin, you are automatically in replace mode. The first time you press the INSERT 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 INSERT key again, Edlin re-enters replace mode with the cursor at the same character in the template as when you entered insert mode.

The F5 Key

Creating a new template

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 activates the ENTER and LINEFEED keys. When you press F5, Edlin empties the current line and turns off insert mode.

Note F5 performs the same function as the ESCAPE key, but it changes the template, printing an at symbol instead of a backslash.

As an example of how to use the F5 key with Edlin, create 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 press the following sequence of keys and type the following words (the results are shown below each key sequence):

```
F2 c
2:*Sharpe Offi_

INSERT cial
2:*Sharpe Official_

INSERT Sharpeware
2:*Sharpe Official Sharpeware_
```

If 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@
```

The at symbol (@) shows that this new line is the new template. To add "Introducing:" followed by a space at the beginning of the line, press INSERT and type the following (the results are shown below each key sequence):

```
INSERT Introducing:
2:*Introducing: _
```

Then press the F3 key to insert the contents of the template:

```
F3
2:*Introducing: Sharpe Official Sharpeware_
```

The BACKSPACE Key

Deleting one character

The BACKSPACE key deletes a character and places the cursor back one character in the template. For example, suppose that you typed the following:

```
1:*Sharpe Office Supplier_
```

Then you realize that you meant to type "Supplies" instead of "Supplier." To correct this error, you would press the BACKSPACE key and type **s**:

```
BACKSPACE s
1:*Sharpe Office Supplies_
```

Note You can also press the LEFT key or CONTROL+H to backspace.

Using Edlin Commands

Edlin commands

Edlin includes several commands that help you to edit the lines of your file. The following list summarizes these commands:

Command	Purpose
a	Appends lines from disk to memory.
c	Copies lines.
d	Deletes lines.
line	Edits the line number or numbers specified.
e	Ends the editing session and saves edits.
i	Inserts lines of text.
L	Lists a range of lines.
m	Moves a range of text to a specified line.
p	Pages through a file 23 lines at a time.
q	Quits the editing session without saving the file.
r	Replaces text.
s	Searches for text.
t	Transfers the contents of another file into the file being edited.
w	Writes specified lines from memory to disk.

Note A capital L is used in this chapter for the **L** (list) command to avoid confusion with the number one. A lowercase l works just as well.

Each of these commands is discussed in detail later in this chapter.

Some Tips for Using Edlin Commands

Editing tips

Once you have started editing a file with Edlin, you can use the Edlin commands to edit lines of text in the file. 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 refer to lines by using numbers relative to the current line (identified 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. For example, to list 10 lines before the current line, the current line, and 10 lines after the current line, you could type this command:

  ```
  -10,+10L
  ```

  Edlin ignores spaces between the line number and command.

- Generally, Edlin allows you to type one command after another on the same command line. However, if you want to use the Edlin *line* (edit) command to edit a specific line, you must separate the line number from the other commands with a semicolon. For example, the following command makes line 15 the current line and displays lines 10 through 20 on the screen:

  ```
  15;-5,+5L
  ```

Using control characters

- When using CONTROL+*key* sequences, simply press and hold down the CONTROL key while you press and release the appropriate key (such as Z or C or V). For example, if you want to search for a phrase that includes the control character CONTROL+Z, you could use a command like this one:

  ```
  smonthly budget CONTROL+Z -5,+5L
  ```

  Note that you do not type the characters "CONTROL+Z" but instead press the CONTROL key

as you press the Z key. For more information about control characters, see Chapter 4, "MS OS/2 Editing Keys."

- You can insert a control character, such as CONTROL+C, into text by using the quotation mark character, CONTROL+V, before it while in insert mode. CONTROL+V tells MS OS/2 to recognize the next *capital* letter typed as a control character. For example, the following command line finds the first occurrence of CONTROL+Z (the end-of-file mark) in a file:

```
s CONTROL+V Z
```

  To insert CONTROL+V into the text, press CONTROL+V and type **V**.

- The CONTROL+Z character is usually an end-of-file identifier for Edlin. 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 when you start Edlin. For example, the following command lets you start editing the file MACRO.ASM and ignores any CONTROL+Z characters:

```
edlin macro.asm /b
```

Edlin Command Options

Using command options with Edlin

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 the *line* option in one of three ways:

Line option	Description
linenumber	You may use any number less than 65,534. If you specify a number larger than the largest existing line number, then *linenumber* refers to the line after the last line number.
period (.)	If you specify a period for *line*, it refers to the current line number. The current line is the last line you edited, not necessarily the last line you displayed. Edlin marks the current line with an asterisk (*) between the line number and the first character.
number sign (#)	The number sign indicates the line after the last line number in the file. If you type # for *line*, it is the same as typing the last line number plus one.

Note If you type a command and then press the ENTER 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 **y** or press the ENTER 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.

Understanding Edlin Commands

The remaining pages in this chapter describe the Edlin commands. Each description includes the correct use (syntax) of a command. Each command description also includes comments and examples that offer advice, help, and some shortcuts for using Edlin.

Edlin's Append Command: a

Appending a line

Syntax:

[*n*]**a**

Comments:

If you are editing a large file that is too large to read into memory all at once, you can use the **a** (append) command. This command lets you read in portions of your file to memory as you need to work on them. The *n* parameter is the number of lines that you want to read into memory.

When you start Edlin, it reads as many lines as possible into memory. If the size of your file exceeds available memory, you must edit your file in stages. That is, after you have edited the first part of a large file, you must write lines that you have already edited to your disk. Then you can load unedited lines from your disk into memory by using the **a** command.

Notes:

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. If available memory is already full, you may be able to free memory by quitting other real-mode applications or by restarting MS OS/2. Restarting MS OS/2 clears memory that is being used by real-mode programs that remain resident in memory even after they finish running.

After the **a** command reads the last line of the file into memory, Edlin displays the message "End of input file."

Example:

Suppose you have a file so large that the last 100 lines cannot fit into memory. After editing the first part of the file, and writing a portion back to the disk, you could use the following command line to read the remaining 100 lines into memory:

```
100a
```

For information about how to write edited lines to your disk, see the **w** (write) command in this chapter.

Edlin's Copy Command: c

Syntax:

[*line*],[*line*],*line*[,*count*]**c**

Comments:

The **c** (copy) command 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. 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 places the copied lines.

You must not overlap the line numbers or you will get an "Entry error" message. For example, this 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 once and automatically renumbers the file after the copy.

Examples:

The following command line copies lines 1 through 5 and duplicates them one at a time, beginning on line 6; thus lines 1 through 5 and 6 through 10 are identical:

```
1,5,6c
```

Deleting lines

Edlin's Delete Command: d

Syntax:

[*line*][,*line*]**d**

Comments:

The **d** (delete) command deletes a specified range of lines in a file. If you omit the first *line* option, Edlin uses the current line as the default line. If you omit the second *line* option, Edlin deletes just the first *line*. Remember, too, that when you delete lines, Edlin automatically renumbers the file.

Examples:

If you type the following command line, Edlin deletes line 7 then renumbers line 8 and all following lines:

```
7d
```

If you want to delete a block of text on lines 22 through 32, you should type the following:

```
22,32d
```

Finally, suppose that you want to delete a range of lines beginning with the current line, line 7, through line 11. Type the following:

```
,11d
```

Edlin's Line Edit Command

Editing specified lines

Syntax:

[*line*]

Comments:

The *line* option allows you to specify the line number 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 ENTER key.

If you press the ENTER key without typing a line number, Edlin lets you edit the line after the current line.

When you have edited the line, simply press the ENTER key to accept the line.

Warning If you press the ENTER 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.
```

Suppose that you want to insert the product's name, X-1000, in line 5. To edit line 5, type **5**. Edlin then displays the contents of the line with the cursor below the line:

```
5:*shock from our Automatic
5:*_
```

Now you simply press the F2 key and type **A** to skip to the "A" in the word "Automatic," then press the INSERT key and type **X-1000** (the result appears below the key sequence in the following example):

```
F2 A
INSERT X-1000
5:*shock from our X-1000
```

To view the edited line, press F3 and then ENTER:

```
F3 ENTER
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.
```

Edlin's End/Save Command: e

Ending and saving edits

Syntax:

e

Comments:

The **e** (end) command saves the edited file on your disk, renames the original input file *filename*.BAK, and then exits from Edlin. If you created the file during this editing session, Edlin does not create a backup (.BAK) file.

The **e** command takes no options. This means that you must select the directory that you want to save the file in *when you start* Edlin. If you don't select a directory when you start Edlin, 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 OS/2 **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 and some or all of the edited file will be lost.

Notes:

If a .BAK file already exists and is a read-only file, the updated file is saved by using the **e** command. However, the original file is not saved in a .BAK file. MS OS/2 detects such a condition when you first start to edit the file and displays this message:

```
Warning: Backup file is read only -- backup
will not be made.
```

Example:

To end an Edlin session and save the edits you have made, simply type the following:

```
e
```

Edlin saves your edited file and returns you to MS OS/2.

Inserting lines of text

Edlin's Insert Command: i

Syntax: [*line*]**i**

Comments:

The **i** (insert) command allows you to insert text immediately before the specified *line*. If you are creating a new file, you must type the **i** (insert) command before you can insert a line of text. Text begins on line 1, and the next line number appears automatically each time you press the ENTER key.

Edlin remains in insert mode until you press CONTROL+C. When you finish the insertion and exit from insert mode, the line immediately following the inserted lines becomes the current line. Edlin automatically renumbers the lines that follow the inserted section.

If you do not specify *line*, Edlin inserts the lines before the current line. If *line* is a number larger than the last line number, or if you specify a number 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
```

Inserting text in a file

To insert text before line 8, type **8i**. The result is as follows:

```
 8:*_
```

Now type the following lines, which will begin on line 8, pressing the ENTER key after each line:

```
 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.
```

To end the insertion, press CONTROL+C on the *next* line.

Inserting a blank line in a file

To insert a blank line immediately before the current line (line 12), first type **i**. The result is as follows:

```
12:*_
```

Then insert a blank line by pressing ENTER. End the insertion by pressing CONTROL+C on the next line. 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
```

Listing a range of lines

Edlin's List Command: L

Syntax:

[*line*][,*line*]**L**

Comments:

The **L** (list) command displays a range of lines, including the two lines specified. If you only specify one of the line options, Edlin uses default values. For example, if you omit the first *line* option, as in the following example, Edlin displays 23 lines, beginning 11 lines before the current line and ending with the specified *line*:

,*line* **L**

The beginning comma shows that you omitted the first *line* option. If you omit the second *line* option, Edlin displays 23 lines, starting with the specified *line*. If you type **L** with no *line* option at all, Edlin displays 23 lines—beginning with the 11 lines before the current line.

Notes:

- 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.
- An uppercase L has been used here to avoid confusion with the number 1. A lowercase l would work just as well.

Example:

For example, to list lines 5 through 10, inclusively, type the following:

```
5,10L
```

Edlin's Move Command: m

Moving specific lines

Syntax:

[*line*,][+]*line*,*line*m

Comments:

The **m** (move) command lets you transfer 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 to which you want to move the first line in the range.

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.

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: 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

If you would prefer to have the motto at the start of the letter, you could move lines 16—18 to line 1 by typing the following command line:

```
16,18,1m
```

The result of this command is as follows:

```
 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
```

Edlin's Paging Command: p

Syntax:

[*line*][,*line*]**p**

Comments:

The **p** (page) command displays a file one screen (23 lines) at a time. The first *line* option specifies the first line of the display. The second *line* option specifies how many lines appear on the screen. 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 displays 23 lines on the screen.

Example:

To view lines 100 through 200 and to see the text one screen at a time, you would type this command:

```
100,200p
```

Quitting an editing session

Edlin's Quit/No-Save Command: q

Syntax:

q

Comments:

The **q** (quit) command is useful if you don't want to make any changes to a file. This command returns you to MS OS/2 and does *not* save any editing changes. If you use the **q** (quit) command, Edlin prompts you to make sure you don't want to save the changes. If you want to save changes as you exit from Edlin, use the **e** (end) command.

Notes:

When you exit from Edlin using the **e** command, Edlin erases any previous copy of the file that has a .BAK extension. But if you quit Edlin (**q**) and type **y** (for Yes) in response to the "Abort edit (Y/N)?" message, Edlin does not delete your previous backup copy.

Example:

The following example shows how to quit Edlin without saving your changes:

1. Press CONTROL+C to leave insert mode.
2. At the asterisk (*) prompt, type **q**. The message "Abort edit (Y/N)?" is displayed.
3. Type **y** (for Yes) to abort the edit, then press the ENTER key.

Edlin's Replace Command: r

Replacing text

Syntax:

[*line*][,*line*][?]**r** *text1* CONTROL+Z *text2* CONTROL+Z

Comments:

The **r** (replace) command replaces all occurrences of a string of text in a range with a different string of text. The *line* options show the range that **r** uses. Each time Edlin finds *text1*, it replaces it with *text2*. Then Edlin displays each line that changes.

For example, the following command would change the word "mine" each time it occurred in a 20-line file to the word "ours":

```
1,20rmine CONTROL+Z ours
```

Note that you press CONTROL+Z; do not type the characters "CONTROL+Z."

Approving each replacement

If a line contains two or more replacements, it is displayed once for each change. If you include a question mark (?) in your command, Edlin asks you to confirm each replacement. If you type **y** (for Yes) or press the ENTER key, *text2* replaces *text1*, and Edlin looks for the next occurrence of *text1*. If you press any other key in response, Edlin does not make the change for that occurrence of *text1*. When Edlin has made all the changes, the asterisk prompt reappears.

When you do not specify *text1*, the **r** command assumes the old (any previous) value. If this is the first replacement that you have done during this 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 ENTER 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 the line following the last line of the file (represented by the number symbol, #.

If you end *text1* by pressing 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 would simply type the following command, press CONTROL+Z, and then press ENTER key:

```
rclients
```

The next command replaces "clients" with the previous *text2*:

```
rclients
```

The following command makes the previous *text1* become the previous *text2*:

```
r
```

Note that "previous" refers to an earlier string of text specified in an **s** or **r** command.

Example:

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 ENTER key. The result is the following:

```
 5: shock from the X-1000 Automatic
 8: As a result of ythe accident, we
 9: are redesigning the manual to
10: warn the customers against trying
```

Replacing text selectively

In the previous example, some unwanted changes occurred. In the next example you will see how to replace only certain occurrences of "our" with "the" in the original file. At the Edlin prompt, include the following command line, and then press the ENTER key:

```
1,15? rour CONTROL+Z the
```

The result is as follows:

```
 5: shock from the X-1000 Automatic
O.K.? y
 8: As a result of ythe accident, we
O.K.? n
 9: are redesigning the manual to
O.K.? y
10: warn the customers against trying
O.K.? n
*_
```

Type the list command, **L**, to see the result of 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
   .
   .
```

Searching for text

Edlin's Search Command: s

Syntax:

[*line*][,*line*][**?**]**s***text*

Comments:

The **s** (search) command searches a range of lines for a string of text. The first and second *line* options specify the range of lines for Edlin to search. You end the *text* option by pressing the ENTER 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** 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 type **y** (for Yes) or press the ENTER 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 do not type the first line number, Edlin uses the line after the current line as the default; and if you do not type the second line number, it uses the line after the last line of the file as the default.

If you omit the *text* option, Edlin uses the text from any previous **s** or **r** command. If this is the first **s** or **r** command you have used this session, and you have not specified a search string, the **s** command ends.

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
```

Searching for a word in a file

To search for the first occurrence of the word "to," type the command line **2,12 sto** and press the ENTER key. Edlin displays the following lines:

```
 3: I was sorry to hear of your recent
```

To search through several occurrences of a string until the correct string is found, type the command line **1, ? sto**. The result is as follows:

```
 3: I was sorry to hear of your recent
O.K.?_
```

If you press any key (except Y or ENTER), the search continues, so type **n** (for No):

```
O.K.? n
```

The search then continues:

```
 4: hospitalization due to electrical
O.K.?_
```

Type **y** to terminate the search:

```
O.K.? y
*_
```

Transferring text from another file

Edlin's Transfer Command: t

Syntax:

[*line*]t*filename*

Comments:

The **t** (transfer) command puts the contents of a file into the file you are working in. Edlin inserts the file represented by *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 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
```

Edlin's Write Command: w

Writing lines to disk

Syntax:

[*n*]**w**

Comments:

The **w** (write) command writes a specified number of lines to disk. 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.

Notes:

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. After Edlin writes the line to your disk, it renumbers all of the lines so that the first remaining line becomes line number 1.

Example:

Suppose you have a file so large that the last 100 lines cannot fit into memory. After you edit the first part of the file, you could free up enough space to edit the last part of the file with this command line:

```
125w
```

6 Using MS OS/2 Commands

Microsoft Operating System/2 commands are as powerful as they are versatile. In addition to commands used with Edlin and the CONFIG.SYS file, there are over 60 commands you can use with MS OS/2. A few of these commands are designed for real mode only or for protected mode only. Some are designed for use primarily in batch files. With many commands, you can use redirection symbols to send output to a file or other specified location. In protected mode, you can use command-grouping symbols to combine more than one MS OS/2 command on a single command line.

In this chapter, you will learn the following:

- Which MS OS/2 commands work in real mode only or in protected mode only
- Which MS OS/2 commands will not work over a network
- How to identify the types of MS OS/2 commands by the symbols used in this guide
- How internal commands and external commands differ
- How to use redirection, pipe, and command-grouping symbols

Types of MS OS/2 Commands

Real-mode and protected-mode commands

You have learned already that MS OS/2 has two different operating environments—real mode and protected mode. Some MS OS/2 commands work only in one of these two modes. The following symbols indicate that a command may be used in only one of the two MS OS/2 modes:

['Real Mode Only' 'Protected Mode Only' icons]

The following commands work only in real mode:

append
assign
break
graftabl
join
setcom40

The following commands work only in protected mode:

ansi
detach
dpath
start

Commands that are not for network use

You will also see the following symbol associated with certain MS OS/2 commands. This symbol means that a command cannot be used over a network:

['No Network' icon]

The following MS OS/2 commands do not work over a computer network:

chkdsk
diskcomp
diskcopy
fdisk
format
label
recover
subst
sys

If you try to use these commands on a network, MS OS/2 displays the following error message:

```
Cannot command to a network device
```

The *command* placeholder is the name of the command you typed.

Batch commands

This symbol identifies batch commands. Batch commands are most commonly used from within batch files:

['Batch Command' icon]

Internal and external commands

These last two symbols distinguish internal MS OS/2 commands from external MS OS/2 commands:

['Internal Cmd' 'External Cmd' icons]

See the following two sections for information on these two types of commands.

Internal Commands

What are internal commands?

Internal commands are commands that are built into the command processor. Real-mode internal commands are built into COMMAND.COM; protected-mode internal commands are built into CMD.EXE. Because they are part of the command-processor file, internal commands do not appear when you list the directory of your MS OS/2 program disk.

When you start the first protected-mode session, CMD.EXE is loaded into memory, along with all of the protected-mode internal commands. Likewise, when you start the real-mode session, COMMAND.COM is loaded into memory. So, because the internal commands are resident in memory, MS OS/2 can perform them immediately.

The following is a list of the MS OS/2 internal commands. These commands are described in Chapter 7, "MS OS/2 Command Reference:"

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

Using pathnames

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 (*path*), MS OS/2 copies all the files that you specified in the first *pathname* into that directory, as in the following example:

```
copy \movies\video\*.* sales
```

- **del** *pathname* [...]

  If *pathname* is a directory (*path*), all the files in that directory are deleted. If you try to delete a directory, MS OS/2 displays the prompt "Are you sure (Y/N)?" Type **y** (for Yes) to complete the command, or **n** (for No) to stop the command. The following is an example of a **del** command line:

```
del \news\national
```

- **dir** *path* [...]

  The following command line displays the directory specified by *path*:

```
dir \event\sports
```

- **type** *path* [...]

  You must specify a *pathname* (or *filename*) for this command. MS OS/2 then displays this file on your screen in response to the **type** command. The following is a typical **type** comand line:

```
type \weather\norwest\precip.nov
```

Note The [...] in the command formats applies to protected mode only. This notation means you can type more than one pathname on the protected-mode command line. For example, the following command deletes files in the \DANCERS\BEARS directory and the \DANCERS\BALLET directory:

```
del \dancers\bears \dancers\ballet
```

External Commands

What are external commands?

Any filename with an extension of .COM, .EXE, .BAT, or .CMD is considered an external command. For example, files such as FORMAT.EXE and DISKCOPY.EXE are external commands. Because all external commands are also files, you can create new commands and add them to MS OS/2. 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 OS/2 will run only one of them, according to the following order of precedence: .COM, .EXE, .BAT (or .CMD).

Suppose, for example, that your working directory includes the files FORMAT.EXE and FORMAT.CMD. If you type the external command **format** in protected mode from this directory, MS OS/2 runs the program FORMAT.EXE. To run the protected-mode batch file FORMAT.CMD, you would have to place it in a separate directory and give a path along with the external command.

The following is a list of external commands. These commands are described in Chapter 7, "MS OS/2 Command Reference":

ansi	**fdisk**	**patch**
append	**find**	**print**
assign	**format**	**recover**
attrib	**graftabl**	**replace**
backup	**helpmsg**	**restore**
chkdsk	**join**	**setcom40**
cmd	**keyb**	**sort**
command	**label**	**subst**
comp	**mode**	**sys**
diskcomp	**more**	**tree**
diskcopy		**xcopy**

External commands and paths

Before MS OS/2 can run external commands, it must read them into memory from the disk. When you type an external command, MS OS/2 immediately checks your working directory to find that command. If it isn't there, you must tell MS OS/2 which directory the external command is in. You do this with the **path** command.

With the Install program, MS OS/2 automatically creates AUTOEXEC.BAT and INITENV.CMD files with **path** commands pointing to the directories that contain external command files: \OS2\BIN, \OS2\RBIN, and \OS2\PBIN. For more information about the AUTOEXEC.BAT and INITENV.CMD files, see the *Microsoft Operating System/2 Setup Guide.*

Redirecting Command Input and Output

Usually, MS OS/2 receives input from the keyboard and sends its output to the screen. However, you can redirect this flow of command input and output. For instance, you may want input to come from a file rather than the keyboard, and you may want output from a command to go to a file or printer instead of to the screen. You can also create pipes that let the output from one command become the input for another. For example, if you want to read a file one screen at a time, you can pipe output from the **type** command to the **more** command by using a command line with the following format:

type *filename* | **more**

How to Redirect Output

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

```
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 OS/2 creates it and stores your directory listing in it. If CONTENTS already exists, MS OS/2 replaces what is in the file with the new data.

Appending output

If you want to add one file to another (instead of replacing the entire file), you can use two greater-than signs (>>) to tell MS OS/2 to append the output of the command (such as a directory listing) to the end of a specified file. For example, the following command line appends your directory listing to an existing file named CONTENTS:

```
dir >> contents
```

How to Redirect Input

It is often useful to have input for a command come from a file instead of from the keyboard. This is possible in MS OS/2 by using a less-than sign (<) in your command. For example, suppose you had a file called NAMES that listed the names of several clients. If this file was not in alphabetical order, you could sort the file's contents into a new file called NAMELIST with the following command line:

```
sort < names > namelist
```

Filter Commands and Pipes

Three filter commands

A filter is a command that reads your input, transforms it in some way, and then displays it on your screen.

There are three MS OS/2 filters: **find**, **more**, and **sort**. Their functions are as follows:

Command	Action
find	Searches for text in a file.
more	Displays the contents of a file one screen at a time.
sort	Sorts the contents of a file alphabetically.

You can redirect the output from a filter into a file or use it as input for another filter by using pipes. Piping is done by separating commands with the pipe symbol (¦). The following command line, 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 as input to the **sort** command.

You can also use piping with redirection symbols if you want to send the output to a file. For example, the following command line creates a file named DIRECT.LST in your working directory:

```
dir ¦ sort > direct.lst
```

The DIRECT.LST file now contains a sorted listing of the working directory.

You can also specify a drive other than the default drive. For example, suppose 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
```

You can include several pipes in a command line. The following command line, for example, sorts your directory listing and shows it to you one screen at a time.

```
dir ¦ sort ¦ more
```

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

Protected-Mode Grouping Symbols

In this chapter you have seen how you can use the redirection and piping symbols (<, >, >>, ¦) to give MS OS/2 more than one command on a line. In protected mode you can use other symbols, called grouping symbols, to group two (or more) commands on one command line. The following sections describe how each of these symbols is used.

The And Symbol (&&)

Using the and symbol

The and symbol (&&) performs the command to the left of the symbol. If that command is successful, it performs the command to the right of the symbol. If the command on the left fails (produces an error), the command on the right is not executed.

For example, the following command displays the file BUDGET.DOC only if it exists in the current directory:

```
dir budget.doc && type budget.doc
```

The Or Symbol (||)

Using the or symbol

The or symbol (||) performs either the command to the left of the symbol or the command to the right. If the command on the left is successful, it does not perform the command on the right. If the command on the left fails (produces an error), then the command on the right is performed.

For example, the following command deletes either the file called LETTER, if it exists, or removes the directory called LETTER, if it exists:

```
del letter || rmdir letter
```

The Command-Separating Symbol (&)

Using the separating symbol

The command-separating symbol (&) performs the commands on both sides of the symbol, in order, from left to right.

The following command, for example, displays the directory of the disk in drive A followed by the directory of the disk in drive B:

```
dir a: & dir b:
```

The Escape Symbol (^)

Using the escape symbol

The escape symbol (^) lets you use special symbols as regular characters. If you type the escape symbol before a special symbol such as the command-separating symbol (&) or the pipe symbol (|), you remove the character's special meaning to MS OS/2.

For example, suppose you wanted to use the **echo** command to create a new file. You could type a command like the following:

```
echo hello>greetings
```

This command creates a one line file containing the word "hello" and names it GREETINGS. You can use the escape symbol (^) to remove the special meaning of the greater-than symbol (>), as in the following example:

```
echo hello ^>greetings
```

The following message appears on your screen in response:

```
hello>greetings
```

For more information on **echo**, see Chapter 3, "Using Batch Files."

Command Grouping

You can group internal and external commands by using parentheses. If your command line uses more than one grouping symbol, MS OS/2 executes the command groups in the following order:

Escape symbol (^)
Command groupers [()]
I/O redirectors (<, >, >>)
Pipe symbol (|)
And Symbol (&&)
Or symbol (||)
Command separator (&)

You can use the command-grouper symbols (parentheses) around any command group to ensure that MS OS/2 executes the commands in the proper order.

The following command lines, for example, check to see whether the file PRODUCTS exists and, if it does, they sort it and put the results in the PRODUCTS.LST file:

```
dir  products && type products | sort >
products.lst

dir  products && (type products | sort >
products.lst)
```

Because of page limitations, each command line appears on two lines; however, when you type these command lines, you should not press the ENTER key until you have typed the entire command.

You must be careful how you use command grouping. For example, the following command sorts the directory entry for PRODUCTS.LST as well as the contents of the file:

```
(dir products && type products) | sort >
products.lst
```

Note If you omit a closing parenthesis, MS OS/2 prompts you to complete the command line. You should finish the command line with the appropriate closing parenthesis and press the ENTER key, or press CONTROL+C to cancel the command so that you can enter it properly.

In the next chapter

In this chapter, you have learned the difference between internal commands and external commands. You have also learned how to use redirection and grouping symbols with MS OS/2 commands. The next chapter describes each MS OS/2 command and illustrates how it is used.

7 MS OS/2 Command Reference

This chapter provides details about all of the Microsoft Operating System/2 commands. In this chapter, you will learn the following:

- The structure of the command page
- What command options are
- The notational conventions used in this chapter
- How to use each MS OS/2 command
- How to use each batch command

Command pages are of the following form:

Sample command page

Sample Command

Purpose:

The "Purpose" section tells what the command is used for.

Syntax:

commandname [*options*]

where:
commandname is the name of an MS OS/2 command. *options* may include *drive*:, *path*, *filename*, *pathname*, *switch*, and/or *argument*.

Comments:

Sideheads help you find information fast

The "Comments" section describes how to use the command. It also explains why the command is useful and explores each of the command's options.

Notes:

The "Notes" section discusses important points related to the command, as shown in the following:

- You can specify a drive and/or path before any command, unless the "Notes" section states otherwise.

See Also:

The "See Also" section lists helpful cross-references.

For more information about command options and notational conventions used in this manual, see the following section called "Command Options."

Examples:

The "Examples" section gives one or more examples that illustrate how to use the command.

Command Options

What are command options?

Command options give MS OS/2 extra information about a command. If you omit options, MS OS/2 either prompts you to supply them or uses a default value. See the individual command descriptions in this chapter for default values.

MS OS/2 commands use the following syntax:

command [*options*]

The *command* placeholder refers to an MS OS/2 command, and the [*options*] placeholder refers to one or more of the following:

Option	Description
drive:	Specifies a disk-drive name. You need only specify a drive name if you are using a file that is *not* on the default drive. Information transferred between two disks is sent from a source drive to a target drive.
path	Specifies a directory name with the following syntax: [*directory*][*directory...*]*directory*
filename	Specifies a file, and includes any filename extension. The *filename* option does not refer to a device or drive name.
pathname	Specifies a directory and a file in the directory. A *pathname* uses the following syntax: [*directory*][*directory...*]*filename*
switch	Controls an MS OS/2 command. A switch begins with a slash; for example, **/p**.

argument	Provides more information to MS OS/2 commands. You usually choose between arguments; for example, **on** or **off**.
string	Many commands work with 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.

Conventions for command options

The *Microsoft Operating System/2 User's Reference* uses the following conventions for command options:

Convention	Use
italics	You must supply the text for any of the variables shown in italics. For example, when *filename* appears, you should type the name of your file.
[brackets]	Items in brackets are optional. To include optional information, type only the information within the brackets. Do not type the brackets themselves.
... (ellipsis)	An ellipsis (...) means that you can repeat an item as many times as necessary.
separators	Unless otherwise specified, you must use spaces to separate commands from their options, as shown in the following: `rename stage.doc location.doc` With some commands you may use a semicolon (;), an equal sign (=), or a tab to separate MS OS/2 commands from their options. These characters are also known as separators. In this manual, spaces separate commands from their options.

Command List

The following list briefly describes each MS OS/2 command and batch command:

Command	Purpose
ansi	Installs ANSI escape sequence support in protected mode.
append	Sets a search path to locate files outside of the current directory that have extensions other than .EXE, .COM, and .BAT.
assign	Assigns a drive letter to a different drive.
attrib	Displays or changes the attributes of selected files in a directory.
backup	Backs up one or more files from one disk to another.
Batch:	
call	Calls one batch file from another.
echo	Turns the batch **echo** feature on or off.
endlocal	Restores the drive, directory, and environment settings in effect before the **setlocal** command was received.
extproc	Defines an external batch processor for a batch file.
for	Performs a command for a set of items.
goto	Processes commands starting with the line after the specified label.
if	Performs a command based on the result of a condition.
pause	Suspends execution of the batch file.
rem	Displays remarks during execution of a batch file.
setlocal	Defines drive, directory, and environment variables for the current batch file.
shift	Changes the position of replaceable parameters in batch-file processing.

break	Sets the CONTROL+C check.
chcp	Displays or changes the current code page for the command processor CMD.EXE or COMMAND.COM.
chdir	Changes your working directory to the directory you specify; when used without parameters, displays the name of your working directory.
chkdsk	Scans the disk in the specified drive and checks it for errors.
cls	Clears the screen.
cmd	Starts a secondary protected-mode command processor.
command	Starts a secondary real-mode command processor.
comp	Compares the contents of two files or sets of files.
copy	Copies one or more files to another location. This command also appends files.
date	Specifies the date to the system.
del	Deletes all files specified by the drive and pathname.
detach	Detaches a special process so that it runs in the background.
dir	Lists the entries in a directory.
diskcomp	Compares the contents of the floppy disk in the source drive to the floppy disk in the target drive.
diskcopy	Copies the contents of the floppy disk in the source drive to a formatted or unformatted floppy disk in the target drive.
dpath	Tells the application which directories it should search to locate data files (extensions other than .EXE, .COM, .BAT, and .CMD) outside of the current directory.
exit	Exits from the command processor (CMD.EXE in protected mode or COMMAND.COM in real mode) and returns to a previous level, if one exists.

fdisk	Configures a hard disk for use with MS OS/2.
find	Searches for a specific string of text in a file or files.
format	Formats the disk in the specified drive to accept MS OS/2 files.
graftabl	Enables an extended character set to be displayed when using display adapters in graphics mode.
helpmsg	Provides help information related to a warning or error message.
join	Joins an existing disk drive to a specific path.
keyb	Specifies a country-specific keyboard layout when one other than a United States keyboard layout is used.
label	Creates or changes the volume label on a disk.
mkdir	Makes a new subdirectory in the working directory.
mode	Sets operation parameters for communication and output devices.
more	Sends output to the console one screen at a time.
patch	Makes changes to program code.
path	Specifies a command search path.
print	Prints a text file on a lineprinter while you are processing other MS OS2 commands (usually called background printing).
prompt	Changes the MS OS/2 command prompt.
recover	Recovers a file or disk containing bad sectors.
rename	Changes the name of a file.
replace	Updates files on your hard disk with new versions of software.

restore	Restores files that were backed up using the **backup** command.
rmdir	Removes a directory from a multilevel directory structure.
set	Equates one string of characters in the environment with another string for later use in programs.
setcom40	Sets or clears the apparent COM port availability to some real-mode application programs.
sort	Reads input, sorts the data, then writes the sorted data to your screen, to a file, or to another device.
spool	Starts the printer spooler for background printing while you are processing other MS OS/2 commands or programs.
start	Begins a new process in a new protected-mode session.
subst	Substitutes a drive letter for a path.
sys	Transfers the MS OS/2 system files from the disk in the default drive to the disk in the specified drive.
time	Specifies the time to the system.
tree	Displays the path (and as an option lists the contents) of each directory and subdirectory on the given drive.
type	Displays the contents of a text file on the screen.
ver	Displays the MS OS/2 version number.
verify	Turns verification on or off when writing to a disk.
vol	Displays the disk volume label if it exists.
xcopy	Copies files and directories, including subdirectories if they exist.

These commands are described in detail in the following pages.

Ansi

The Ansi command

Protected Mode Only

Purpose:

Installs ANSI escape sequence support in protected mode.

Syntax:

ansi [**on**]

or

ansi [**off**]

Comments:

The **ansi** command lets you use the ANSI escape sequences if MS OS/2 is operating in protected mode. An ANSI escape sequence is a series of characters (beginning with an escape character or keystroke) that you can use to define functions in MS OS/2. For more information about ANSI escape sequences, see the *Microsoft Operating System/2 Setup Guide.*

If you set **ansi** to **off**, the ANSI escape sequences are not supported. The default is **on**. If you type **ansi** without an argument, the current setting appears on your screen.

Notes:

- In protected mode, the ANSI escape sequences to set video to graphics modes, **Esc[=4h**, **Esc[=5h**, and **Esc[=6h**, respectively, are not supported.
- To use ANSI escape sequences in real mode, include the following line in your CONFIG.SYS file:

  device=[*drive*:][*path*]**ansi.sys**

Examples:

If you had a program that required ANSI display control, you would want ANSI to be on. But, if you ran a graphics program that did not require ANSI display control, you would type this command before you ran your program:

```
ansi off
```

Afterward, you would type the following to turn ANSI support back on:

```
ansi on
```

The Append command

Append

Real Mode Only

Purpose:

Sets a search path to locate files outside of the current directory that have extensions other than .EXE, .COM, and .BAT.

Syntax:

First use only:

append [**/e**]

To specify directories to be searched:

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

To delete appended paths:

append;

where:
path is the directory MS OS/2 searches for a data file.

Comments:

The **append** command allows you to specify a search path for data files. MS OS/2 first searches the working directory, or the directory specified in a command, for data files. If the data files are not there, MS OS/2 searches the directories of the appended path in the order you specify.

You can specify more than one *path* to search by separating each with a semicolon (;). The entire **append** command line cannot exceed 128 characters.

If you type the **append** command with the *path* option a second time, MS OS/2 discards the old search path and uses the new one.

The Append /e switch

The **append** command accepts the **/e** switch the first time the command is invoked only. This switch is used to store the appended paths in the MS OS/2 environment.

If you type the **append** command by itself, MS OS/2 displays the current data path.

And if you use the following command, MS OS/2 sets the NUL data path:

```
append;
```

This means that MS OS/2 searches only the working directory for data files.

Notes:

- You can use the **append** command across a network to locate remote data files.
- Some applications may read a file from an appended path, but write it to the current directory, leaving the original copy unchanged.
- If you are using the MS OS/2 **assign** command, you must set the data path with the **append** command before giving the **assign** command.

See Also:

For more information about setting a search path for data files in protected mode, see the **dpath** command in this chapter. If you want to set a search path for external commands, see the **path** command in this chapter.

Example:

Setting a search path

The following command-line example tells MS OS/2 to search three directories—\SCENE\MERTZAPT, \CHARACTR\TRUMBEL, and B:\EPISOD92—to find data files (the command line has been broken because of page restrictions):

```
append \scene\mertzapt;\charactr\trumbel;
b:\episod92
```

Assign

Real Mode Only

Purpose:

Assigns a drive letter to a different drive.

Syntax:

assign [*x*[=]*y*[...]]

where:
x is the drive that MS OS/2 currently reads and writes to.
y is the drive that you want MS OS/2 to read and write to.

Comments:

The **assign** command lets you use drives other than those recognized for a given application. For example, if your application only lets you read and write files on drives A and B, you can use the **assign** command to read and write files from other drives. You cannot, however, assign a drive being used by another program, and you cannot assign an undefined drive.

Do not type a colon after the drive letters *x* and *y*.

Notes:

- To ensure compatibility with future versions of MS OS/2, you should use the **subst** command instead of **assign**. The following commands, therefore, are equivalent:

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

- Since the **assign** command disguises the true device type, you should not use **assign** in the following situations:
  - with commands that require drive information (**backup**, **restore**, **label**, **join**, **subst**, **print**)
  - during normal use of MS OS/2

Two other commands, **format** and **diskcopy**, ignore drive reassignments.

Examples:

If you want to run an application on a hard-disk drive, C, and this application requires you to put your program disk into drive A and your data disk into drive B, you would type this command line:

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

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

Setting drive assignments

To reset all drives to their original assignments, type the following command and press the ENTER key:

```
assign
```

The Attrib command

Attrib

Purpose:

Displays or changes the attributes of selected files in a directory.

Syntax:

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

where:
+**r** sets the read-only attribute of a file.
−**r** clears the read-only attribute of a file.
+**a** sets the archive attribute of a file.
−**a** clears the archive attribute of a file.

Comments:

The **attrib** command sets read-only and/or archive attributes for files. You may use wildcards to specify a group of files. The attributes of those files matching *filename* are displayed or modified based on the switch selection. **Attrib** does not accept a directory name as a valid filename.

The *drive* and *pathname* specify the location of the file or files you want to use. The /**s** switch processes all sub-directories as well as the path specified.

The **backup**, **restore**, and **xcopy** commands use the archive attribute as a control mechanism. You can use the +**a** and −**a** options to select files that you want to back up with the **backup /m** command line, or copy with the **xcopy /m** or **xcopy /a** command lines.

Notes:

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

Examples:

Displaying file attributes

To display the attribute of a file called NEWS86 on the default drive, you would type the following command line:

```
attrib news86
```

Setting read-only permissions

The following command line gives the file REPORT.TXT read-only permission:

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

Setting a file as read-only prevents you from accidentally deleting or modifying it.

To remove read-only permission from the files in the \SCHEDULE\NEWSHOWS directory on drive B, and the files in any of its subdirectories, you would type the following command line:

```
attrib -r b:\schedule\newshows /s
```

Using the archive switch

Suppose you want to give a coworker a disk that contains all files in the default directory of the disk in drive A, except for files with the extension .BAK. To copy these files to a disk in drive B, you would type the following two command lines, pressing the ENTER key after each one:

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

You then would type one of the following two command lines:

```
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.

The Backup command

Backup

Purpose:

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

Syntax:

backup [*drive1*:][*path*][*filename*] [*drive2*:] [**/s**][**/m**][**/a**] [**/f**] [**/d:***date*] [**/t:***time*] [**/L:***filename*]

where:
drive1 is the disk drive that you want to back up.
drive2 is the target drive to which the files are backed up.

Comments:

The **backup** command can back up files on disks of different media (hard disks and floppy disks). **Backup** also backs up files from one floppy disk to another, even if the disks have a different number of sides or sectors.

Backup switches

The **backup** command accepts these switches:

Switch	Purpose
/s	Backs up subdirectories.
/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. This switch will not be accepted if files exist that were backed up using **backup** from MS-DOS version 3.2 or earlier.
/f	Specifies that an unformatted target disk will be formatted. This switch does not allow formatting of nonremovable media.
/d:*date*	Backs up only those files that you last modified on or after *date*.
/t:*time*	Backs up only those files that you last modified at or after *time*.

/L:*filename* Makes a backup-log entry in the specified file. If you do not specify *filename*, **backup** places a file called BACKUP.LOG in the root directory of the disk that contains the files being backed up.

Backup-log files

A backup-log file uses the following format:

- The first line lists the date and time of the backup.
- A line for each backed-up file lists the filename and number of the backup disk on which the file resides.

If the backup-log file already exists, **backup** appends the current entry to the file.

You can use the backup-log file when you need to restore a particular file from a floppy disk. You must specify which disk to restore so that the **restore** command does not have to search for files. The **restore** command always puts a file back in its original place. **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, MS OS/2 lets you back up only those files to which you have access.

Notes:

- Unless you use the **/a** option, **backup** erases the old files on a backup disk before adding new files to it.
- 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** commands. If you do, you may not be able to restore the files with the **restore** command.

Example:

Backing up files

Suppose you want to back up all of the files in the \FILM\CRITIQUE directory on drive C to a blank, formatted disk in drive A. To do this, you would type the following:

```
backup c:\film\critique a:
```

Batch Command: Call

Purpose:

Calls one batch file from another.

Syntax:

call [*drive*:][*path*] *batchfile* [*argument*]

where:
batchfile is the batchfile you want to call.
argument is the command in this batch file that will be run following *batchfile*.

Comments:

The **call** command is used within one batch file to call another batch file. *Batchfile* must have a filename extension of .BAT in real mode, or .CMD in protected mode.

When *batchfile* terminates, the calling batch file resumes running at *argument*. If *argument* is omitted, the calling batch file resumes running at the command immediately following the **call** command.

Notes:

- Do not use pipes and redirection symbols with the **call** command.
- A batch file can make a recursive call to itself, but there should be a termination condition that is eventually met.

Example:

To run the CHECKNEW.CMD file from another protected-mode batch file, you would use the following command line within the first batch file:

```
call checknew
```

Batch Command: Echo

Batch command (Echo)

Purpose:

Turns the batch echo feature on or off.

Syntax:

echo [**on**]

or

echo [**off**]

or

echo [*message*]

where:
message is a line of text.

Comments:

Normally, commands in a batch file are displayed ("echoed") on the screen when MS OS/2 receives them. 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**.

Sending a message

The **echo** *message* command line is useful only if **echo** is set to **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 display 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.

If you type **echo** without specifying any option, MS OS/2 displays the current setting.

Notes:

An at symbol (@) placed in front of a command line in a batch file prevents that line from echoing.

Examples:

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.
```

If you want to turn **echo** off, and do not want the command itself to be echoed, include the @ symbol before the command line:

```
@echo off
```

Batch Command: Endlocal

Purpose:

Restores the drive, directory, and environment settings in effect before the **setlocal** command was received.

Syntax:

endlocal

Comments:

The **endlocal** command terminates the **setlocal** command, restoring the previous drive, directory, and environment settings.

See Also:

The section on the batch command **setlocal** in this chapter.

For more information about the MS OS/2 environment, see the section on the **set** command in this chapter.

Examples:

Suppose you used the **setlocal** command to set an alternate search path in your batch file. At the end of the batch file you should reset the original path used by your MS OS/2 environment. To do this, you would include the following line in your batch file:

```
endlocal
```

Batch Command: Extproc

Purpose:

Defines an external batch processor for a batch file.

Syntax:

extproc [*batch-processor-name*] [*argument*]

where:
argument is any valid option for *batch-processor-name*.

Comments:

You can use the **extproc** command only as the first line of a batch file to define an alternate command processor. This command processor is used to process the batch file.

Examples:

Suppose you wanted to create a batch file to run under a command processor called BORNE.EXE instead of CMD.EXE. You would include this line as the first line of your batch file:

```
extproc borne.exe
```

Batch Command: For

Batch command (For)

Purpose:

Performs a command for a set of conditions.

Syntax:

for *%%c* **in** (*item*[...]) **do** *command*

where:
c is any character except 0,1,2,3,...,9 (to avoid confusion with the *%0—%9* batch parameters).
item is a filename or any portion of a command line.
command is the desired command.

Comments:

The **for** command sequentially sets the *%%c* variable to each member of *item*, and uses the variable to evaluate *command*. If *item* is an expression involving a wildcard (* or ?), then the variable *%%c* represents each matching *item* from the disk.

Notes:

- In batch files, the percent sign (%) is used to flag variable batch parameters. Except for the batch parameters *%0—%9*, you must use two percent signs (%%) before the name of a variable batch parameter. Otherwise, the variable batch parameter name will not be recognized as valid. Thus, if a batch file included the variable *%f*, instead of *%%f*, the batch parameter processor would consider the variable batch parameter name *f* invalid, and the **for** command containing that parameter name would ignore the parameter.

  If the **for** command is *not* in a batch file, you use only one percent sign.
- For interactive processing from the MS OS/2 prompt, type the **for** command in this format:

  for *%c* **in** (*item*[...]) **do** *command*

Examples:

Assigning a variable to a set of files

The following example replaces the variable *%%f* with 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
```

Batch Command: Goto

Purpose:

Processes commands starting with the line after the specified label.

Syntax:

goto *label*

where:
label specifies another location in the batch file.

Comments:

The **goto** command directs MS OS/2 to read a command from a line specified by *label. Label* may include spaces, but not other separators, such as semicolons or equal signs. A colon (:) must precede each label used in a batch file. If your batch file does not contain *label,* the batch file is terminated.

Notes:

Any line in a batch file that starts with a colon (:) is ignored during batch processing, because MS OS/2 assumes the line is a label.

Example:

The following example sends the program processor to the label **end** only if no errors occurred when you were formatting 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.
```

Batch command (If)

Batch Command: 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:

Using conditional statements

The **if** statement allows conditional execution of commands. When the condition is true, MS OS/2 executes *command*; otherwise it ignores *command.*

The following list describes the possible conditions:

Condition	Description
errorlevel *number*	True only if the previous program executed by CMD.EXE had an exit code equal to, or greater than, *number.* (When a program finishes, it returns an exit code via MS OS/2.) You can use this condition to perform other tasks that are based on the previous program's exit code.
*string*1 = = *string*2	True 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 only if *filename* exists.

If you specify the **not** parameter, MS OS/2 executes *command* when the condition is false.

Notes:

For more information about exit codes related to MS OS/2 commands, see the appendix, "MS OS/2 Command Exit Codes."

Examples:

The following example prints the message, "Can't find data file," if the file PRODUCT.DAT does not exist on the disk:

```
if not exist product.dat echo can't find
data file
```

The following example sends the program processor to the label **end** only if no errors occurred when you were formatting 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.
```

Batch Command: Pause

Purpose:

Suspends execution of the batch file.

Syntax:

pause [*comment*]

where:
comment is any special message.

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. If you press CONTROL+C, the batch process is terminated.

When the command processor encounters **pause**, it prints *comment* and then prints the following message:

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

Unless **echo** is off, **pause** displays any *comment* before the "Strike a key" message.

If you press CONTROL+C , MS OS/2 does the following:

- When running in real mode, **pause** displays a prompt:

  ```
  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.
- When running in protected mode, MS OS/2 terminates the batch process, and returns control 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.

Notes:

The **pause** *comment* command 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. If **echo** is off, you can use the **echo** command to display a message before **pause** displays the "Strike a key" message.

Batch command (Rem)

Batch Command: Rem

Purpose:

Displays remarks during execution of a batch file.

Syntax:

rem [*comment*]

where:
comment is a line of text to document the batch file's function.

Comments:

When a batch file encounters a **rem** command, the *comment* associated with it appears. *Comment* is commonly used to describe the batch file.

The only separators allowed in *comment* are spaces, tabs, and commas.

If *comment* is omitted, the **rem** command appears on a line by itself, adding spacing for readability.

Notes:

If **echo** is **off**, the **rem** comment is not displayed.

Example:

Using remarks in a batch file

The following batch file 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:
```

Batch Command: Setlocal

Purpose

Defines local drive, directory, and environment variables for the current batch file.

Syntax:

setlocal

Comments

The **setlocal** command saves the current drive, directory and environment variables, and lets you define the local variables for a batch file. The original drive, directory and environment settings are restored when MS OS/2 receives the **endlocal** command, or when the batch file terminates.

You can include multiple **setlocal** commands without including matching **endlocal** commands. If MS OS/2 doesn't find a matching **endlocal** command, it restores all saved elements when the batch file issuing the **setlocal** command terminates. This way, any environment variable and the current drive and directory can be altered without affecting the CMD.EXE command processor.

See Also:

The section on the batch command **endlocal**.

Example:

Suppose for your batch job you wanted to read files from a directory called TESTACCT. To use this new search path instead of the current search path set up by the **path** command, you would include the following lines in your batch file:

```
setlocal
path \testacct
```

You would include the **endlocal** command to return to the original search path.

Batch command (Shift)

Batch Command: Shift

Purpose:

Changes the position of replaceable parameters in batch-file processing.

Syntax:

shift

Comments:

You can use the **shift** command to change the positions of command-line parameters.

Usually, command files are limited to handling ten parameters, *%0* through *%9*. The parameter *%0* is always replaced with the drive name (if you specified it) and the filename of your batch file. You can specify nine replaceable parameters, *%1* through *%9*.

You use the **shift** command to 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 fewer than ten parameters.

Notes:

There is no backward **shift** command. Once you have executed **shift**, you cannot recover the first parameter (*%0*) that existed before the shift.

Examples:

Shifting replaceable parameters

For example, suppose parameter *%3* through *%9* are empty and the other parameters are as follows:

%0 = drive C
%1 = station
%2 = network

Including the **shift** command would result in the following:

%0 = station
%1 = network

Parameters *%2* through *%9* are empty.

Using replaceable parameters

The following batch file, called COPYSET.BAT, uses the **shift** command to copy a list of files to a specific directory:

```
rem copyset.bat copies
rem any number of files
rem to a directory.
rem The command is
rem copyset dir files
:one
if "%1" = " " goto two
set todir = %1
shift
copy %1 %todir%
goto one
:two
set todir=
echo All done
```

The Break command

Break

Real Mode Only

Purpose:

Sets the CONTROL+C check.

Syntax:

break [**on**]

or

break [**off**]

Comments:

Depending on the program you are running, you may press CONTROL+C (or CONTROL+BREAK) to stop an activity (for example, to stop sorting a file). Normally, MS OS/2 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.

Notes:

Some programs set themselves to respond to CONTROL+C at any time. The **break** command does not affect these programs.

Examples:

Checking for CONTROL+C

To have MS OS/2 check for CONTROL+C only during screen, keyboard, and printer reads and writes, type the following:

```
break off
```

To find out whether **break** is currently on or off, type **break** and press the ENTER key.

Chcp

Purpose:

Displays or changes the current code page for the command processor CMD.EXE or COMMAND.COM.

Syntax:

chcp [*nnn*]

where:
nnn is the code page to start.

Comments:

The **chcp** command accepts one of the two prepared system code pages as a valid code page. An error message is displayed if a code page is selected that has not been prepared for the system.

If you type **chcp** without a code page, it displays the active code page and the prepared code pages of the command processor in that session.

You may select one of the prepared system code pages defined by the **codepage** command in the CONFIG.SYS file.

The following are valid code pages:

Value	Code Page
437	United States
850	Multilingual
860	Portuguese
863	French-Canadian
865	Nordic

Any program that you run after starting a new code page will use the new code page. Programs that started before the new code page will still use the original code page.

To see what the current code-page setting is, type the following:

```
chcp
```

The Chcp command

Checking the code page

MS OS/2 will respond with a message similar to the following:

```
Active code page: 850
Prepared system code pages: 850 437
```

If a code page is selected that is not prepared for the system, MS OS/2 displays a message like the following:

```
Code page 850 not prepared for system
Active code page: 437
Prepared system code pages: 437 865
```

If a device (the screen, keyboard, or printer) is not prepared for a code page, the following error message is displayed:

```
Code page 850 not prepared for device xxx
```

Notes:

- This command sets the code page for an individual session.
- If you use the **chcp** command to select a code page that has not been prepared for a device, but is prepared for the system, the system code page will be changed. If you want to change the active code page to the original code page, then use the **chcp** command with the original code page selected.

See Also:

For more information about using code pages and related commands, see the *Microsoft Operating System/2 Setup Guide.*

Example:

If you want to set the code page for the current session to 863 (French-Canadian), you would type the following command:

```
chcp 863
```

Chdir (cd)

Purpose:

Changes your working directory to the directory you specify; when used without parameters, displays the name of your working directory.

Syntax:

chdir [*drive:*][*path*]

Comments:

A shorthand notation for the **chdir** command is **cd**. Thus, either of the following commands will change your current directory to the directory called PRIMETIM:

```
chdir \primetim
cd \primetim
```

There are two shortcuts you can use when you want to change your directory to a parent directory or subdirectory of your working directory. To illustrate, suppose you have a directory called SPECIALS that has a subdirectory called SPONSORS. Usually, to change your working directory to \SPECIALS\SPONSORS, you would type the following:

```
cd \specials\sponsors
```

However, if your working directory is SPECIALS, you can type the following command line to change to the \SPECIALS\SPONSORS directory:

```
cd sponsors
```

If you want to change your working directory back to the parent directory, \SPECIALS, you could type this command line:

```
cd ..
```

You can type **cd ** to return to the root directory. The root directory is the highest-level directory on your disk and is usually the directory that you're in when you start MS OS/2.

The Chdir (cd) command

Shortcuts

Examples:

If your working directory is \RERUNS\SITCOM and you want to change your path to another directory (such as \RERUNS\DETECTIV), type the following command line and press the ENTER key:

```
cd \reruns\detectiv
```

Displaying your working-directory name

If you use the **cd** command without a path, you can display the name of your working directory. For example, if your working directory is \RERUNS\DETECTIV on drive C, and you type the **cd** command, then press the ENTER key, MS OS/2 displays the following:

```
c:\reruns\detectiv
```

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

```
cd a:
```

Chkdsk

The Chkdsk command

Purpose:

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

Syntax:

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

Comments:

The **chkdsk** command shows the status of your disk. You should run **chkdsk** occasionally on each disk to check for errors. **Chkdsk** will display any error messages, followed by a status report.

Chkdsk status report

A typical status report might look like this:

```
31768576    bytes total disk space
   45056    bytes in 2 hidden files
   34816    bytes in 15 directories
 6895616    bytes in 462 user files
24793088    bytes available on disk
```

If **chkdsk** is running in real mode, a real-mode memory report like the following also appears:

```
524288 bytes total storage
415136 bytes free
```

If you type a filename after **chkdsk**, MS OS/2 displays a status report for the disk and for the individual file.

Chkdsk switches

The **chkdsk** command accepts the following switches:

Switch	Purpose
/f	Fixes errors on the disk. If you do not specify this switch, **chkdsk** does not correct errors that it finds in your directory. However, it does display messages about files that need to be fixed.
/v	Displays the name of each file in each directory as it checks the disk.

If you specify the **/f** switch, **chkdsk** will show an error if there are any open files on the disk. If you do not specify **/f** and there are open files, **chkdsk** may report that there are lost clusters on the disk. This happens when the File Allocation Table has yet to be updated regarding open files. If a large number of clusters are reported as lost, you should consider repairing the disk.

You should not try to fix errors while you are using the multitasking features of MS OS/2, such as background printing or spooling.

Notes:

- The number of free bytes reported may vary greatly if more than one task is being performed, or if you have used commands or programs that remain resident in memory. You may be able to free some memory by restarting MS OS/2.
- **Chkdsk** doesn't work on drives used in the **subst** or **join** command.

Examples:

To check the status of your disk type the following:

```
chkdsk
```

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. In this example, the status report is sent to a file called STATUS:

```
chkdsk a: >status
```

Don't 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 finds on the disk in drive A, prompting you for further information as needed.

Cls

The Cls command

Purpose:

Clears the screen.

Syntax:

cls

Comments:

The **cls** command clears your terminal screen, leaving only the MS OS/2 prompt and a cursor.

Example:

Clearing your screen

If you want to start a new process with a clear screen, type the following:

```
cls
```

The Cmd command

Cmd

Protected Mode Only

Purpose:

Starts a secondary protected-mode command processor.

Syntax:

cmd [*drive*:][*path*] [**/c** *string*]

or

cmd [*drive*:][*path*] [**/k** *string*]

where:
string is a command that you want to pass to the command processor.

Comments:

The **cmd** command starts a new command processor in protected mode.

Cmd switches

The **cmd** command accepts these switches:

Switch	Purpose
/c *string*	Tells the command processor to perform the command or commands specified by *string* and then return automatically to the primary command processor.
/k *string*	Tells the command processor to perform the command or commands specified by the *string* and to keep the new command processor in memory after the command is completed.

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 environment.

The command processor is loaded into memory in two parts: transient and resident. Transient information is loaded into memory only temporarily. Resident information stays in memory and may not be overwritten. Some applications write over the transient memory part of CMD.EXE when they run. When this happens, the resident part of the command processor looks for the CMD.EXE file on disk so it can reload the transient part.

See Also:

For more information about using an alternate protected-mode command processor, see the *Microsoft Operating System/2 Setup Guide.*

Examples:

Starting a protected-mode process

The following command tells the MS OS/2 command processor to do three things:

- Start a new command processor under the current program
- Run the **chkdsk b:** command
- Return to the primary command processor when the command is completed

```
cmd /c chkdsk b:
```

If you want to keep the secondary command processor in memory after MS OS/2 performs the **chkdsk** command, you would type this command:

```
cmd /k chkdsk b:
```

When you no longer need the secondary command processor, type **exit** to remove that session from memory.

The Command command

Command

Real Mode Only

Purpose:

Starts a secondary real-mode command processor.

Syntax:

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

Comments:

The **command** command starts a new subprocess using the COMMAND.COM command processor in real mode.

Command switches

The **command** command accepts the following switches:

Switch	Purpose
/p	Keeps the secondary command processor in memory and does not automatically return to the primary command processor.
/c *string*	Tells the command processor to perform the command or commands specified by *string* and then to return automatically to the primary command processor.
/e:*nnnnn*	Specifies the environment size, where *nnnnn* is the size in bytes ranging from 160 to 32768. This number is rounded up to the next logical paragraph boundary by MS OS/2. The default value is 160 bytes.

If you try to set an environment size of less than 160 bytes, MS OS/2 uses the default value of 160 bytes and displays the following message:

```
Invalid environment size specified
```

If you try to set an environment size greater than 32,768 bytes, MS OS/2 displays the same message, but uses a default value of 32,768 bytes.

When you start the real-mode command processor you create a new command environment. You can change the new environment without affecting the old environment.

The command processor is loaded into memory in two parts: transient and resident. Transient information is loaded into memory only temporarily. Resident information stays in memory and may not be overwritten. 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 it can reload the transient part.

Notes:

You cannot increase the size of the environment *after* you have run an MS OS/2 command that increases the resident size of MS OS/2. You must set the environment size first.

Example:

The following command tells the MS OS/2 command processor to do three things:

- Start a new command processor under the current program
- Run the **chkdsk b:** command
- Return to the primary command processor when the command is completed

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

The Comp command

Comp

Purpose:

Compares the contents of two files or sets of files.

Syntax:

comp [*drive*:][*pathname*] [*drive*:][*pathname*]

Comments:

The **comp** command compares one file or set of files (first *pathname*) with a second file or set of files (second *pathname*).

These files can be on the same drive or on different drives. They can also be in the same directory or different directories.

The two sets of files you want to compare can have the same path and filenames—provided they are on different drives. If you only type a drive for the second option, **comp** assumes that the second *pathname* is the same as the first *pathname*. You can use wildcards (* and ?) to specify the *pathname*. For more information about using wildcards, see the *Microsoft Operating System/2 Beginning User's Guide*.

If you don't type the *pathname* options or if you omit the second *pathname* option, **comp** prompts you for them. If either option contains only a drive or a path with no filename, **comp** assumes the filename is *.* (all files).

Comparing files on different disks

If the files you want to compare are not on the disk that contains the **comp** command, type the **comp** command with no options. When **comp** prompts you for the *pathname* options, you can insert the correct disk and type the filenames to be compared.

As **comp** proceeds, it displays the paths and names of the compared files. A message appears if **comp** cannot find a file matching the second *pathname* option, or if a directory path is invalid. If no file matches the first *pathname* option, **comp** prompts you for both the first and second *pathname* options again.

During the comparison, a message appears for any location in the two files that contains mismatching information. The message indicates the offset into the files of the mismatching bytes and the contents of the bytes themselves (all in hexadecimal notation). The message has the following format:

```
Compare error at OFFSET XXXXXXXX
file1 = XX
file2 = XX
```

In this format, FILE1 is the first filename typed in the command; FILE2 is the second filename typed. After ten unequal comparisons, **comp** stops comparing and displays the following message:

```
10 Mismatches - ending compare
```

If the file sizes are different, **comp** displays the following message:

```
Files are different sizes, do you wish
to continue (Y/N)?
```

If you choose to continue (type **y**), **comp** compares the files until it reaches the end of the shorter file.

After a successful comparison, **comp** displays the following message:

```
Files compare OK
```

After the comparison of the two files ends, **comp** proceeds with the next pair of files that match the two *pathname* options, until no more files can be found that match the first *pathname* option. Then **comp** displays the following message:

```
Compare more files (Y/N)?
```

You now can compare two more files or end the comparison. If you want to compare two more files, type **y**. **Comp** prompts you for two new path options.

For all file comparisons, **comp** first ensures that both files include end-of-file (CONTROL+Z) marks. If not, **comp** displays this message and the files are not compared:

```
EOF mark not found
```

Example:

In the following example, **comp** compares each file with the extension .ASM in the current directory on drive C with each file of the same name (but with an extension of .BAK) in the current directory on drive B:

```
comp c:*.asm b:*.bak
```

The Copy command

Copy

Purpose:

Copies one or more files to another location; also appends files.

Syntax:

To copy files:

copy [*drive*:]*pathname*[*drive*:][*pathname*] [**/v**] [**/a**] [**/b**]

or

copy [*drive*:]*pathname* [**/v**] [**/a**] [**/b**] [*drive*:][*pathname*]

To append files:

copy *pathname* + *pathname* [...] *pathname*

Comments:

If you do not specify the second *pathname*, the copy is created in the working directory on the disk in the default drive. This copy has the same name, creation date, and creation time 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 OS/2 displays the following error message:

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

Copy switches

The **copy** command accepts these switches:

Switch	Purpose
/v	Verifies that the sectors written on the target disk are recorded properly.
/a	Copies ASCII (text) files. This switch applies to the filename preceding it, and to all remaining filenames in the command, until **copy** encounters another **/a** or **/b** switch. This switch tells the command processor to read until the end-of-file mark.

/b	Copies binary files. This switch applies to the filename preceding it, and to all remaining filenames in the command, until **copy** encounters another **/a** or **/b** switch. This switch tells the command processor to read the number of bytes specified by the file size in the directory.

If MS OS/2 cannot verify that the file has been copied correctly, it displays an error message. Although recording errors rarely occur when you copy files, the **/v** switch lets you verify that critical data has been correctly recorded. However,it also causes the **copy** command to run more slowly because MS OS/2 must check each entry recorded on the disk.

Notes:

Using switches /a and /b

- The **copy** command switches **/a** and **/b** perform differently depending on whether you place them after the source filename or after the target filename.

  When used after the source filename, the following occurs:

Switch	Result
/a	Treats the file as an ASCII (text) file. Data in the file is copied up to but not including the first end-of-file mark (CONTROL+Z). The remainder of the file is not copied. The following command line shows the placement of the **/a** switch: `copy memo.doc /a letter.doc`
/b	Copies the entire file, including any end-of-file marks. The following command line shows the placement of the **/b** switch: `copy billing.asm /b billing2.asm`

When used after the target filename, the following occurs:

Switch	Result
/a	Adds an end-of-file character as the last character of the file.
/b	Does not add an end-of-file character.

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

- Do not try to append files if one of the source filenames has the same name or extension as the target.

See Also:

Copying files and subdirectories

If you want to copy all of a directory's files and subdirectories, you should use the **xcopy** command. See the section on the **xcopy** command in this chapter for more information.

Examples:

To copy a file called ANIMAL.SHO from your working drive and directory to a directory on drive C called FALLSHOW, type the following:

```
copy animal.sho \fallshow
```

Appending files

The **copy** command also lets you append files. To do this, simply list any number of files as options to **copy**, each separated by a plus sign (+), and then specify a target file to send the combined files to. The following command combines files named INTRO.RPT, BODY.RPT, and SUM.RPT (from drive B), and places them in a file called REPORT on the default drive:

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

When you are appending files, the target file is created with the current date and time. If you omit the target file, MS OS/2 combines the files, and stores them under the name of the first specified file.

Combining files

You can also combine several files into one by using wildcards. The following command takes all files with an extension of .TXT and combines them into one file named COMBIN.DOC:

```
copy *.txt 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, VIDEO.TXT is combined with VIDEO.REF to form VIDEO.DOC, AUDIO.TXT with AUDIO.REF to form AUDIO.DOC, and so on:

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

The following **copy** command line combines all files with a .TXT extension and all files with a .REF extension into one file named COMBIN.DOC:

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

The Date command

Date

Purpose:

Specifies the date to the system.

Syntax:

date [*mm-dd-yy*]

Comments:

Changing the date

You can change the date from your terminal or from a batch file. (MS OS/2 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 OS/2 records the date in the directory listing when you create or change a file.

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

mm = 1–12
dd = 1–31
yy = 80–79 or 1980–2079

The date, month, and year entries can be separated by hyphens (-), slashes (/), or periods (.). MS OS/2 changes months and years correctly, whether the month has 28, 29, 30, or 31 days.

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 lets you change the date format to the European standard *dd-mm-yy*.

Notes:

This command sets the internal calendar in your computer; you do not need to set the date each time you start your computer.

The format *mm-dd-yy* may vary if you are using a code page other than the one for United States.

See Also:

For more information on the CONFIG.SYS file, see the *Microsoft Operating System/2 Setup Guide.*

Examples:

Displaying the current date

If you simply type the **date** command, MS OS/2 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 ENTER key.

If you want to change the system date without first viewing the current date, you can type a particular date after the **date** command, as in the following example:

```
date 3-9-88
```

In this case, the "Enter new date" prompt does not appear after you press the ENTER key.

The Del (Erase) command

Del (Erase)

Purpose:

Deletes all files specified by the drive and pathname.

Syntax:

Real Mode:

del[*drive*:]*pathname*

or

erase[*drive*:]*pathname*

Protected Mode:

del [*drive*:]*pathname* [...]

or

erase [*drive*:]*pathname* [...]

Note: In real mode, there is no space before the *drive*: option. In protected mode, a space before the *drive*: option is required.

Comments:

Using wildcards with Del

The **del** command lets you use wildcards (* and ?) to delete more than one file at a time. While convenient, this method of deletion should be used cautiously so that you don't delete files unintentionally. For more information about wildcards, see the *Microsoft Operating System/2 Beginning User's Guide.*

If you type **del *.***, this tells MS OS/2 that you want to delete all of the files in the working directory. MS OS/2 displays the prompt "Are you sure?" If you type **y** in response, all files in the working directory are deleted.

To delete all the files in another directory, type the **del** command followed by the directory name.

In protected mode, you can specify multiple filenames with **del**.

Warning Once you have deleted a file from your disk, you cannot recover it.

Examples:

Deleting a file

The following command line deletes a file named VACATION:

```
del vacation
```

If you have files named VACATION.FEB and VACATION.APR, you can delete them both with the following command line:

```
del vacation.*
```

In protected mode, you can specify multiple files with **del**. For example, the following command line deletes all the files with a .TXT extension in the default directory of drive B and the file MEMO on drive A:

```
del b:*.txt a:memo
```

To delete all the files in a subdirectory named SALES, you can use either of the following command lines:

```
del sales
```

```
del \sales\*.*
```

The Detach command

Detach

Protected Mode Only

Purpose:

Detaches a special process to run in the background.

Syntax:

detach *command* [*arguments*]

where:
command is any protected-mode MS OS/2 program or command that does not require you to type input from the keyboard.
arguments are any valid arguments for *command.*

Comments:

Running a process in the background

The **detach** command lets you perform special commands or programs in the background. This means that MS OS/2 starts an independent process for that command or program, displays the process identification (PID) number of the detached process, and immediately displays the MS OS/2 prompt. Then you can execute another command while MS OS/2 continues to run your detached command in the background.

If you detach a subprocess and later delete the parent process, the detached process continues to run.

Notes:

- If you try to detach a command or program that shouldn't be run in the background, you could ruin files or lose valuable information.
- Detached programs must come to an end on their own. There is no way for you to end a detached program.

Example:

Suppose you want an alphabetical listing of your directory. The following command would create a sorted file, SORT.DIR, in the background, enabling you to run another process at the same time:

```
detach dir | sort > sort.dir
```

MS OS/2 assigns a process identification number to each process you run. A message of the following form appears:

```
The process identification number is 48.
```

The Dir command

Dir

Purpose:

Lists the entries in a directory.

Syntax:

Real Mode:

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

Protected Mode:

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

Comments:

The **dir** command, typed by itself, lists all directory entries in the working directory on the default drive. If you include a drive letter with the **dir** command, all entries in the default directory of the disk in the specified drive are listed.

Dir switches

The **dir** command accepts the following switches:

Switch	Purpose
/p	Selects page mode, causing the directory display to pause once the screen is filled. To resume scrolling the display, press any key.
/w	Selects wide display and causes MS OS/2 to display only filenames and not other file information. The wide display lists up to five files per line.

In protected mode, when you type more than one filename as a parameter, **dir** displays information according to the following criteria:

- For the first filename on the list and any subsequent filename on a different drive, **dir** prints a message that shows the volume label for the disk.
- For the first filename in the list or any subsequent filename in a different directory, **dir** prints a directory message.
- For files on each drive and directory, **dir** prints file information (size in bytes, time and date of last modification).

- For each drive in the list, **dir** prints information on the number of files and the amount of free space on the drive.

Dir lists all files with their size in bytes and the time and date of last modification.

You can use the ? and * wildcard symbols in the *pathname* option of the dir command. The following table shows some cases in which the * wildcard is optional:

This command	Does the same as this
dir	**dir** ***.***
dir *filename*	**dir** *filename*.*
dir *.ext*	**dir** ***.***ext*

Notes:

If the **country** command in the CONFIG.SYS file is set to a country other than the United States, the directory date and time formats may differ. For more information on the CONFIG.SYS file, see the *Microsoft Operating System/2 Setup Guide.*

Examples:

Displaying a long directory

If you want to view the directory listing for your current directory, but the directory contains more entries than you can see on the screen at one time, type the following:

```
dir /p
```

This command line displays the directory one screen at a time. As one screen fills up, you can press any key to see the next screen of the directory listing.

The Diskcomp command

Diskcomp

Purpose:

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

Syntax:

diskcomp [*drive1*:] [*drive2*:]

where:
*drive*1 is the source drive.
*drive*2 is the target drive.

Comments:

Comparing disks track-by-track

The **diskcomp** command performs a track-by-track comparison of the disks in the source and target drives. It automatically determines the number of sides and sectors per track based on the format of the source disk. Both the source and target disks must be the same type (for instance, high density 5-1/4-inch disks).

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

```
Compare OK
```

If the tracks are not the same, **diskcomp** displays a "Compare error" message that includes the track number and side (0 or 1) where it found the mismatch.

If the target disk and source disk are of different types, **diskcomp** displays the following message:

```
Drive types (double, single sided) or
diskette types not compatible
```

When **diskcomp** completes the comparison, it displays the following message:

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

If you type **y**, **diskcomp** prompts you to insert the proper disks and then performs the next comparison. If you type **n**, **diskcomp** ends. If the disk in the default drive does not contain MS OS/2 system files and you end **diskcomp**, you will receive the following message:

```
Insert disk with CMD.EXE in drive A
and strike any key when ready
```

Comparing disks with one 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. If you do not specify a drive, **diskcomp** prompts you for one.

Notes:

- When comparing floppy disks created by this version of MS OS/2 or a later version, **diskcomp** skips the four-byte volume serial number during its comparison.
- 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 **comp** command to compare individual files on the disk.
- **Diskcomp** does not work on network drives, and you cannot use it with assigned, joined, or substituted drives. If you try to use the **diskcomp** command with these types of drives, an error message will appear.

See Also:

For more information on comparing files, see the section on the **comp** command in this chapter.

Example:

Comparing two disks

If your computer has one floppy-disk drive, drive A, and you want to compare two floppy disks, type the following command:

```
diskcomp a:
```

MS OS/2 prompts you to insert each disk in turn, as required, to compare the contents of the two disks.

The Diskcopy command

Diskcopy

Purpose:

Copies the contents of the floppy disk in the source drive to a formatted or unformatted floppy disk in the target drive.

Syntax:

diskcopy [*drive1:*] [*drive2:*]

where:
*drive*1 is the source drive.
*drive*2 is the target drive.

Comments:

Drive1 and *drive*2 may be the same. If you omit the *drive* options, MS OS/2 prompts you for the drives. If the target disk is not formatted, **diskcopy** formats it with the same number of sides and sectors per track as the source disk. This is because **diskcopy** assumes that your target and source disks are the same type (for example, 1.2 megabyte disks). So, for example, you cannot use the **diskcopy** command to copy from a high-density disk to a low-density disk. If the target disk and the source disk are of different types, **diskcopy** displays the following message:

```
Drive types (double-sided, high capacity)
or diskette types not compatible.
```

Warning **Diskcopy** destroys the existing 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.

If it finds errors on either disk, **diskcopy** indicates the drive, track, and side in error, and proceeds with the copy.

If the source disk was created by this version of MS OS/2, **diskcopy** generates a new (four-byte) volume serial number on the target disk and displays the following message:

```
New Volume Serial Number on target drive
is NNNN-NNNN
```

After copying, **diskcopy** displays the following message:

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

If you type **y**, MS OS/2 prompts you to insert the source and target disks, and performs the next copy on the drives that you originally specified.

To end the **diskcopy** program, type **n**.

Using one drive to copy disks

If you specify the same drive as the source and the target, **diskcopy** uses just the one drive to perform the copy. **Diskcopy** will prompt you to insert disks as necessary.

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

A fragmented disk can delay the finding, reading, or writing of a file. To prevent further fragmentation, you should use either the **copy** or **xcopy** command to copy your disk. These two commands copy files sequentially to a disk, thus avoiding fragmentation.

Notes:

Diskcopy doesn't work on drives used in the **subst** or **join** commands. **Diskcopy** ignores any assignments created by the **assign** command. The source and target disks cannot be a virtual or assigned disk.

See Also:

For more information on copying files, see the sections on the **copy** and **xcopy** commands in this chapter.

Example:

Copying a disk

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

```
diskcopy a: b:
```

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

The Dpath command

Dpath

Protected Mode Only

Purpose:

Specifies which directories the application should search to locate data files (files with extensions other than .EXE, .COM, .BAT, and .CMD) outside of the current directory.

Syntax:

To specify directories to be searched:

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

To clear all settings:

dpath;

To display the list of data path directories:

dpath

Comments:

The **dpath** command lets you tell the application which directories it should search to locate data files not found in the current or specified directory. Each time you use the **dpath** command, it takes the place of the preceding **dpath** utility.

Notes:

- The function of the real-mode **append** command is similar to the function of the protected-mode **dpath** command.
- **Dpath** defines the environment used by an application, much like the **path** command does. That is, both of these commands define where an application may search for files. The **dpath** command defines paths for data files, while the **path** command defines paths for external commands.
- The **dpath** command is assigned for a single application program's session. If you start a new command processor from within a session where **dpath** is defined, the new session inherits the **dpath** settings.

See Also:

For more information about setting a search path for data files in real mode, see the section on the **append** command in this chapter. If you want to set a search path for external commands, see the section on the **path** command in this chapter.

Example:

Setting a data search path

The following command line tells MS OS/2 to search three directories—\BANDS\COUNTRY, B:\BANDS\ROCK, and \SYMPHONY—to find data files:

```
dpath \bands\country;b:\bands\rock;\symphony
```

The Exit command

Exit

Purpose:

Exits from the command processor (CMD.EXE in protected mode or COMMAND.COM in real mode) and returns to a previous level, if one exists.

Syntax:

exit

Comments:

If you use the **cmd** or **command** command to start a new command processor, you can use the **exit** command to return to the original command processor. **Exit** also lets you exit from an application program to the MS OS/2 command processor, and then return. If a higher-level command processor shell does not exist, nothing happens when you type **exit**.

Notes:

You cannot specify a drive before this command.

See Also:

For more information on protected-mode and real-mode sessions, see the sections on the **cmd** and **command** commands in this chapter.

Example:

Exiting from a new command processor

Suppose you want to start a new command processor within an existing session. You could type the following command to do so:

```
cmd c:\
```

When you had finished using the new command processor, you could return to the previous command processor by typing the following:

```
exit
```

Fdisk

The Fdisk command

Purpose:

Configures a hard disk for use with MS OS/2.

Syntax:

fdisk

Comments:

The **fdisk** command displays a series of menus to help you partition your hard disk for MS OS/2. With **fdisk**, you can do the following:

- Create a primary or extended MS OS/2 partition
- Change the active partition
- Delete an MS OS/2 partition
- Display partition data
- Select the next fixed-disk (hard-disk) drive for partitioning on a system with multiple fixed disks

See Also:

For more information on how to use **fdisk**, see the *Microsoft Operating System/2 Setup Guide.*

Notes:

Fdisk cannot be used with drives that are used by the **subst** or **join** command.

The Find command

Find

Purpose:

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

Syntax:

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

where:
string is a group of characters you want to search for.

Comments:

The **find** command looks for *string* in one or more files. After searching the specified files, **find** displays any lines it has found that contain the specified string.

Uppercase characters in *string* will not match lowercase characters.

String must be enclosed in quotation marks. If *string* contains quotation marks, you must enclose it in an additional pair (""*string*"").

If you omit *pathname*, **find** acts as a filter. It takes input from the MS OS/2 standard input (usually from the keyboard, a pipe, or redirected file) and displays any lines that contain *string*.

Wildcards (* and ?) are not allowed in filenames or extensions.

Find switches

You can use the following switches with the **find** command:

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

If **/c** is specified with **/v**, **find** displays the number of lines that do not contain the string you typed. If **/c** is specified with **/n**, **find** ignores **/n**.

Examples:

Finding a text string in a file

The following command line displays all lines from the file APPLIANC.AD that contain the string "slices and dices":

```
find "slices and dices" applianc.ad
```

The next command line displays all names of the files on the disk in drive B that do not contain the string "date":

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

Suppose you wanted to find the following string in the file STORY.DOC:

```
"Open wide," said Dr. Cole.
```

To do so, you would type the following command line:

```
Find """Open wide," said Dr. Cole."" story.doc
```

The Format command

Format

Purpose:

Formats the disk in the specified drive to accept MS OS/2 files.

Syntax:

format [*drive*:] [**/4**] [**/s**] [**/t:***tracks*] [**/n:***sectors*] [**/v:***label*]

Comments:

The **format** command creates the directory and the file allocation tables on a disk. You must use this command to format all new disks so that MS OS/2 can use them.

Warning Formatting destroys any previously existing data on a disk, and it ignores drive assignments created with the **assign** command.

You must specify the drive that you want to use to format a disk. **Format** then uses the drive type to determine the default format for the disk.

Format switches

You can use the following switches to modify the **format** command:

Switch	Purpose
/4	Formats a 5-1/4-inch, double-sided disk in a high-capacity disk drive. If you are using a single- or double-sided drive, you may not be able to read reliably disks formatted with this switch.
/s	Copies the operating-system files listed in the file FORMATS.TBL from the disk in the default drive to the newly formatted disk. The newly formatted disk must be 1.2 megabytes or greater in size; otherwise, **format** rejects the command. 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).

/t:*tracks* Formats a 3-1/2-inch floppy disk to the number of tracks specified, where *tracks* is the number of tracks on the disk. For 720-kilobyte disks and 1.44-megabyte disks, this value is 80 (**/t:80**).

/n:*sectors* Formats a 3-1/2-inch disk to the number of sectors specified, where *sectors* is the number of sectors per track. For 720-kilobyte disks, this value is 9 (**/n:9**). For 1.44-megabyte disks, this value is 18 (**/n:18**).

/v:*label* *label* is a volume label. The MS OS/2 **format** command automatically prompts you for a volume label whether you specify this switch or not. This switch, which is also supported by MS-DOS 3.3, is maintained for compatibility with DOS. A volume label identifies the disk and can be up to 11 characters in length (no tabs are allowed). An example of a volume label is PROGRAMS.

Formatting a hard 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 ENTER key. (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 you type a volume label that does not match the label on the hard disk, **format** displays the following message:

```
Invalid Volume ID Format failure
```

Otherwise, the command 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 ENTER key. If not, type **n** and press the ENTER key.

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.

Notes:

- You should not use the **format** command with drives used in the **assign**, **join**, or **subst** commands.
- You cannot format drives over a network.

See Also:

For more information about formatting your hard disk, see the *Microsoft Operating System/2 Setup Guide.*

For more information about disk volume labels, see the sections on the **dir**, **label**, and **volume** commands in this chapter.

Examples:

Formatting a floppy disk

To format a floppy disk in drive A and copy the operating system to it, type the following command:

```
format a: /s
```

Graftabl

Real Mode Only

Purpose:

Enables an extended character set to be displayed when using display adapters in graphics mode.

Syntax:

graftabl [*xxx*]

or

graftabl ?

or

graftabl /status

where:
xxx is a code-page identification number.

Comments:

When you type the **graftabl** command by itself, the default United States extended character table is loaded into memory. You may also specify another extended character set by using the *xxx* code-page option. The code pages that are supported by the **graftabl** command include the following:

Value	Code Page
437	United States (default)
860	Portuguese
863	French-Canadian
865	Nordic

If you type the **graftabl** command followed by the **/status** switch, MS OS/2 displays the active character set. If you type **graftabl ?**, MS OS/2 displays the **graftabl** options in addition to the current status of **graftabl**.

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

```
Graphics characters loaded
```

Since you can load the graphics table only once each time you start MS OS/2, you could put the **graftabl** command in your AUTOEXEC.BAT or STARTUP.CMD file to save time. If you try to load the table a second time, **graftabl** displays the following message:

```
Graphics characters already loaded
```

Notes:

- The **graftabl** command increases the size of MS OS/2 resident in memory.
- You may type **/sta** in place of the **/status** switch. Therefore, these two commands perform the same function:

```
graftabl /status
graftabl /sta
```

See Also:

For more information about using code pages, see the section on the **chcp** command in this chapter and the *Microsoft Operating System/2 Setup Guide.*

The *Microsoft Operating System/2 Setup Guide* also provides more information about AUTOEXEC.BAT and STARTUP.CMD files.

Example:

Loading the graphics table

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

```
graftabl
```

Helpmsg

The Helpmsg command

Purpose:

Provides information related to a warning or error message.

Syntax:

helpmsg *messageid*

where:
messageid is the unique identifier for the message in question.

Comments:

Helpmsg is an on-line documentation utility that explains what MS OS/2 messages mean and what action to take in response to each message.

Each MS OS/2 message has a message identification number (*messageid*) that consists of the letters "SYS" followed by a four-digit message number. To find out more about an MS OS/2 message, type **helpmsg** followed by the message identification number.

In response, MS OS/2 displays the original message followed by an explanation of the message and the suggested action to be taken. If the original message refers to a variable, MS OS/2 may use *** in its place.

Examples:

Suppose when you typed a command and pressed the ENTER key, this message were displayed:

```
SYS0100: File not found
```

You could type the following to get more information about this error message:

```
helpmsg sys0100
```

MS OS/2 would respond with the following information:

```
SYS0100: File not found
Explanation: The filename is incorrect or
does not exist.
Action: Check the filename and retry
the command.
```

The Join command

Join

Real Mode Only

Purpose:

Joins an existing disk drive to a specific path.

Syntax:

join [*drive*: *drive*:*path*]

or

join *drive*: **/d**

Comments:

The **join** command lets you access a disk drive by specifying a drive and path. With the **join** command you don't need to name physical drives with separate drive letters. You can refer to all the directories on a specific drive with one path. So, if your program expects to store data files on the default disk, you can use **join** to store the files on another disk.

The **join** command without arguments lists the drives that are currently joined. The **/d** switch disables a previous **join** command.

If the path already existed before you gave the **join** command, you cannot use it while the **join** is in effect. To avoid this, you should specify a path that doesn't exist on the second drive. You cannot join a drive if it is being used by another process.

If the *path* does not exist, MS OS/2 tries to make a directory with that path. The directory must be empty. After you give the **join** command, the first drive name becomes invalid, and if you try to use it, MS OS/2 displays the "Invalid drive" error message.

Notes:

The following commands do not work on drives used in the **join** command (or the **subst** command):

chkdsk	**format**	**recover**
diskcopy	**label**	**sys**
fdisk		

Examples:

Joining a drive

Since you can join a drive only with a root level directory, the following command will work:

```
join e: c:\sales
```

But the preceding example would not work if the pathname were \SALES\REGIONAL instead of \SALES.

To cancel the preceding **join** command, you could type the following:

```
join e: /d
```

If you have a program that stores files in a directory called \DATA on drive A, and you want to save the files on drive C, you can use the following command line:

```
join c: a:\data
```

Now all references to the path A:\DATA will automatically go to drive C.

The Keyb command

Keyb

Purpose:

Specifies a country-specific keyboard layout when one other than a United States keyboard layout is used.

Syntax:

keyb *yy*

where:
yy is a two-letter country code.

Comments:

This command may be typed from a protected-mode session only, but it affects all sessions, including the real-mode session.

To select a special keyboard option, type one of the following two-letter codes:

Code/Keyboard Type	Code/Keyboard Type
US United States	SG Swiss-German
CF French-Canadian	UK United Kingdom
LA Latin America	DK Denmark
NL Netherlands	SV Sweden
BE Belgium	NO Norway
FR France	GR Germany
SP Spain	PO Portugal
IT Italy	SU Finland
SF Swiss-French	

Notes:

You cannot specify a drive before this command.

Example:

To use a German keyboard, type the following command:

```
keyb gr
```

Label

The Label command

Purpose:

Creates or changes the volume label on a disk.

Syntax:

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

where:
label is the new volume label, up to 11 characters.

Comments:

A volume label is a name you can specify for a disk. MS OS/2 displays the volume label of a disk as a part of its directory listing to show you which disk you are using.

If a volume serial number exists, **label** will also display this eight-character number:

```
Volume Serial Number in drive x is nnnn-nnnn
```

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

```
Volume in drive x is xxxxxxxxx
Type a volume label of up to 11 characters or
press Enter for no volume label update: _
```

Naming disks

Type the volume label that you want and press the ENTER key. A volume label may be up to 11 characters in length and may include spaces, but not tabs. Or, you can press the ENTER key immediately if you want to delete the volume label. MS OS/2 will prompt you with the following message:

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

If you type **y**, MS OS/2 deletes the volume label on the disk. Otherwise, the volume label stays the same.

Notes:

- You can use the MS OS/2 **dir** or **vol** command to determine if the disk already has a volume label.
- The **format** command also prompts you to type a volume label when you format a disk.

- **Label** doesn't work on drives used by the **subst** or **join** commands.
- Do not use the following characters in a volume label:

  * ? / \ | . , ; : + = < > [ ] ( ) & ^

Example:

Labeling a disk

To label a disk in drive A that contains sales information for 1987, you might type the following:

```
label a:sales1987
```

Mkdir (md)

The Mkdir (md) command

Purpose:

Makes a new subdirectory in the working directory.

Syntax:

Real Mode:

mkdir [*drive*:]*pathname*

or

md [*drive*:]*pathname*

Protected Mode:

mkdir [*drive*:]*pathname* [[[*drive*:]*pathname*] ...]

or

md [*drive*:]*pathname* [[[*drive*:]*pathname*] ...]

Comments:

The **mkdir** command lets you create a multilevel directory structure. Remember, however, that directories created with **mkdir** are always subdirectories of your working directory unless you explicitly specify a different path with the **mkdir** command.

In protected mode, you can create more than one directory at a time by typing multiple pathnames.

Notes:

You cannot specify a drive before this command because MS OS/2 always assumes that the **mkdir** command is on the current drive. For example, you could not type **a:mkdir newdir**.

Examples:

Creating a subdirectory

If you wanted to create a directory to keep all of your tax information, you could type this from your root directory:

```
mkdir taxes
```

Now, suppose you want to create a directory named RENTAL under the TAXES directory to keep track of information about a duplex that you rent out. To do this from the root directory, you simply type the following command line:

```
mkdir \taxes\rental
```

To create the same subdirectory from the \TAXES directory, you could type either the command listed previously, or type the following:

```
mkdir rental
```

Using multiple pathnames

To create two directories called \CLIENT and \CLIENT\PETE with one command line while working in protected mode, type the following:

```
mkdir \client \client\pete
```

Mode

The Mode command

Purpose:

Sets operation parameters for communication and output devices.

Syntax:

Parallel-printer mode:

mode LPT*n*[**:R**][*chars*][,[*lines*][,**p**]]

Asynchronous-communication mode in real mode:

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

Asynchronous-communication mode in protected mode:

mode COM*m*[**:**] [*baud*[,*parity*][,*databits*] [,*stopbits*] [,[**to**=*state*][,[**xon**=*state*] [, [**idsr**=*state*][,[**odsr**=*state*] [, [**octs**=*state*][,[**dtr**=*dtrstate*][, [**rts**=*rtsstate*]]]]]]]]]]

Display mode:

mode *display*[,[*rows*]]

Comments:

The **mode** command prepares MS OS/2 for communication with devices such as parallel and serial printers, modems, and screens. You can also use it to redirect output.

Parallel-Printer Modes

Modes for parallel printers

For parallel-printer modes, you can use PRN and LPT1 interchangeably. You can use the following options with the **mode** command to set parameters for a parallel printer:

Option	Purpose
n	Specifies the printer number: 1, 2, or 3; the default value is 1.
chars	Specifies characters per line: 80 or 132; the default value is 80.
lines	Specifies vertical spacing, lines per inch: 6 or 8; the default value is 6.
p	Specifies that **mode** try 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.

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

Asynchronous-Communication Modes

Serial communication modes

You can set up asynchronous communications from real mode or protected mode. Note that some serial devices require parameters that can be set only in protected mode, however.

To display the status of a serial device, type the **mode** command followed by the name of the asynchronous port. For example, the following command is used to see the status of the device connected to COM2:

```
mode com2
```

In real mode, MS OS/2 lists the baud rate, parity, data bits, and stop bits. More information is provided when you type the same command from a protected-mode session.

You can use the following options with the **mode** command to set parameters for serial ports:

Option	Purpose
m	Specifies the asynchronous communications (COM) port number, where *m* is a number ranging from 1 to 8. The default value is 1.
baud	Specifies the first two digits of the transmission rate: 110, 150, 300, 600, 1200, 2400, 4800, or 9600.
parity	Specifies the parity: N (none), O (odd), E (even), M (mark—parity equals 1), or S (space—parity equals 0). The default value is E, meaning even parity.
databits	Specifies the number of data bits: 5, 6, 7, or 8 bits of data. The default value is 7.
stopbits	Specifies the number of stop bits: 1, 1.5, or 2. If *baud* is 110, the default value is 2; otherwise, the default value is 1. If 1.5 stopbits is specified, *databits* must be 5.

to=*state*	Specifies whether an infinite timeout processing is enabled (on) or whether normal timeout processing is to be used (off). The default is **to=off**.
xon=*state*	Specifies whether automatic transmit flow control is enabled (on) or disabled (off). The default is **xon=off**.
idsr=*dsrstate*	Specifies whether the input handshake using DSR (Data Set Ready) is enabled (on) or disabled (off). The default is **idsr=on**.
odsr=*dsrstate*	Specifies whether the output handshake using DSR (Data Set Ready) is enabled (on) or disabled (off). The default is **odsr=on**.
cts=*state*	Specifies whether output handshake using CTS (Clear to Send) is enabled (on) or disabled (off). The default is **octs=on**.
dtr=*state*	Determines the state of DTR (Data Terminal Ready): on (enable DTR), off (disable DTR), or hs (enable DTR handshaking). The default is **dtr=on**.
rts=*state*	Determines the state of RTS (Request to Send): on (enable RTS), off (disable RTS), hs (enable RTS handshaking), tog (enable RTS toggling). The default is **rts=on**.
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 **to**, **xon**, **idsr**, **odsr**, **cts**, **dtr**, and **rts** parameters may be listed in any order following the *stopbits* parameter.

Setting display modes

Display modes

You can use the following options with the **mode** command to set parameters for a display:

Option	Purpose
display	Specifies one of the following values: 40, 80, BW40, BW80, CO40, CO80, or MONO. For each of these options, 40 and 80 indicate the number of 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. The default value is 40.
rows	Specifies the number of rows per screen: 25, 43, or 50. The display-adapter type determines which of these values are valid. The default value is 25.

When you type **mode** with the **display** option, it only affects the current session.

Notes:

- If you want to redirect output from a parallel printer to a serial device, use the **spool** command described in this chapter.
- If you want to print files whenever you start MS OS/2, include **mode** commands in your AUTOEXEC.BAT or INITENV.CMD file.
- You must install the device driver before you can use an asynchronous port in protected mode.

See Also:

For more information about the AUTOEXEC.BAT and INITENV.CMD files, or about installable device drivers for asynchronous ports, see the *Microsoft Operating System/2 Setup Guide.*

Examples:

Sending output to a parallel printer

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 this:

```
mode lpt2: 80,8
```

Or, since 80 characters per line is the default value, you could type the following:

```
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 command line:

```
mode lpt2:80,8,p
```

To stop retrying to print, you can press CONTROL+BREAK, or type the **mode** command without the **p** option.

The More command

More

Purpose:

Sends output to the console one screen at a time.

Syntax:

more < *source*

or

source | **more**

where:
source is a file or command.

Comments:

Displaying output screen-by-screen

More is a filter that reads from standard input (from a pipe or redirected file) and displays one screen of information at a time. **More** is commonly used to view long files.

For example, you may use the **dir** command, the **sort** command, or a filename as a source. The **more** command then displays one screen of information and the message "--More--" at the bottom of your screen. Press the ENTER key to display another screen of information, then keep pressing it until you have read all the data.

Notes:

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, **more** will not work.

See Also:

For more information about using redirection symbols with commands, see Chapter 6, "Using MS OS/2 Commands."

Example:

Viewing long files

Suppose you have a long file called CLIENTS.NEW that you want to view on your screen. The following command redirects the file through the **more** command to show the file's contents one screen at a time:

```
more < clients.new
```

Patch

The Patch command

Purpose:

Changes program code.

Syntax:

patch [[*drive*:]*pathname*][**/a**]

Comments:

The **patch** command lets you make patches to MS OS/2 or application-program code. A patch is a section of program code inserted into an existing program to change the way the program runs. **Patch** can change bytes at any position in a file or add bytes to the end of a file. Any file that can be written to can be patched.

Changing program code

Patch has two modes of operation: automatic and interactive.

The **/a** switch specifies automatic operation. With the **/a** option, the MS OS/2 PATCH.COM file runs on a prespecified drive and pathname that contain patch instructions for patching one or more files automatically.

Interactive mode is the default. In this mode, you supply the name of the file you want to patch on the **patch** command line. **Patch** then prompts you for the offset at which a patch is to be made and the patch contents. You must enter both the offset and the patch contents in hexadecimal notation.

After you supply the hexadecimal offset, the sixteen bytes at that offset are displayed. You can then change any or all of the sixteen bytes. If you decide not to change any, you can press the ESCAPE key.

The cursor is initially positioned on the first byte. To change this byte, type one or two hexadecimal digits. To leave the byte unchanged and move to the next byte, press the SPACEBAR. Press the BACKSPACE key to move the cursor back if you make a mistake. You can continue changing bytes until you press the ENTER key. If you move the cursor past the sixteenth byte, the next sixteen bytes are displayed. This cycle continues until you press the ENTER key.

After you press ENTER, **patch** saves the patch information. MS OS/2 then asks if you want to make any more patches. If you type **y**, **patch** again prompts you for an offset. Any more patch requests are also saved. After all

patches are entered, MS OS/2 displays them on the screen and asks if they should be applied. If you type **y**, all of the saved patch requests are written to disk in the same order they were entered.

Notes:

- You should use **patch** only if you understand the need for a patch, how to make the patch, and the effect the patch will have on program operation.
- Before you use the **patch** command, be sure to back up the files to which the patches will be applied.

Path

The Path command

Purpose:

Specifies a command search path.

Syntax:

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

or

path ;

Comments:

The **path** command specifies which directories MS OS/2 should search for external commands—after it searches your working directory. This search path is valid until you specify another path, or until you exit from MS OS/2. The default value is no path.

To specify multiple search paths, separate each path with a semicolon. MS OS/2 searches the paths in the order specified in the **path** command. If you type an external command, and if that command exists in more than one directory, MS OS/2 will run the first copy of the command that it finds in the search path.

To see the current path, type the **path** command without options.

If you use the following command, MS OS/2 sets the NUL path:

```
path ;
```

This means that MS OS/2 searches only the working directory for external commands.

Notes:

- You cannot specify a drive name before this command.
- When you install MS OS/2 with the Install program, the **path** command is automatically placed in the AUTOEXEC.BAT and INITENV.CMD files.

See Also:

For more information about the Install program, or about the AUTOEXEC.BAT and INITENV.CMD files, see the *Microsoft Operating System/2 Setup Guide.*

For more information about setting search paths for data files in real and protected modes, see the sections on the **append** and **dpath** commands, respectively.

Example:

Setting a search path

Suppose you want MS OS/2 to search for external commands first in a directory called OS2 on drive C, and then in a directory called FASTAPPS on drive A. To do this, use the following command:

```
path c:\os2;a:\fastapps
```

Print

The Print command

Purpose:

Prints a text file on a lineprinter while you are processing other MS OS/2 commands (usually called background printing).

Syntax:

print [**/d:***device*] [**/t**] [**/c**] [*drive***:**][*pathname*] [...]

Comments:

You can use the **print** command only if you have an output device, such as a printer or a plotter, connected to one of your computer's serial or parallel ports.

Print switches

You can use the following switches with the **print** command:

Switch	Purpose
/d:*device*	Specifies the print device name. The default device is LPT1. Other possible print device names for parallel ports are PRN, LPT2, and LPT3. COM*x*, where *x* is a number from 1 to 8, refers to a serial port. (LPT1 and PRN both refer to the first parallel port on your computer.)
/t	Deletes all files in the print queue (those waiting to be printed).
/c	Turns on the cancel mode and removes the preceding filename and all following filenames from the print queue.

Printing output to your screen

The **print** command, when used with no options, displays the contents of the print queue on your screen without affecting the queue.

Notes:

- Each print-queue entry may contain a maximum of 64 characters, including the drive name. So you may need to change directories first to avoid using extensive pathnames.
- Some applications have their own print commands. You should use the application's print facility to print files that you create with the application.

- You must install the device driver before you can use an asynchronous port in protected mode.

See Also:

For more information about installable device drivers for asynchronous ports, see the *Microsoft Operating System/2 Setup Guide.*

Examples:

Printing a file

To print the file PENCIL.TST on the first parallel printer connected to your computer, you would type the following command:

```
print /d:lpt1 pencil.tst
```

The following command empties the print queue for the device named LPT1:

```
print lpt1 /t
```

The next command removes the PENCIL.TST file from the default print queue:

```
print a:pencil.tst /c
```

Prompt

The Prompt command

Purpose:

Changes the MS OS/2 command prompt.

Syntax:

prompt [[*text*][*characters*]...]

Comments:

The **prompt** command lets you change the MS OS/2 system prompt. If you do not type a new value after **prompt**, the prompt is set to the default value, which includes the default drive designation (for example, A>). Note that the protected-mode system prompt differs from the real-mode system prompt. The default real-mode system prompt is the current drive letter followed by the greater-than symbol (>). The default protected-mode system prompt is the current drive letter enclosed within brackets, for example, [C:\].

Valid Prompt characters

You can type the following characters after the **prompt** command to create special prompts:

Characters	Prompt
$$	Dollar sign ($)
$t	Current time
$d	Current date
$p	Working directory of the default drive
$v	Version number
$n	Default drive
$g	Greater-than symbol (>)
$l	Less-than symbol (<)
$b	Pipe symbol (¦)
$_	Activates the ENTER and LINEFEED keys
$e	ANSI escape code
$q	Equal sign (=)
$h	Backspace character (to erase a character that has been written to the prompt line)
$c	Left parenthesis [(]
$f	Right parenthesis [)]
$a	Ampersand (&)

Notes:

The default real-mode prompt and protected-mode prompt are purposely distinguishable from one another. If you modify these prompts, it would be a good idea to keep them unique.

See Also:

For more information about using ANSI escape sequences, see the section on the protected-mode **ansi** command in this chapter. See the *Microsoft Operating System/2 Setup Guide* for information about the real-mode ANSI.SYS installable device driver and a list of ANSI escape sequences.

Examples:

Setting your prompt

The following example sets the drive prompt to *drive*: *current directory* >:

```
prompt $p$g
```

For example, if you are currently on drive C in the \ACCOUNTS\OVERDUE directory, your prompt would look like this:

```
c:\accounts\overdue >_
```

You can type the following command line to create a prompt that tells you the time and the drive and directory you're working in:

```
prompt The time is $t$h$h$h$h$h$h$_$p $g
```

This results in a prompt of the following form:

```
The time is 13:37
C:\DOS >
```

Note that the **$h** characters were used to erase seconds and hundredths of seconds from the time display.

Using escape sequences

If your terminal has an ANSI escape-sequence driver, then you can use escape sequences in your prompts. The following command, for example, sets the prompts in inverse video mode and returns to video mode for other text:

```
prompt  $e[7m$n:$e[0m
```

Recover

The Recover command

Purpose:

Recovers a file or disk containing bad sectors.

Syntax:

recover [*drive*:][*path*]*filename*

or

recover *drive*:

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 OS/2 to read the file sector by sector and to skip the bad sectors. When MS OS/2 finds a bad sector, it no longer allocates your data to that sector.

Notes:

- The **recover** command does not work on a network from a remote work station.
- **Recover** doesn't work on drives used in the **subst** or **join** commands.

Examples:

Recovering a disk

Suppose you used the **chkdsk** command to check a file named PENCIL.AD, and **chkdsk** found a few bad sectors on the disk on which this file had been written. To recover this file you would use the following command:

```
recover pencil.ad
```

To recover an entire disk in drive A you would use the following command:

```
recover a:
```

The Rename (Ren) command

Rename (Ren)

Purpose:

Changes the name of a file.

Syntax:

rename [*drive:*]*path filename1 filename2*

or

ren [*drive:*]*path filename1 filename2*

where:
filename1 is the old name.
filename2 is the new name.

Comments:

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

The **ren** command renames files within a directory. You can use wildcards (* or ?) in either *filename* option. The **ren** command renames all files matching the first *filename*. If wildcards appear in the second *filename*, **ren** does not change the corresponding character positions.

You cannot use **ren** to move files from one drive or directory to another. If you try to type a drive or path in front of the second *filename*, MS OS/2 sends an error message and your file is not renamed.

Notes:

You cannot specify a drive for this command.

Examples:

Renaming files

The following command changes the extension of all filenames in the current directory ending in .TXT to .DOC:

```
ren *.txt *.doc
```

In the next example, **ren** renames a file named CHAP10.TXT in the \MEMOS directory on drive C to PART10.TXT:

```
ren c:\memos\chap10.txt part10.txt
```

The newly renamed file, PART10.TXT, remains in the \MEMOS directory on drive C.

Replace

The Replace command

Purpose:

Updates files on your hard disk with new versions of software.

Syntax:

replace [*drive*:]*pathname1* [*drive*:][*pathname2*] [**/a**][**/p**][**/r**][**/s**][**/w**]

where:
*pathname*1 is the source path and filename.
*pathname*2 is the target path and filename.

Comments:

Replace performs two functions:

- It replaces files in the target directory with files in the source directory that have the same name.
- 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.

Replace switches

You can use the following switches to modify the result of the **replace** command:

Switch	Purpose
/a	Adds new files to the target directory instead of replacing existing ones. You may not use this switch with the **/s** switch.
/p	Prompts you with the following message before it replaces a target file or adds a source file: `Replace` *filename*`? (Y/N)_`
/r	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 replacement process.

/s	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.
/w	Waits for you to insert a disk before beginning to search for source files. If you do not specify the **/w** switch, **replace** begins replacing or adding files immediately.

If you specify **/w** but not **/a**, **replace** displays the following message:

```
Press any key to begin replacing files
```

If you specify both **/w** and **/a**, **replace** displays the following message:

```
Press any key to begin adding file(s)
```

As files are replaced or added, **replace** displays the filenames on the screen. At the conclusion of the **replace** operation, it displays a summary line:

nnn `file(s) added/replaced`

or

```
No files added/replaced
```

Notes:

- You cannot use the **replace** command to update hidden files or system files.
- You may use ? or * wildcards in source filenames.

Examples:

Replacing files

Suppose your hard disk, drive C, contains several files named PHONES.CLI that contain client names and phone numbers. To replace these files with the latest version of the PHONES.CLI file on the disk in drive A, you would type

```
replace a:\phones.cli c:\ /s
```

This command line 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. To do this, you would type the following:

```
replace a:*.prd c:\mstools /a
```

This command adds any files from the default directory of drive A with an extension of .PRD (that do not currently exist in the MSTOOLS directory on drive C) to C:\MSTOOLS.

The Restore command

Restore

Purpose:

Restores files that were backed up using the **backup** command.

Syntax:

restore *drive1*: [*drive2*:][*pathname*] [**/s**] [**/p**] [**/b:***date*] [**/a:***date*] [**/e:***time*] [**/L:***time*] [**/m**] [**/n**]

where:
*drive*1 contains the backed-up files.
*drive*2 is the target drive.
pathname identifies the file(s) you want to restore.

Comments:

The **restore** command can restore files from similar or dissimilar disk types.

Restore switches

You can use the following switches with the MS OS/2 **restore** command:

Switch	Purpose
/s	Restores subdirectories as well as files in the specified directory.
/p	Prompts for permission to restore any files matching the file specification that are read-only or that have changed since the last backup.
/b:*date*	Restores only those files last modified on or before *date*.
/a:*date*	Restores only those files last modified on or after *date*.
/e:*time*	Restores only those files last modified at or earlier than *time*.
/L:*time*	Restores only those files last modified at or later than *time*.
/m	Restores only those files modified since the last backup.
/n	Restores only those files that no longer exist on the target disk.

Once MS OS/2 has restored the file, use the **dir** or **type** command to make sure that the file was restored properly.

Notes:

- **Restore** cannot restore the system files named OS2BIO.COM, OS2DOS.COM, CMD.EXE, and COMMAND.COM. Use the **sys** command to restore OS2BIO.COM and OS2DOS.COM. Use the **copy** command to restore CMD.EXE and COMMAND.COM.
- The **restore** command will restore files backed up with the MS OS/2 **backup** command, or with the MS-DOS **backup** command. (This includes MS-DOS versions 3.2 and earlier, which back up files using a different backup file structure.)

See Also:

For more information about restoring and backing up files, see the section on the **backup** command in this chapter.

Example:

Restoring a file

To restore the \ADVERTIS\INVEST.MNT file from the backup disk in drive A to the ADVERTIS directory on drive C, type the following:

```
restore a: c:\advertis\invest.mnt
```

The Rmdir (rd) command

Rmdir (rd)

Purpose:

Removes a directory from a multilevel directory structure.

Syntax:

Real Mode:

rmdir [*drive*:]*path*

or

rd [*drive*:]*path*

Protected Mode:

rmdir [*drive*:]*path*[...]

or

rd [*drive*:]*path*[...]

Comments:

Rmdir removes a directory that is empty except for the listings for the working directory (.) and the parent directory (..). Before you can remove a directory entirely, you must delete its files and subdirectories.

Using multiple pathnames

In protected mode, you can type more than one path on the **rmdir** command line to remove more than one directory at a time.

Notes:

- You cannot specify a drive name before for this command.
- You cannot remove a directory that contains hidden files. In addition to MS OS/2, which has some hidden files, certain application programs also create their own hidden files.

Example:

If you wanted to remove a directory named \TIMESLOT\FALL82, you would first see if the directory is empty by typing the following:

```
dir \timeslot\fall82
```

Then, from any directory except the one you want to remove, type the following command:

```
rmdir \timeslot\fall82
```

If you are working in the same directory that you are trying to remove, you'll receive the following error message:

```
Invalid path, not directory, or directory
not empty.
```

The Set command

Set

Purpose:

Equates one string of characters in the environment with another string for later use in programs.

Syntax:

set [*string1*=[*string2*]]

where:
string1 is any set of characters.
string2 is the assigned equivalent to *string1*.

Comments:

You should use this command only if you want to set values for programs you have written.

When MS OS/2 sees a **set** command, it inserts the given *string* and its equivalent into a part of memory reserved for the environment. If *string* already exists in the environment, it is replaced with the new one.

If you specify just *string1*, **set** removes any previous setting of that string from the environment. Or, if you type **set** with no options, MS OS/2 displays the current environment settings.

The **set** command is especially useful in the AUTOEXEC.BAT and INITENV.CMD files. This way, strings or parameters are automatically set the first time you start a real-mode session (with AUTOEXEC.BAT) or a protected-mode session (with INITENV.CMD). For more information about the AUTOEXEC.BAT and INITENV.CMD files, see the *Microsoft Operating System/2 Setup Guide.*

Setting parameter values

In 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 command line **type** *%file%*, you can use the **set** command to set the name that MS OS/2 will use for that variable. In the following command line, **set** replaces the *%file%* parameter with the filename TAXES.86 :

```
set file=taxes.86
```

Therefore, to change the parameter names, you need not edit each batch file.

Notes:

You cannot specify a drive name before this command.

Example:

The following command sets the string "include" to "c:\inc":

```
set include=c:\inc
```

Setcom40

Real Mode Only

Purpose:

Sets or clears the apparent COM port availability to some real-mode application programs.

Syntax:

setcom40 com*x*=on

or

setcom40 com*x*=off

where:
x is a number from 1 to 8 that identifies the serial port.

Comments:

The **setcom40** command sets the appropriate serial port address before starting the MS-DOS application in real mode that will use the port. You will need to use this command if you installed COM.SYS (or a comparable driver) with the **device** command in CONFIG.SYS.

Often, MS-DOS applications use a particular area on the disk (the BIOS COM port base addresses, which begin at storage location 400H) to determine the presence of serial ports (COM1 through COM8). The MS OS/2 installable device driver COM.SYS places zeros in these locations to prevent real-mode programs from interfering with protected-mode programs that may try to use those ports.

If **com*x*=on**, MS OS/2 will write the port address for COM*x* in the BIOS data area of the disk. If **com*x*=off**, MS OS/2 will remove the address from the BIOS data area when the real-mode program no longer needs it.

Notes:

- The **setcom40** command does not affect settings made with the **mode** command.
- Real-mode programs should not try to use serial ports that are being used by protected-mode programs, or by the **spool** command.
- For real-mode programs using serial ports, it is important to end the program before switching out of the real-mode session. If you switch to a protected-mode session or program selector before the real-mode program has ended, that program may end abnormally.

- Two types of real-mode programs may have problems using serial ports: those using intermittent hardware interrupts and those that are time-dependent.

  Some BASIC programs will interfere with the port even when not performing input/output to COM if the address space (40: area) is not zero. Real-mode programs that are interrupt-driven and execute too long on the interrupt thread will severely degrade the protected-mode environment.
- **Setcom40** affects COM1 through COM3 if any is supported by a device driver that names the port in its device header.

See Also:

For more information about the COM.SYS installable device driver, see the *Microsoft Operating System/2 Setup Guide.*

Example:

Suppose you have a serial printer on COM1 and have installed the COM.SYS device driver with a **device** command in CONFIG.SYS. If you want to print out some information from a real-mode program to that printer, you would type this command line:

```
setcom40 com1=on
```

The Sort command

Sort

Purpose:

Reads input, sorts the data alphanumerically, then writes the sorted data to your screen, to a file, or to another device.

Syntax:

[*source*] ¦ **sort** [**/r**] [**/+***n*]

or

sort [**/r**] [**/+***n*] < *source*

where:
source is a filename or command.

Comments:

The **sort** command is a filter that lets you alphabetize a file according to the character in a certain column. The **sort** command uses the collating sequence table, based on the country code and code-page settings.

The pipe (¦) and less-than (<) redirection symbols direct data through the **sort** command from *source*. For example, you may use the **dir** command, or a filename as a source. You may use the **more** command or a filename as a destination.

Sort switches

The **sort** command accepts these switches:

Switch	Purpose
/r	Reverses the sort; that is, sorts from Z to A, and then from 9 to 0.
/+*n*	Sorts the file according to the character in column *n*, where *n* is some number. If you do not specify this switch, the **sort** command sorts the file according to the character in the first column.

Unless you specify a source, **sort** acts as a filter and accepts input from the MS OS/2 standard input (usually from the keyboard, from a pipe, or redirected from a file).

Notes:

- **Sort** does not distinguish between uppercase and lowercase letters.
- Characters above ASCII code 127 are sorted based on information found in the CONFIG.SYS file.

See Also:

For more information about using redirection symbols with commands, see Chapter 6, "Using MS OS/2 Commands."

For more information about country-code and code-page settings in your CONFIG.SYS file, see the *Microsoft Operating System/2 Setup Guide.*

Examples:

Sorting a file

The following command reads the file EXPENSES.TXT, sorts it in reverse order, and displays it on your screen:

```
sort /r < expenses.txt
```

The following command line 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 line does the same thing as the previous one, except that the **more** filter displays the sorted directory one screen at a time:

```
dir | sort /+14 | more
```

The Spool command

Spool

Purpose:

Starts the printer spooler for background printing while you are processing other MS OS/2 commands or programs.

Syntax:

spool[*drive:*][*path*] [**/d:***device1*] [**/o:***device2*]

where:
[*drive:*][*path*] specifies the print-spool subdirectory, where the temporary spool files will be created. The default is \SPOOL.

Comments:

Spool lets you send more than one job to the printer at a time while you run other programs or MS OS/2 commands. **Spool** does this by intercepting data that is going to the printer device drivers. It separates each print job in a queue so that the jobs don't intermix.

If your application program is still writing or updating a print file, **spool** stores the file in a separate temporary file in the print-spool subdirectory. When the application program is finished and the file is ready to print, the file is put into the print queue, and prints as soon as the printer is available.

Spool switches

There are two **spool** switches:

Switch	Purpose
/d:*device1*	Names the parallel port connected with the print device. MS OS/2 uses LPT1 as a default. The input device (*device1*) may not be a serial printer. Any character device (printer, plotter, etc.) that supports monitors may be used. Only the output portion of the device is affected.
/o:*device2*	Names the output print device. You may specify any parallel and serial port. LPT1, LPT2, LPT3, and PRN name parallel ports. COM*x*, where *x* is a number from 1 to 8, names a serial (asynchronous) port. You may use this switch to redirect output. MS OS/2 uses **/d:***device* as a default.

If **spool** does not find the default spool directory, \SPOOL, or other specified directory, it displays this message and the command is not executed:

```
The system cannot find the spool subdirectory
specified.
```

Notes:

- You cannot specify a drive letter before this command.
- To use the **spool** command, you must specify the **/t** or **/c** switch with the **print** command.
- To send files to a separate spool directory, you must first create the directory with the **mkdir** command.
- You must install the COM.SYS device driver before you can use an asynchronous port in protected mode.

See Also:

For more information, see the **print** and **mkdir** commands in this chapter.

Example:

The following command line would spool your print jobs in a directory called \PRINTER and would print them on a serial printer connected to COM1:

```
spool \printer /d:com1
```

Start

Protected Mode Only

Purpose:

Begins a new process in a new protected-mode session.

Syntax:

start ["*session name*"] [**/c**] *command* [*arguments*]

where:
session name is the name that will be displayed by the program selector.
command initiates the protected-mode program or MS OS/2 command.
arguments are any valid arguments for *command.*

Comments:

The **start** command is used to start another program or command in a new protected-mode session. You can switch to the new protected-mode session, just as you can switch to any other session, by using the program selector.

The **/c** switch tells the CMD.EXE command processor to perform *command*, then end the session and return automatically to the program selector when *command* is completed. Once the command processor returns to the program selector, the *session name* will no longer appear on the program-selector screen.

The *session name* may be up to 30 characters long, and must be surrounded by quotation marks.

If you type **start** without the *command* option, MS OS/2 starts a new protected-mode session by initiating the CMD.EXE command processor.

Notes:

You can use the **start** command one or more times in your INITENV.CMD file to begin programs each time you begin MS OS/2.

See Also:

For more information about CMD.EXE, see the section on the **cmd** command in this chapter.

Example:

Suppose you wanted to start a program named Videophile in a new protected-mode session. If the command that initiated this program is **videop**, you would type this command to run the program in its own session:

```
start "Videophile" /c videop
```

Because you included the **/c** switch, when your program ended, you would return to the program-selector screen.

The Subst command

Subst

Purpose:

Substitutes a drive letter for a path.

Syntax:

subst [*drive*: *drive*:*path*]

or

subst *drive*:[*path*] **/d**

Comments:

Making a pathname alias

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 physical drive.

When MS OS/2 finds a command that uses a virtual drive, it replaces the drive letter with the path and treats that new drive letter as though it belonged to a physical drive.

If you type the **subst** command without options, MS OS/2 displays the names of the virtual drives in effect.

Use the **/d** switch to delete a virtual drive.

Notes:

The following commands do not work on drives used in the **subst** command (or the **join** command):

chkdsk	**format**	**recover**
diskcopy	**label**	**sys**
fdisk		

Example:

Creating a virtual drive

The following command line creates a virtual drive, Z, for the pathname B:\RERUNS\THRILLER\MONSTER:

```
subst z: b:\reruns\thriller\monster
```

Now, instead of typing the full pathname, you can move to this directory by simply typing the name of the virtual drive:

```
z:
```

Sys

The Sys command

Purpose:

Transfers the MS OS/2 system files from the disk in the default drive to the disk in the specified drive.

Syntax:

sys *drive:* [**/s**]

Comments:

Updating your system

You usually use the **sys** command to update the system or place it on a formatted disk that contains no files. You must include the drive letter of the target disk with this command. You must also be working in the directory in which the MS OS/2 system files are located.

The **/s** switch copies the remaining system files listed in the FORMATS.TBL file from the source disk to the newly formatted target disk.

Notes:

- **Sys** does not work on drives used by the **subst** or **join** command.
- **Sys** does not work on a network.
- **Sys** does not transfer the COMMAND.COM (the real-mode command processor) file or CMD.EXE (the protected-mode command processor) file. Use the **copy** command to transfer these files to the target disk.

Example:

If you wanted to copy the MS OS/2 system files from your working directory to a disk in drive A, you would type the following:

```
sys a:
```

The Time command

Time

Purpose:

Specifies the time to the system.

Syntax:

time [*hours*:*minutes*[:*seconds* [.*hundredths*]]]

Comments:

The **time** command sets the internal clock in your computer. MS OS/2 keeps track of time in a 24-hour format, and uses the time information to update directory listings whenever you create or change a file.

Displaying the current time

The **time** command without options displays the current time, and gives you an opportunity to change it:

```
Current time is hh:mm:ss.cc
Enter new time:_
```

If you do not want to change the time shown, simply press the ENTER key.

If you do want to change the time, type in a new value in the 24-hour clock format. The following are valid values:

hours = 0—23
minutes = 0—59
seconds = 0—59
hundredths = 0—99

These elements can be separated by colons (:) or periods (.). Seconds and hundredths of seconds are optional.

If you do not type a valid time, MS OS/2 displays the following message and then waits for you to type a valid time:

```
Invalid time
Enter new time:_
```

You can also type the new time directly on the command line.

Notes:

- You can change the **time** command format by changing the **country** command in the CONFIG.SYS file. For more information, see the *Microsoft Operating System/2 Setup Guide.*

- You cannot specify a drive for this command.

Example:

To reset the time of day on your computer's clock, you can type the **time** command by itself and MS OS/2 will prompt you for the correct time. Or you can include the correct time when you type the command. For example, if you want to set your computer's clock at 1:36 p.m., you could type the following command:

```
time 13:36
```

The Tree command

Tree

Purpose:

Displays the path (and, as an option, lists the contents) of each directory and subdirectory on the given drive.

Syntax:

tree [*drive*:] [**/f**]

Comments:

Finding directory names

The **tree** command lists the full path of each directory on the specified drive, along with the names of their subdirectories.

The **/f** switch displays the names of the files in each directory.

Notes:

Another way to list all of the subdirectories in your working directory is to type the **dir *** command line. This will also report all files that have no file extensions. Directories are identified with the label <DIR>.

Examples:

If you want to see names of all directories and subdirectories on your computer, simply type the following:

```
tree
```

If you also want to see the files in all of the directories on drive C one screen at a time, type this command line:

```
tree c: /f ¦ more
```

Printing a Tree listing

To print that same list on a printer, use the following command:

```
tree c: /f > prn
```

Type

The Type command

Purpose:

Displays the contents of a text file on the screen.

Syntax:

Real Mode:

type [*drive*:][*path*]*filename*

Protected Mode:

type [*drive*:][*path*]*filename* [...]

Comments:

You can use the **type** command to view a text file without modifying it. (Use **dir** to find the name of a file and Edlin to change the contents of a file.) In protected mode, you can type more than one filename on the **type** command line. The files will be displayed in the order that you typed them.

Note that when you use **type** to display a file that contains tabs, all the tabs are expanded to the current setting for tabs (generally eight spaces wide). Also, if you try to display a binary file or a file created by an application program, you may see unusual characters on the screen, such as bells, formfeeds, and escape-sequence symbols.

Notes:

You cannot specify a drive name before this command.

Examples:

Displaying a file

To display the contents of a file called HOLIDAY.MAR, you would type the following command line:

```
type holiday.mar
```

In protected mode, to display two short files named HOLIDAY.APR and HOLIDAY.MAY, you would type the following command line:

```
type holiday.apr holiday.may
```

If the file you want to display is fairly long, you could use a command line like this to display the file's contents one screen at a time:

```
type holiday.88 ¦ more
```

The Ver command

Ver

Purpose:

Displays the MS OS/2 version number.

Syntax:

ver

Comments:

If you want to know what version of MS OS/2 you are using, you simply type the **ver** command. The version number then appears on your screen.

Notes:

You cannot specify a drive for this command.

Example:

Displaying the MS OS/2 version number

Suppose you had a partition on your hard disk on which you had loaded MS OS/2 version 10.00. If you typed the **ver** command, the following message would appear:

```
MS OS/2 Version 10.00
```

Verify

The Verify command

Purpose:

Turns verification 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 (free from bad sectors, for example). MS OS/2 verifies the data as it is written to a disk. You will receive an error message only if MS OS/2 is unable to successfully write your data to a disk.

Notes:

You cannot specify a drive name before this command.

Examples:

Displaying the current verify setting

If you want to know the current setting of **verify**, use the **verify** command without an option:

```
verify
```

MS OS/2 displays a message like this:

```
VERIFY is off
```

Once you have typed **verify on**, verification remains in effect until a program changes it, or until you type the following:

```
verify off
```

The Vol (Volume) command

Vol

Purpose:

Displays a disk's volume label if one exists.

Syntax:

Real Mode:

vol [*drive*:]

Protected Mode:

vol [*drive*:][...]

Comments:

Displaying a volume label

This command displays the volume label of the disk in a specific drive. If you do not type a drive letter, MS OS/2 displays the volume label of the disk in the default drive. In protected mode, you can type more than one drive name on the command line, and **vol** displays their volume labels consecutively.

Notes:

You cannot specify a drive name before this command. For example, if you typed **b:vol c:**, MS OS/2 would display an error message. MS OS/2 assumes that this command resides on the drive you are working from.

See Also:

For more information about how MS OS/2 uses volume labels, see the sections on the **label** and **format** commands in this chapter.

Examples:

If you want to find out what the volume label for the disk in drive A is, you would type the following:

```
vol a:
```

If the volume label is NEW IDEAS, MS OS/2 responds by displaying the following message:

```
Volume in drive A is NEW IDEAS
```

Listing more than one label

In protected mode, you can specify more than one drive name with the **vol** command; **vol** then displays their volume labels consecutively. For example, if you typed **vol c: d:**, the output might look like this:

```
Volume in drive C is CAUSE
Volume in drive D is EFFECT
```

The Xcopy command

Xcopy

Purpose:

Copies files and directories, including subdirectories if they exist.

Syntax:

xcopy [*drive*:][*pathname1*] [[*drive*:][*pathname2*]] [**/s**] [**/e**] [**/p**] [**/v**] [**/a**] [**/m**] [**/d:***date*]

Comments:

The first set of *drive*: and *pathname* parameters specifies the source file or directory that you want to copy. The second set names 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 *pathname* option, **xcopy** uses the default directory with the default filename, *.*.

Xcopy switches

The following switches modify the result of the **xcopy** command:

Switch	Purpose
/s	Copies directories and subdirectories, unless they are empty. If you omit this switch, **xcopy** works within a single directory.
/e	Copies any subdirectories, even if they are empty. You must use the **/s** switch with this switch.
/p	Prompts you with "(Y/N?)," allowing you to confirm whether you want to create each target file.
/v	Verifies each file as it is written to the target to make sure that the target files are identical to the source files.
/a	Copies source files that have their archive bit set. Does not modify the archive bit of the source file.

/m	Same as the **/a** switch, but after copying a file, it turns off the archive bit in the source file.
/d:*date*	Copies source files modified on or after the specified date. The *date* format may vary depending on the country code that you are using.

Notes:

Copying to a disk with a different 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 should use the **xcopy** command rather than **diskcopy**. The **diskcopy** command copies disks track-by-track; it requires your source and target disks to have the same format.

See Also:

For information on setting the archive attribute, see the section on the **attrib** command in this chapter.

For information on the *date* format, see the section on the **date** command in this chapter.

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.

Appendix MS OS/2 Command Exit Codes

When a program ends, it returns an exit code (sometimes called an error code), a number that indicates whether an error occurred while the program was running. Zero is returned if no error occurred; a value greater than zero is returned if there was a problem. Exit codes generally don't appear on screen.

You can use the **if** and **goto** batch commands to perform error processing based on the errorlevel (the exit code) returned by any of the Microsoft Operating System/2 commands listed in this appendix. For more information about these commands, see Chapter 7, "MS OS/2 Command Reference."

The following exit codes are grouped according to the MS OS/2 command that issues them.

Backup Command Exit Codes

Backup exit codes

Upon completion, the **backup** command returns one of the following exit codes:

Code	Meaning
0	The files were backed up successfully.
1	No files were found to back up.
2	Some files were not backed up due to sharing conflicts.
3	You terminated the command.
4	An error caused the command to end.

Diskcomp Command Exit Codes

Diskcomp exit codes

Upon completion, the **diskcomp** command returns one of the following exit codes:

Code	Meaning
0	The disks compared exactly.
1	The disks are not the same.
2	You terminated the command.
3	A nonrecoverable read or write error occurred; the disks did not compare.
4	There is not enough memory to compare the disks, or you specified invalid drives or typed an invalid command-line syntax.

Diskcopy Command Exit Codes

Diskcopy exit codes

Upon completion, the **diskcopy** command returns one of the following exit codes:

Code	Meaning
0	The disk was copied successfully.
1	A nonrecoverable but nonfatal read or write error occurred.
2	You terminated the command.
3	**Diskcopy** was unable to read the source disk or format the target disk.
4	There is not enough memory to copy the disk, or you specified invalid drives or typed an invalid command-line syntax.

Format Command Exit Codes

Format exit codes

Upon completion, the **format** command returns one of the following exit codes:

Code	Meaning
0	The disk was formatted successfully.
3	You terminated the command.
4	A fatal error occured (any error other than 0, 3, or 5).
5	You typed **n** in response to the prompt, "Proceed with format (Y/N)?"

Graftabl Command Exit Codes

Graftabl exit codes

Upon completion, the **graftabl** command returns one of the following exit codes:

Code	Meaning
0	The extended character table was loaded successfully.
1	The specified table is already loaded.
2	A file error occurred.
3	You used an incorrect parameter.
4	You are using an incorrect version of MS OS/2.

Replace Command Exit Codes

Replace exit codes

Upon completion, the **replace** command returns one of the following exit codes:

Code	Meaning
0	The specified files were replaced successfully.
1	The command line you typed contains an error.
2	A file was not found.
3	A path was not found.

5	The access code for reading and writing a file is not correct for accessing the file. Try the command again using the **/r** switch.
8	There is insufficient memory to replace the files.
11	You used an invalid command-line format.
15	You specified an invalid drive.

Restore Command Exit Codes

Restore exit codes

Upon completion, the **restore** command returns the following exit codes:

Code	Meaning
0	The files were restored successfully.
1	No files were found to restore.
3	You terminated the command.
4	An error caused the command to end.

Xcopy Command Exit Codes

Xcopy exit codes

Upon completion, the **xcopy** command returns one of the following exit codes:

Code	Meaning
0	The files were copied successfully.
1	No files were found to copy.
2	You terminated the command.
4	There is not enough memory to copy the files, or you specified an invalid drive or typed an invalid command-line syntax.
5	A machine error caused the command to end while reading or writing a file to disk.

Glossary

The following terms are used in the three Microsoft Operating System/2 user's manuals:

- *Microsoft Operating System/2 Beginning User's Guide*
- *Microsoft Operating System/2 Setup Guide*
- *Microsoft Operating System/2 User's Reference*

A

Alphanumeric Refers to a name, such as a filename, which may include a combination of letters and numbers. If a list is sorted alphanumerically, numbers (0 to 9) appear first, followed by letters (A to Z).

ANSI The American National Standards Institute.

ANSI escape sequence A series of characters (beginning with an escape character or keystroke). This character set was developed by the American National Standards Institute. The ANSI character set may be loaded into protected mode using the **ansi** command. This character set may be used in real mode by installing the ANSI.SYS device driver with a **device** command in the CONFIG.SYS file.

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.

Argument Refers to switches, options, and/or variables accepted by an MS OS/2 command or other process.

B

Backup disk A copy of any disk you make with the **diskcopy**, **copy**, or **backup** command. You should always make a backup copy of the MS OS/2 program disk before you begin using MS OS/2 on a routine basis. Store the program disk in a safe place and use the copy for your work.

BASIC Beginner's All-Purpose Symbolic Instruction Code. BASIC is a general-purpose computer programming language and is often the first computer language that people learn.

Block A group of data records written in a way that is "machine-readable" (that is, understandable by your computer, modem, or other device).

Boot To start (or initialize) MS OS/2. There are two ways to boot MS OS/2. One is to turn your computer on. MS OS/2 boots automatically. If your computer is already on, you can press CONTROL+ALT+DEL. The term "reboot" generally refers to the CONTROL+ALT+DEL method.

Break To halt a process.

Buffer A work area for your computer.

Byte A unit of information used by a computer. You can use the **dir** command to see how many bytes, or characters, are in a file. The **chkdsk** command shows you how many bytes are available on a disk.

C

Central processing unit (CPU) The part of the computer that allows processes like MS OS/2 commands and application programs to run in memory.

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

Command A short program that tells MS OS/2 how to do a specific task.

Command processor An MS OS/2 program that interprets MS OS/2 commands. In protected mode, this refers to the CMD.EXE file, and in real mode, to the COMMAND.COM file, unless the user specifies alternative command-processor files.

Configuration How your personal computer system is set up. Configuration commands used in your CONFIG.SYS file help you customize the way MS OS/2 runs on your computer.

CONTROL key Used in combination with other keys to produce control characters, which affect a command MS OS/2 is executing. For example, CONTROL+C tells MS OS/2 to stop the current command.

CPU *See* Central processing unit.

Cursor Usually a blinking line or small box on the computer screen that shows where the next character you type will appear.

Data storage Magnetic hard disks and floppy disks. Your computer's memory and compact disks are also forms of data storage.

Default disk drive The drive where MS OS/2 searches for any filenames that you may type. MS OS/2 looks for files in the default drive unless you specify a different drive. The standard MS OS/2 prompt contains the default drive letter. For example, if the prompt is A>, then drive A is the default drive.

Device A piece of hardware that performs a specific function. Examples of devices include a mouse, a printer, a plotter, a network, and a modem. *See also* Device driver.

Device errors Errors that occur when one of your computer's devices, usually a disk drive or printer, is not ready or has a problem. When these errors occur, MS OS/2 displays a device error message.

Device driver A file that allows your computer to recognize a device. MS OS/2 provides seven such device drivers: ANSI.SYS, COM.SYS, EGA.SYS, EXTDSKDD.SYS, MOUSEA*xx*.SYS, POINTDD.SYS, and VDISK.SYS. *See also* Device.

Directory Part of a structure for organizing your files into convenient groups. A directory is like a file drawer that holds a particular group of files. A directory can contain both files and subdirectories. You can use the **dir** command to list the contents of a directory.

Disk *See* Floppy disk; Hard disk.

Disk buffer *See* Buffer.

Disk drive A piece of hardware attached to your computer. A disk drive can be either a floppy-disk or a hard-disk drive. You insert floppy disks into floppy-disk drives. Hard disks are usually built into the computer. See your computer manual for more information.

Disk operating system A group of programs that acts as a translator between you and your computer. MS OS/2 is a disk operating system.

Drive name Consists of a drive letter and a colon. You include a drive name in a command line to tell MS OS/2 which disk drive to search for a file. For example, the command **type a:progress.rpt** contains a drive name (**a**:) that tells MS OS/2 to look on the disk in drive A for the file called PROGRESS.RPT.

DOS 3.x compatibility mode *See* Real mode.

E

Editor A program that lets you manipulate text and data on the computer. Editors allow you to move, add, and delete characters and lines, and save files. The MS OS/2 real-mode line editor is called Edlin.

Edlin A line-oriented editor that comes with MS OS/2. Edlin works in real mode only. *See also* Editor.

ENTER key The key you usually press after typing data or text to move to a new line, or after you type an MS OS/2 command to tell MS OS/2 to execute the command.

Error messages Messages that appear on the screen if MS OS/2 detects a problem while processing a command or program.

F

FCB *See* File control block.

File A collection of related information. A file on a disk can be compared to a file folder in a file cabinet. 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.

File control block (FCB) A data structure in real mode used to control open files.

Filename A unique name for a file that can be from one to eight characters in length and may be followed by a filename extension consisting of a period (.) and one to three characters. An example of a filename with an extension is PROGRESS.RPT. Certain filenames are reserved by MS OS/2 and should not be used when naming your files. These filenames are as follows (where *x* is a number): CON, PRN, NUL, LPT*x*, COM*x*, SCREEN$, KBD$, CLOCK$. *See also* Filename extension.

Filename extension The period (.) and one to three characters that may be appended to a filename. Most application programs supply their own extensions to files they create. For example, all BASIC files use a filename extension of .BAS. *See also* Filename.

Fixed Disk *See* Hard disk.

Floppy disk A flexible, magnetic disk used for storing programs and files. You insert a floppy disk into a floppy-disk drive.

Format A way of preparing a disk so that it can hold information. Formatting a disk erases whatever informatin was previously on it. You can use the **format** command to prepare disks for use by MS OS/2.

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

Hardware The equipment that makes up a computer system, not to be confused with the programs, or software.

Initialization The starting of software. Application software is generally started (initialized) with a command. For example, the command **word** starts the Microsoft Word word-processing software package. MS OS/2 initialization is generally called a "system boot." *See also* Boot.

Input Information that is given to the computer. This information can come from the keyboard (such as when you type commands), programs, or other computers. *See also* Output.

I/O Input/output.

Memory The active part of computer storage that is used when the computer runs a program or command.

Mouse A hardware device that is moved by hand over a desktop or other flat surface, causing its corresponding pointer (cursor) to move on the screen. You press buttons located on the mouse to choose commands, select options, and work with programs.

Multitasking The ability to run more than one program at the same time. MS OS/2 is a multitasking operating system.

Operating system A group of programs that translates your commands to the computer, helping you perform such tasks as creating files, running programs, and printing documents.

MS OS/2 program disk The floppy disk (or disks) that contains the MS OS/2 system files. You should always make a backup copy of the program disk (or disks) before you start using MS OS/2 on a routine basis.

Output Information that is transferred from the computer to a device, such as a printer, disk drive, or screen. This information is produced as a result of a command or program.

P

PID *See* Process identification number.

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

Priority *See* System priority.

Process A command or program that runs on your computer.

Process identification number (PID) A number MS OS/2 appends to each process it runs. If you run a process in the background, MS OS/2 displays this number on the screen.

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

Program selector The program included with MS OS/2 that lets you switch from one session to another. *See also* Session.

Prompt The symbol that tells you an operating system or program is ready to receive a command. You type commands at the prompt. In real mode, the MS OS/2 prompt usually consists of a default drive letter (usually A, B, or C) and a greater-than sign (for example, C>). In protected mode, the current drive and directory are enclosed in brackets (for example, [C:\DOS]).

Protected mode One of two modes in which processes can run using MS OS/2. This mode allows you to run multiple programs that are "protected" from one another. If one program is using a set of data, that data is protected from corruption by any other program that is running. *See* Real mode.

R

Real mode One of two modes in which processes can run using MS OS/2. This mode is sometimes called DOS 3.*x* compatibility mode. You may run only one program or application at a time in real mode. *See also* Protected mode.

Record In a file, one unit of data. Typically, it is one line in a data file.

Root directory The directory that you are in when you first start a session (unless you specify another directory from your AUTOEXEC.BAT or STARTUP.CMD file). All other directories on your computer are subdirectories of your root directory. The root directory is represented by a backslash (\). To change your working directory to the root directory, type **cd **.

Session A work environment in MS OS/2 that contains a program you are running. In protected mode, each program usually runs in its own session. You can run only one program in the real mode session.

S

Software The programs, routines, or instructions written in a computer language that instruct the computer to perform one or more tasks. Some examples of software include operating systems, word-processing programs, and spreadsheet programs.

Storage *See* Data storage.

Subtask A task started by another task. *See also* Task.

Swap Refers to the ability to swap segments of data in and out of memory. Swapping is a time-sharing method that provides memory resources to a maximum number of processes.

Swap file The temporary file that contains the data or portion of a program that has been swapped out of memory while another process runs.

System priority A method of determining which process will run when. If a process is assigned a high system priority, it will run before another process with a low system priority.

T

Task An action your computer performs when you give it a command. Usually there is only one task associated with each program; for example, the **del** command deletes the file you specify. Some applications, however, start other tasks, called subtasks.

Thread A discrete activity within a program or other process.

Timeslice An interval of time used by the computer system.

Truncate Literally, this means "to cut off." When a copy of a file is truncated, for example, part of the file did not get copied.

V

Volume Label A computer-readable name on a disk. You should create volume labels for each of your disks to help you identify them.

W

Wildcard character A character that can be included in a filename in a command line to indicate any character or group of characters that might match that position in other filenames. In MS OS/2, you can use an asterisk (*) or a question mark (?) as wildcards. The asterisk denotes "all" and the question mark denotes "any character." For example, the filename MARKET.* refers to all files named MARKET with any filename extension. The filename ANNEX?.LST tells MS OS/2 that the question mark is a place holder for any single character. Thus, MS OS/2 equates ANNEX1.LST and ANNEX2.LST with the filename ANNEX?.LST.

Write-protected disk A floppy disk that lets you read information but not change it. These disks usually have a small tab covering a notch on the right edge of the disk. Once you remove the tab, you can change the information on the disk. If a disk does not have a write-protect notch, you cannot change any information on the disk.

Index