```
md. finance
```

If the new directory will be a subdirectory of the current directory, you need only type the part of the path that is not current.

Whenever you have a group of files that belong together (such as specialized files from the same program, files that belong to a certain person, files that are from the same project) consider storing them in their own directory. To create the directory, you use the **md** (make directory) command. For example suppose that the current directory is the root (\). To create a subdirectory to the root called WORD, you enter the following command:

```
md word
```

MS-DOS makes a new subdirectory of the root directory called \WORD. Instead of **md** you can use **mkdir** and get the same result. For example, this command does exactly the same thing as the preceding example:

```
mkdir word
```

The **md** command makes a subdirectory in the current directory unless you specify otherwise. For example, if the current directory is \WORD, to make a directory called \WORD\HOME, you need only type the last subdirectory in the path:

```
md home
```

MS-DOS makes a subdirectory in the \WORD directory called HOME. If you are not making a subdirectory of the current directory, you have to type the entire path of the new directory. For example, if the current directory is \WIN, to make \WORD\HOME, you would enter the following command:

```
md \word\home
```

If you include a drive letter in the **md** command, you can make a directory on a disk that is not the current disk. For example, suppose you wanted to copy the files in the current directory, C:\WORD\HOME to the disk in drive A. To keep the files separate, you might want to create a subdirectory on A to hold them. To create a directory on the disk in drive A without having to leave the current directory, you enter this command:

```
md a:\home
```

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

1. Display the File System screen.
2. Select the directory for which you want to create a new subdirectory.
3. Choose Create Directory from the File menu.

MS-DOS displays the Create Directory dialog box. The currently selected directory is shown as the parent directory.

4. Type the name of the new directory in the xx box.

   A name can be up to eight characters long, and you may add a three-character extension.

5. Choose OK or press ENTER.

   MS-DOS adds the new directory to the directory tree, as a subdirectory of the currently selected directory.

You can add directories at any level in the tree, but the directory you create will always be a subdirectory of the currently selected directory.

Moving Between Directories

The disks in each drive of your system have a current directory. If you have two floppy disks and one hard disk, there are three current directories.

If a disk has no subdirectories, the root directory is always the current directory for that disk. If a disk has subdirectories, you can use the **cd** (change directory) command to move from the one directory to another.

Finding Out Which Directory Is Current

In Brief

To display the current directory on the current drive, enter a **cd** command with no parameters as in the following command:

```
cd
```

If the command prompt is set up to show you the path of the current directory on the current drive, you need only look at the prompt to see which directory is current. If the prompt does not show the current directory or if you want to see which directory is current on another drive, you can use the **cd** command. For example, the following command displays the path of the current directory on the current drive:

```
cd
```

The following command displays the current directory on drive B:

```
cd b:
```

It does not make drive B current.

Changing Directories

In Brief

To move to a different directory on the current disk, use the **cd** command (also called **chdir**). For example, the following command changes the current directory to \WIN\EXCEL\FINANCE:

```
cd \win\excel\finance
```

Use the path ".." to change to the parent directory current as in the following command:

```
cd ..
```

Suppose that the current directory is the root (\). You can move to the \WORD directory and make it current by entering the following command:

```
cd word
```

Instead of **cd** you can use **chdir** and get the same result. For example, this command is the same as the previous example:

```
chdir word
```

As in the **md** command, unless you specify otherwise, MS-DOS assumes you want to change to a subdirectory of the current directory. For example, if the current directory is \WORD, to change to \WORD\HOME, you need only type the last subdirectory in the path:

```
cd home
```

If the current directory is \WIN, to change to \WORD\HOME, you have to enter the entire path:

```
cd \word\home
```

To change to the parent of the current directory (the directory one level closer to the root) you can use the ".." combination. For example, if the current directory is \WORD\HOME, you can change to \WORD with this command:

```
cd ..
```

No matter which directory is current, you can change to the root directory of the current drive by entering this command:

```
cd \
```

You can't use the **cd** command to change the current drive, but you can use it to change the current directory on a drive that is not current. For example, suppose the current drive is A. To change the current directory on drive C to \WORD *without making drive C current,* use the following command:

```
cd c:\word
```

Now, when you type C: in a command without a directory, MS-DOS assumes you want to use the \WORD directory. For example, if A is the current drive and \WORD is the current directory on drive C, the following command copies all the files from the current directory on drive A to C:\WORD:

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

The current directory on a drive is the root unless you change it.

Shell In the Shell, the files in the currently selected directory are displayed on the right side of the File System screen. The name of the currently selected directory appears in the directory indicator just above the drive indicator in the upper-left corner of the screen. In the directory tree, the currently selected directory is highlighted.

For information about selecting directories, see "XX".

Deleting Directories

In Brief

To delete a directory, use the **rd** command (also called **rmdir**), as in the following example:

```
rd \win\excel\finance
```

MS-DOS removes the FINANCE subdirectory from the \WIN\EXCEL\ directory. The directory you remove must contain no files or subdirectories.

If you no longer need a directory, you can remove it with the **rd** command. Before you remove a directory, it must be empty and it must not be the current directory. It cannot contain any files or subdirectories. If you display the contents of an empty directory with a **dir** command, it should have only two items listed: the period (..), which stands for the directory itself, and the double period (.), which represents the parent directory.

If a directory is empty, you can enter a **rd** command to delete it. For example, suppose you have a directory called \WORD. Before you can remove the directory, you must delete its contents with the following command:

```
del \word\*.*
```

You can then enter the following command to remove the directory:

```
rd word
```

MS-DOS removes \WORD from the current drive. If \WORD has subdirectories, you must remove them before you can remove \WORD.

Instead of **rd** you can use **rmdir** and get the same result. For example, this command is the same as the preceding example:

```
rmdir word
```

As with the **md** and **cd** commands, MS-DOS assumes you want to remove a subdirectory of the current directory. For example, if the current directory is \WORD, to remove \WORD\HOME you need only type the last subdirectory in the path:

```
rd home
```

If the current directory is \WIN, to remove \WORD\HOME, you must enter the entire path:

```
rd \word\home
```

If you include a drive letter in the **rd** command, you can remove a directory from a disk that is not the current disk. For example, to remove the A:\HOME directory while the current drive is C, use the following command:

```
rd a:\home
```

Shell ► **To delete a directory or subdirectory:**

1. Make sure the directory does not contain any files or subdirectories.
2. Select the directory.
3. Choose Delete from the File menu.
   MS-DOS displays the Delete Directory dialog box.
4. Choose option 2 to delete the directory. Choose option 1, or Cancel, if you don't want to delete the directory.
5. Choose OK or press ENTER.

If you try to delete a directory that contains files or subdirectories, the shell displays the Delete Directory dialog box with an "Access denied" message. If this message appears, delete the files and subdirectories in the directory and try again.

Copying Directories

To copy directories and their subdirectories, you use the **xcopy** command. The **xcopy** command is similar to the **copy** command. Both commands copy files from one directory or disk to another. However, while the **copy** command is designed to work with single files or groups of files, the **xcopy** command is designed to work with single directories or groups of directories. Both commands create new files in the "destination" directory, but only the **xcopy** command creates new subdirectories as well.

Copying All Files in a Directory

In Brief

To copy single directories (without subdirectories), use the **xcopy** command with no switches. For example, the following command copies all files in the C:\WIN\EXCEL\FINANCE directory to the \FINANCE directory on drive A:

```
xcopy c:\win\excel\finance a:\finance
```

If you want to copy all the files in a directory to another existing directory, the **copy** command is equivalent to the **xcopy** command. Unlike the following **copy** command which uses wildcards to copy all of the files from drive A to drive B:

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

Because the **xcopy** command assumes you want to copy whole directories, you don't need to use wildcards. For example, the following **xcopy** command performs exactly the same function as the preceding command:

```
xcopy a: b:
```

Before MS-DOS copies the files, it displays a "Reading source file(s)..." message while it prepares to copy the files. As in the **copy** command, MS-DOS displays the names of the files it copies and tells you how many files were copied when the operation is complete.

Shell ▶ **To copy all of the files in a directory to another existing directory:**

1. Select the disk drive or directory containing the files you want to copy.
2. Choose Select All from the File menu to select all the files in the directory.

   If you don't want to copy all of the files, you can cancel the selection of individual files by holding down the CTRL key and clicking them, or by pressing SHIFT+F8, moving the selection cursor to the files you want to deselect, and pressing CTRL+SPACEBAR. You can also use the Deselect All command on the File menu to deselect all the files and start again.

3. Choose Copy from the File menu to copy the files.

   MS-DOS displays the Copy dialog box.

4. Type the path of the directory you want to copy to in the To box. If the directory is on a different drive, include the drive name.

5. Choose OK or press ENTER.

For more information about paths and drive names, see "XX".

Creating Directories as You Copy

In Brief

If the destination path in an **xcopy** command does not exist, MS-DOS creates it. For example, the following command copies all files from the root directory of drive A to the C:\ATMP directory:

```
xcopy a: c:\atmp
```

If the directory does not exist, MS-DOS asks you if you wan to create it (to stop MS-DOS from prompting you by adding a backslash (\) to the end of the directory name.

You can use the **xcopy** command to create directories as you copy files. For example, suppose you want to copy all the files on the disk in drive A to drive C. Using the following **xcopy** command, you can tell MS-DOS to put the files in a directory called \NEWFILES:

```
xcopy a: c:\newfile
```

If the \NEWFILE directory does not already exist on drive C, MS-DOS creates it as a subdirectory of the root directory. Then, MS-DOS copies the files from drive A to the \NEWFILE directory. In this example, only the files in the root directory of drive A are copied. If there are subdirectories on drive A, MS-DOS does not copy them.

If you do not type a path, MS-DOS copies the files to the current directory.

Shell

To duplicate a directory that has several subdirectories, first use the Create Directory command on the File menu to create the same directory structure in the new location. Then copy the files from each of the source directories to each of the destination directories.

Copying Subdirectories

In Brief

To reproduce a directory structure completely in another directory or on another disk, use the /s and the /e switches. For example, the following command recreates the directory structure and files within C:\WIN on drive B:

```
xcopy c:\win b:\ /s /e
```

To tell MS-DOS to copy the files in a directory along with any subdirectories that have files, you add the /s switch to the **xcopy** command. For example, suppose the disk in drive A has the following directories on it: \SCHOOL, \WORK, and \HOME. The following command copies the files in the root directory of A, the three subdirectories, and all their files to the directory on drive C called \MEMOS:

```
xcopy a:\ c:\memos /s
```

The backslash (\) after the A: tells MS-DOS to start at the root directory. The /s tells MS-DOS to copy every file in every subdirectory that has files. MS-DOS copies files from the root directory of drive A to C:\MEMOS, from A:\SCHOOL to C:\MEMOS\SCHOOL, from A:\WORK to C:\MEMOS\WORK, and from A:\HOME to C:\MEMOS\HOME. If any of the directories do not exist on drive C, MS-DOS creates them. In this example, empty subdirectories on drive A are not copied.

To copy empty directories add a /e switch in addition to the /s switch. For example, if the disk in drive A has an empty subdirectory called \MISC, in addition to the three subdirectories mentioned above, MS-DOS would not have copied it. To include the empty directory, enter this command:

```
xcopy a:\ c:\memos /s /e
```

MS-DOS copies the same files as before, but now also creates an empty directory called C:\MEMOS\MISC. You can use the /s switch with or without the /e switch. However, if you do use the /e switch you must also include the /s switch.

Renaming Directories

In Brief

To rename a directory, use the **xcopy**, **del**, and **rd** commands. For example, the following sequence of commands rename the \OPS\STATS directory to \OPS\FIGURES:

```
xcopy \ops\stats \ops\figures

del \ops\stats\*.*
```

```
rd \ops\stats
```

If the original directory has subdirectories, add the /s and /e switch to the **xcopy** command, and delete each subdirectory separately.

▶ **To rename a directory:**

1. Copy the contents of the directory to a directory with the new name.
2. Delete the contents of the original directory
3. Delete the original directory.

For example, suppose you want to rename the C:\TEMP directory to C:\LETTERS. The first step is to copy the contents of the directory to a directory with the new name. You can accomplish this most easily with this **xcopy** command:

```
xcopy c:\temp c:\letters
```

MS-DOS creates a subdirectory of the root directory of drive C called LETTERS, and copies all the files from C:\TEMP into it.

The second step is to delete the files in C:\TEMP. To do this, use the following **del** command:

```
del c:\temp\*.*
```

MS-DOS asks you if you really want to delete all of the files in this directory. If you are sure the files were successfully copied to C:\LETTERS, enter **y**. If you have any doubt, enter **n** and use the **dir** command to double-check.

Once the C:\TEMP directory is empty, the final step is to delete the directory itself by entering the following **rd** command:

```
rd c:\temp
```

MS-DOS removes the old directory. As a result of this three-step process, the C:\TEMP directory has been renamed to C:\LETTERS.

Shell ▶ **To change the name of a directory or subdirectory:**

1. Choose Deselect All from the File menu if any files are selected.
2. Select the directory you want to rename.
3. Choose Rename from the File menu.
   A dialog box appears.

4. Type the new name for the directory in the box.
5. Choose OK or press ENTER.

   The directory name changes.

Updating Directories

Sometimes you will have two directories that should contain the same files. For example, when you are making backups, you will have a primary directory that contains the files you are working on and a secondary directory or disk that contains the backup or most recent version of the files. To keep the secondary directory current, use the **replace** command.

Replacing Outdated Files

In Brief

To replace the files in a destination directory that are older than the corresponding files in a source directory use the **replace** command with a **/u** switch, as in the following command:

```
replace c:\win\*.* a: /u
```

MS-DOS replaces the files in A:\ that have more recent versions in C:\WIN.

Suppose you have a directory called C:\OPS\STATS containing statistics files you periodically update. To keep a backup of these reports you can copy them to a floppy disk and periodically update them with this command:

```
replace c:\ops\stats\*.* a: /u
```

MS-DOS compares the files in C:\OPS\STATS with the files on the disk in drive A. If a file on drive A has a more recent version on drive C, MS-DOS replaces the older version. This form of the command does not add any new files to the backup disk; it only updates those that are already there.

Adding New Files

In Brief

To add files to the destination directory that are currently only in the source directory, use the **replace** command with an **/a** switch. For example, the following command compares the files in C:\WIN with those in A:\ and copies any that exist in C:\WIN, but not in the A:\ directory:

```
replace c:\win\*.* a: /a
```

In the example in the previous section, you used the **replace** command with the /u switch to update the files on the backup disk in drive A. To add files to the backup disk, you use the /a switch rather than the /u switch. For example, the following command compares the files in C:\OPS\STATS with the files on the disk in drive A:

```
replace c:\ops\stats\*.* a: /a
```

If there are any files in the C:\OPS\STATS directory that aren't on the disk in drive A, MS-DOS copies them over.

Telling MS-DOS Where to Look for Files

When you want to run a program, you enter its filename. In order to run the program, MS-DOS must first locate it. Likewise, when you type a filename as part of a command, or when you try to get to a file from inside a program, MS-DOS must locate the file. Unless you tell it otherwise, MS-DOS only looks for files in the current directory.

There are two commands you can use to tell MS-DOS which other directories to search for files:

- The **path** command tells MS-DOS which other directories to search for program files and commands.
- The **append** command tells MS-DOS to search additional directories for all other kinds of files.

Using the Path Command

In Brief

To have MS-DOS search specific directories for program or batch files, use the **path** command. For example, the following command tells MS-DOS to look in the three directories listed in addition to the current directory when it searches for program and batch files:

```
path \;c:\bin;c:\utilities
```

Each directory in the **path** command is separated by a semicolon (;). The first backslash (\) tells MS-DOS to search the root directory of the current drive.

One way to run a program stored in a directory that is not current is to enter its path and filename. Another way is to first make its directory current with the **cd** command. To avoid the extra work involved with either if these procedures, you can type all the paths you commonly use once in a **path** command. Then, if MS-DOS cannot find the file you type in the current directory, it searches the other directories you specified. The **path** command you enter remains active until you restart or reset your system.

For example, suppose you commonly run programs stored in the C:\PBRUSH directory, the C:\WORD directory, and the C:\EXCEL directory. You can save having to type these paths each time you want to use them by entering them once in the following **path** command:

```
path c:\word;c:\excel;c:\pbrush
```

Each directory is separated from the others by a semicolon (;). MS-DOS searches the directories in the order you type them (in this case, MS-DOS would search \WORD before \EXCEL). To have MS-DOS look first in the root directory of the current drive, you add it to the beginning of the list:

```
path \;c:\word;c:\excel;c:\pbrush
```

You can tell MS-DOS to search as many directories as you care to list in the **path** command as long as the command doesn't exceed 127 characters (not including the word *path*).

To set the search directories automatically each time you reset or restart the system, you can include a **path** command in your AUTOEXEC.BAT file. If you find that the search directories have already been set, it means that there is already a **path** command in your AUTOEXEC.BAT file. For more information, about AUTOEXEC.BAT, see xx.

Using the Append Command

In Brief

To have MS-DOS search additional directories for text or data files, use the **append** command. For example, the following command tells MS-DOS to look in the C:\WORD\NOTES directory in addition to the current directory for files that are included in commands or are requested by programs:

```
append c:\word\notes
```

The **append** command is similar to the **path** command except that it applies to non-program files. The **append** command is useful when you have directories that contain files you often use in commands.

For example, suppose you commonly use the files stored in the C:\PBRUSH\PICS directory, the C:\WORD\DOCS directory, and the C:\EXCEL\SHEETS directory. You can save having to type these paths each time you want to use them by entering them once in the following **append** command:

```
append c:\word\docs;c:\excel\sheets;c:\pbrush\pics
```

As in the **path** command, each directory is separated from the others by a semicolon (;). MS-DOS searches the directories in the order you type them (in this case, \WORD\DOCS would be searched before \EXCEL\SHEETS).

Be careful that you don't end up using the wrong file. For example, suppose there is a file called LIST.TXT in both the C:\WORD\DOCS directory and the C:\EXCEL\SHEETS directory. After entering the append command in the preceding example, you might be tempted to delete the C:\EXCEL\SHEETS\LIST.TXT file by simply entering:

```
del list.txt
```

If you did, MS-DOS would delete C:\WORD\DOCS\LIST.TXT, not C:\EXCEL\SHEETS\LIST.TXT as you had intended.

To set the append path automatically each time you start the system, you can include an **append** command in your AUTOEXEC.BAT file. If you find that an append path has already been set, it means that there is already an **append** command in your AUTOEXEC.BAT file. For more information, see xx.

Chapter 7

Managing Disks

7

Disks store information permanently. In contrast to your system's random access memory (RAM), which is lost when you turn the power off, the information on a disk remains intact until you erase it. In addition, floppy disks provide a convenient method for transferring information. Almost all the programs you run come to you on floppy disks, and when you want to transfer files to other systems, you are likely to use floppy disks.

Types of Disks

The disks your system uses store information on magnetic surfaces. Floppy and hard disks get their names from the kind of magnetic surface they have. In floppy disks the magnetic surface is a thin flexible disk inside a protective plastic cover. Hard disks have two or more rigid disks stacked on top of each other in a sealed case. Hard disks are also called fixed disks because they remain in your system.

The information on all disks is divided into tracks somewhat like the grooves on a record. Each track is a concentric circle that can hold a certain amount of information. The more tracks a disk has, the more information it can hold. Floppy disks with the fewest tracks per side are called single sided or low density. Floppy disks with more tracks are called double sided, double density, quad density, or high density, depending on the number of tracks they have.

This artwork is not available
for this release.

dsk_1

Both hard disks and floppy disks have tracks. Hard disks hold more information than floppies because they have more sides and more tracks per side.

Floppy disks vary in physical size and the amount of information they can hold. The following table lists the major kinds of floppy disks that MS-DOS can work with and the amount of information they can hold:

- 5.25-inch, single sided/double density, 160K
- 5.25-inch, single sided/double density, 180K
- 5.25-inch, double sided/double density, 320K
- 5.25-inch, double sided/double density, 360K
- 5.25-inch, double sided/quad density, 1200K (1.2 MB)

- 3.5-inch, double sided/double density, 720K
- 3.5-inch, double sided/quad density, 1440K (1.44 MB)

Most floppy disks have labels that tell you which kind they are. However, sometimes the only way to tell one type of disk from another is to use a **dir** or **chkdsk** command to display information about the storage capacity of a formatted disk.

Bytes, Kilobytes, and Megabytes

Your files are measured in *bytes*. One byte is the amount of space it takes to store a single letter. A kilobyte is 1024 bytes. In this guide, the word *kilobyte* is abbreviated K.

A megabyte is 1024K (about a million bytes). In this guide, the word *megabyte* is abbreviated MB. For example, if a disk can store about 1.2 million bytes of information, it is a 1.2 MB disk. The following terms are equivalent:

```
1.2 MB = 1228.8K = 1,258,291 bytes
```

Kinds of Disk Drives

Hard disks are indistinguishable from hard disk drives because the two are a single unit. Once your hard disk is installed, you should never have to remove it. MS-DOS adjusts its operations to suit the hard drive you use.

On the other hand, for every type of floppy disk there is a floppy disk drive that is designed specifically to work with it. Not all types of floppy disks are compatible with all types of floppy disk drives. In general, the capacity of the disk must be less than or equal to the capacity of the drive in order for them to be compatible. For example, if you have a quad density, 5.25-inch disk drive that is designed to work with 1.2 MB floppy disks, you can use 360K double-density floppy disks with it. However, if you have a 360K drive, you cannot normally use 1.2 MB disks with it. If you are unsure whether a disk will work with a drive, try it. MS-DOS will display a "General failure error" message if the drive and disk are incompatible.

MS-DOS automatically adjusts its operations to suit the disk drive you are using. In some commands, you must type an additional switch if your disk drive and disk don't have the same capacity.

Formatting Disks

MS-DOS must format disks before using them. To format a floppy disk, MS-DOS divides the disk into smaller segments called *sectors* and sets up a file tracking system. Hard disks are already divided into sectors by the time you are ready to format them, so MS-DOS ensures that the disk is usable and set up the file tracking table. If you are using a new hard disk, however, you need to partition it before you can format it. You can use the MS-DOS installation program to partition and format the hard disk automatically when you install MS-DOS. For more information about installing MS-DOS on a hard disk, see "XX". You can also partition a new hard disk by using the **fdisk** command. For more information on partitioning a hard disk with **fdisk**, see "XX".

A sector is the basic unit of storage on a disk. Each sector on a hard disk or floppy holds one half of a kilobyte of information. When MS-DOS formats a disk it checks each sector for defects. MS-DOS marks "bad" sectors so that it will not store information there. When MS-DOS stores a file on the disk it uses groups of sectors called *allocation units*. The number of sectors depends on the size of the disk.

When you format a floppy or hard disk, MS-DOS reserves a small part of the disk for its tracking system. The allocation system consists of two parts: a file allocation table (which tracks the location of each file on the disk) and the root directory (which stores the name, size, creation date and time, and file attributes for the files on the disk).

One of the reasons you can't get information back from a newly formatted disk is because MS-DOS destroys any existing file table to create a new one. With floppy disks there is another reason. As MS-DOS divides a floppy disk into sectors, it destroys the contents of any sectors that were already there. When it formats a hard disk, the old file table is destroyed, but any existing sectors remain intact. The old sectors are filled as you copy new information to the disk.

During the formatting process, you can copy the three main MS-DOS system files to the disk. This makes the disk a system disk that you can use to start your system. For more information about system disks, see "xx" later in this chapter.

CAUTION The **format** command destroys all information on a floppy or hard disk. You should develop the habit of using the **dir** command before formatting your floppy disks or you might accidentally destroy important files. The **format** command is used to prepare your hard disk, so it is possible to format a hard disk drive if you accidentally enter the wrong drive letter. As a safety measure, MS-DOS displays a warning message if you attempt to format your hard disk. If you accidentally format your hard disk you may be able to use the

unformat command to recover its contents. For more information, see "xx" later in this chapter.

Formatting a Single Disk

In Brief

To format a floppy or hard disk for use with MS-DOS, use the **format** command. For example, the following command formats a floppy disk in drive A:

```
format a:
```

You must specify the drive that contains the disk you want to format.

You format disks using the **format** command. For example, the following command formats the floppy disk in drive B:

```
format b:
```

MS-DOS requires you to specify the drive letter when you use the **format** command.

As it formats the disk, MS-DOS displays a message that tells you what percentage of the disk has been formatted. When MS-DOS is finished, it asks you if you want to give the disk a *volume label.* You can enter a label to describe the disk or press ENTER if you do not want a label. If you give the disk a volume label, be sure you also write it on the outside cover of the disk to help identify it.

After MS-DOS asks you for a label, it displays the following information that describes how the disk was formatted:

This artwork is not available
for this release.

dsk_2

The "bytes total disk space" line tells you how much storage the disk contains. The number should be as much as or more than the capacity you expect for the disk. In the preceding example the total disk space is xx, a little more than the 1.2 MB the disk is designed for.

The "bytes used by system" line appears if you have transferred the MS-DOS system files to the disk. This line shows how much disk space is used by the three system files.

The "bytes in bad sectors" line tells you how much of the disk is unusable because of bad sectors. If there are no bad sectors, this line is omitted. The bad sectors that MS-DOS finds during the format will not be a problem because they have been marked. However, if a floppy disk has any bad sectors, it may be indicative of a growing problem with the disk. You should consider not using a floppy disk with bad sectors to store important files or backups. Most hard disks have a small number of bad sectors. In general, the portion of a hard disk taken up by bad sectors should be a very small fraction of the total space available on the hard disk. The exact permissable amount depends on the manufacturer of your hard disk.

The "bytes available on disk" line is the total disk space minus the space taken up the system and any bad sectors. If the disk does not contain the system files and there are no bad sectors, this number should match the total disk space number.

The "bytes in each allocation unit" line and the "allocation units available on disk" line show you the way MS-DOS has divided the available disk for file storage. If you multiply the two numbers on these lines the result will equal the number on the "bytes available on disk" line.

The "Volume serial number is" line shows you the serial number that MS-DOS has assigned to the disk. This number will not change unless the disk is formatted again.

On the last line of the display, MS-DOS asks you if you want to format another disk. Enter y to format another disk in the same drive with the same options. Enter n to return to the command prompt.

Formatting Different Disk Types

In Brief

To format a floppy disk that has a lower storage capacity than your disk drive, use the /f: switch. For example, if drive A is a high density disk drive, the following command allows it to format a 360K disk:

```
format /f:360
```

There are a number of different kinds of floppy disks, each of which MS-DOS must format differently. Unless you state otherwise, MS-DOS assumes that the disk you want to format has the maximum capacity for the drive. For example, if you have a high-density (1.2 MB) 5.25-inch disk drive, MS-DOS assumes that the disks you format in the drive are also high density. To tell MS-DOS that you want to format a disk with a lower capacity, you use the /f: switch.

For example, if drive A is a 1.2 MB, 5.25-inch drive and you want to format a 360K disk in it, you use the following command:

```
format a: /f:360
```

If drive B is a 1.44 MB, 3.5-inch drive and you want to format a 720K disk in it, you use this command:

```
format b: /f:720
```

NOTE Because there are differences in hardware between disk drives, some 360K drives cannot reliably read disks formatted on a 1.2 MB drive with the **/f:360** switch.

Creating System Disks

In Brief

To create a system disk that can start your system, use the **format** command or the sys command.

To create the'system disk during the format operation, add the /s switch to the **format** command. For example, the following command copies the system files to the disk in drive A after it is formatted:

```
format a: /s
```

To create a system disk from a disk that is already formatted, use the sys command. For example, the following command makes the disk in drive B a system disk:

```
sys b:
```

System disks contain the MS-DOS program. They are the kind of disks you can use to start your system.

System disks are just like other disks except that they contain the three main MS-DOS system files: IO.SYS, MSDOS.SYS, and COMMAND.COM. When you switch on your system, these three files are moved from the system disk to your system's random access memory (RAM). The IO.SYS and MSDOS.SYS files are hidden files; you do not see them in directories. The COMMAND.COM file is usually in the root directory of every system disk. The other command and system files that MS-DOS uses need not be present on a system disk. If you have a hard disk, it is usually set up to be your main system disk. However, when you switch on the system, it always checks drive A first to see if you want to start the system from a system disk there. That is why you see an error message if there is a nonsystem disk in drive A when you switch on your system.

You can copy the three system files to a disk as part of the **format** command or with the sys command.

To make a disk a system disk when you format it, you include the /s switch in the **format** command. After the disk has been formatted, MS-DOS copies the three files to the disk. For example, the following command formats the disk in drive B and makes it a system disk:

```
format b: /s
```

To make a disk a system disk after it has been formatted, you use the sys command. For example, the following command makes the disk in drive A a system disk:

```
sys a:
```

After you run this command, the disk has two of the three files you need. Before you start MS-DOS from the disk, you must copy the COMMAND.COM file to the disk.

When a floppy disk has the three system files on it, you can use it to start your system by putting it in drive A and switching on the system. When your hard disk has the system files on it, you can use it to start the system by making sure no other disk is latched into drive A when you switch on your system.

Unformatting a Hard Disk

In Brief

To recover as much information as possible from a hard disk that has been reformatted, use the **unformat** command as in the following example:

```
unformat c:
```

The **unformat** command will not restore floppy disks.

After you first set up your hard disk, it should rarely, if ever, need to be formatted. If you accidentally format your hard disk, you may be able to recover all of the information. If you run the **unformat** command before you store any files on the disk, MS-DOS will restore the disk to its condition before the format. If you have stored any files on the disk since you formatted it, MS-DOS will not be able to restore the entire disk. The amount of the disk MS-DOS can restore depends on where the new file is stored on the disk.

Unless you tell it otherwise, MS-DOS performs a "safe" format on your hard disk. The safe format allows you to use the **unformat** command. To disable safe formatting you include the /u switch with your **format** command.

To unformat the hard disk in drive C, use the following command:

```
unformat c:
```

If MS-DOS cannot restore all the information on the disk, it displays a message asking you if you want to continue.

Using the Shell to Format Floppy Disks

When you format a floppy disk from the shell you are running the MS-DOS **format** command from the Program Manager screen. Therefore you use the same switches and some of the same procedures to format a disk from the shell as you would from the command line.

Switches allow you to format a floppy disk with a smaller storage capacity than your floppy disk drive and transfer the MS-DOS operating system to a disk. For more information about the switches you can use with the **format** command see the previous xx sections.

Shell ▶ **To format a floppy disk:**

1. Choose Format from the DOS Utilities group on the Program Manager screen.

   MS-DOS displays the Format Utility dialog box.

2. To format drive A and use no switches, Choose OK. To format a different drive or change the way the command functions, type a new drive letter and switches in the xx box and then choose OK.

   From this point on, MS-DOS displays the same messages and prompts in the Shell as it does at the command prompt. For more information about messages MS-DOS displays, see xx.

3. After the formatting is complete, to give the disk a volume label, type a name at the volume label prompt. If you don't want to give the disk a label, press ENTER at the prompt.

4. At the next prompt, enter y if you want to format another disk or enter n if you want to return to the MS-DOS Shell.

   MS-DOS returns you to the Program Manager screen when you press any key.

Labeling Disks

Each disk you use, including a hard disk, can have a name and a number. The name is called the volume label. You can use it to identify the disk electronically. The number is called the volume serial number. MS-DOS uses the volume serial number to keep track of which disk is in a drive. MS-DOS assigns a serial number to the disk when you format it. The serial number will not change unless the disk is formatted again. Only disks formatted by MS-DOS versions later than 4.0 have a serial number.

MS-DOS displays the disk's volume label and serial number at the top of every directory.

You can change a disk's volume label with the **label** command. The volume labels you choose must be less than 12 characters long and cannot include the following characters: asterisk (*), question mark (?), slash (/), backslash (\), pipe (|), period (.), comma (,), colon (:), semicolon (;), plus sign (+), equal sign (=), double quotation mark ("), square brackets ([ ]) ampersand (&), and any control characters. Volume labels can include spaces but not tabs.

Assigning and Deleting Labels

In Brief

To assign a volume label to a disk, use the **label** command. For example, the following command gives the disk in drive A the label "backup disk 1":

```
label a:disk 1
```

The label can be no longer than 12 characters including spaces. There should be no spaces between the drive name and the label. If you enter the **label** command without parameters, MS-DOS prompts you for the drive and label you want.

If you work with a large number of disks, you may find it convenient to create a label for the disk that you can see when you use the **dir** or **vol** commands. This electronic label is called a volume label.

To assign a volume label, you use the **label** command. For example, the following command assigns the label "BERNIE'S 2" to the disk in drive A:

```
label a:bernie's 2
```

There must be no spaces between the colon of the drive letter and the label. If you type a drive letter, but no label, MS-DOS prompts you for a label. For example, to label the disk in drive B, you would enter the following command:

```
label b:
```

MS-DOS displays the current label and serial number for the disk in drive B and then prompts you to enter a new volume label.

To delete a volume label, enter the **label** command without a label name. Then, when MS-DOS prompts you to enter a new volume label, press ENTER. If you are giving a new volume label to a hard disk, MS-DOS asks you to confirm that you want to delete the volume label. Enter **y** to delete the label or **n** to exit from the command without changing the current label.

Viewing Labels

In Brief

To display the volume label and serial number of a disk, use the **vol** command. The following command displays this information for the disk in the current drive:

```
vol
```

To view a disk's volume label and serial number, you use the **dir** command or the **vol** command. These commands can help you identify the disk you are using. With the **dir** command, the volume label and serial number for the disk that you specify are displayed above the list of files.

The **vol** command displays the volume label and serial number of the disk in the drive you specify. For example, the following command displays the volume label and serial number of the disk in drive A:

```
vol a:
```

Making Backup Disks

Because both floppy and hard disks can fail occasionally, and just in case you need a file you deleted, it is wise to back up your files on a regular basis. MS-DOS gives you many ways to make backup copies of files. If you want to make a backup copy of a few files, the simplest way is to use the **copy** or **xcopy** command to save the files on other disks. If you have a lot of files to back up, or if you want to automate the process of backing up files, you can use the **backup** command to:

- Back up single directories.
- Back up directories and their subdirectories.
- Back up selected files.
- Add files to a backup disk that you previously created.

Because the **backup** command uses whole disks, be sure you have enough disks to hold all the files you want to back up. MS-DOS deletes the existing files on the disks you use, so choose a disk for backups that you don't need for any other purpose.

You cannot directly access the files that MS-DOS creates on backup disks. If you need to retrieve the files you backed up, you must use the **restore** command. The **restore** command reads the backup disk and puts the files you specify back where they came from. The **restore** command is described in "xx".

Backing Up Single Directories

In Brief

To create and maintain backup disks, use the **backup** command. For example, the following command makes a backup of the files in the C:\WIN directory on drive A:

```
backup c:\win a:
```

The simplest form of the **backup** command backs up single directories. For example, the following command backs up the files in C:\WORD\HOME to the disk in drive B:

```
backup c:\word\home b:
```

MS-DOS displays the following prompt to tell you to insert your backup disk and to remind you that it erases the existing files before creating the backup files.

```
Insert backup disk 01 in drive B:
 WARNING! Files in the target drive
 B:\ root directory will be erased
 Press any key to continue . . .
```

When you press a key, MS-DOS begins copying files from C:\WORD\HOME. Files in subdirectories of this directory are not copied. To cancel the command without erasing any files from the target disk, press CTRL+C.

MS-DOS creates two files on drive B: BACKUP.001 and CONTROL.001. It combines all the files in C:\WORD\HOME and stores them in BACKUP.001. It stores the pathnames of the files in CONTROL.001. MS-DOS also changes the volume label of the disk to BACKUP 001.

If MS-DOS needs more than one disk to back up your files, it prompts you to insert another disk in drive B. The second disk has the files BACKUP.002 and CONTROL.002 (if there is a third disk the files are BACKUP.003 and CONTROL.003, and so on). Usually, if you don't specify a drive, MS-DOS assumes you want to use the current drive. However, with the **backup** command, you must always type at least a drive letter for both the source and destination directories. You do not have to type the pathname of the current directory. For example, if C:\WORD\HOME is the current directory, the following command is equivalent to the preceding example:

```
backup c: b:
```

Backing Up a Directory and its Subdirectories

In Brief

To back up a directory and its subdirectories, include the /s switch with the **backup** command. For example, the following command backs up every file in every directory on drive C:

```
backup c:\ a: /s
```

You can save time by backing up a directory and all its subdirectories with one command. To include subdirectories in your backup, use the /s switch. For example, to back up C:\WORD\HOME and all of its subdirectories, enter the following command:

```
backup c:\word\home b: /s
```

MS-DOS copies the files from C:\WORD\HOME and all of its subdirectories into the BACKUP.001 file on drive B. The directory structure of the files is preserved in the CONTROL.001 file.

To back up all the files on drive C enter this command:

```
backup c:\ b: /s
```

MS-DOS starts at the root directory and backs up all the files on drive C. MS-DOS prompts you to insert new disks as it needs them.

Backing Up Selected Files

In Brief

To back up selected files from a directory, use the MS-DOS wildcards as in the following command:

```
backup c:\win\*.xls a:
```

It is not always necessary to back up all the files in a directory. Sometimes you need to back up only files of a certain type, or files that have changed since the last backup.

To back up a single file, specify its filename after its drive letter and path. For example, the following command backs up the OUTGO.XLS file in the C:\WORD\HOME\ directory:

```
backup c:\word\home\outgo.xls b:
```

To back up files of a certain type, you can use the MS-DOS wildcards. For example, the following command backs up only the files in C:\WORD\HOME that have a .DOC extension:

```
backup c:\word\home\*.doc b:
```

Adding Files to a Backup Disk

In Brief

To copy files to a backup disk without first deleting the files that are already there, use the /a switch with the backup command as in the following example:

```
backup c:\win a: /a
```

To add only files that have later versions than currently backed up file, or that don't already exist on the backup disk, include the **/a** and **/m** switches as in this command:

```
backup c:\win a: /a /m
```

In the preceding section, the **backup** command erased all existing files on the backup disk. To copy files to a backup disk without erasing the ones that are already there, use the **/a** switch. For example, the following command adds the files from C:\WORD\SCHOOL to those already on the disk in drive A:

```
backup c:\word\school a: /a
```

If some of the files from the WORD\SCHOOL directory are already on the backup disk, MS-DOS overwrites them with the files from drive C. When the operation is complete, the backup disk has the files it originally had plus the most current version of the files in C:\WORD\SCHOOL.

If you want to back up only files that have been added or changed since the last time you backed up a directory, use the **/m** switch with the **/a** switch.

For example, suppose that after you backed up the C:\WORD\SCHOOL directory, you added three new files and changed two that were backed up. To back up the files that are new or have changed, you would put the original backup disk in drive A and enter the following command:

```
backup c:\word\school a: /a /m
```

MS-DOS adds the new files to the backup disk and replaces the ones that have changed with the newer version.

NOTE If you use the **/m** switch without the **/a** switch, MS-DOS erases the existing files on the backup disk and copies only those that have changed since the last backup.

Using the Shell to Make Backup Disks

When you back up files from the Shell you are running the MS-DOS **backup** command from the Program Manager screen. Therefore, you use the same switches and wild cards and some of the same procedures to back up files from the Shell as you would from the command line.

Switches and wildcards allow you to back up subdirectories and selected files, and to add files to an existing backup disk. For more information about the switches and wildcards you can use with the **backup** command see the previous xx sections.

You can use the Backup Fixed Disk and Restore Fixed Disk commands only if you have the Shell installed on a hard disk.

Shell ▶ **To backup files:**

1. Choose Backup from the DOS Utilities group on the Program Manager screen.
   MS-DOS displays the Backup Utility dialog box.
2. To back up your entire hard disk, choose OK. To back up a subdirectory or selected files, or to add files to an existing disk, type the appropriate switches and wildcards in the Parameters box and then choose OK.
   From this point on, MS-DOS displays the same messages and prompts in the Shell as it does at the command prompt. For more information about messages MS-DOS displays, see xx.
3. When the backup is finished, press any key to return to Program Manager.
   To restore the files you backed up, you can use the Restore Fixed Disk command described in the following section.

Restoring Directories and Files

If you lose files you have backed up with the **backup** command, you can retrieve the copies stored on the backup disk with the **restore** command. If you use the **backup** command to make backup disks for your files, you must use the **restore** command to retrieve them. For information about the **backup** command, see the previous section.

Once you have created a backup disk, you can use the **restore** command to:

- Restore all the files on the disk to a specific directory or to a directory and its subdirectories.

- Restore selected files.

CAUTION Unless you use the /p switch, the restore command will overwrite files that changed since they were backed up. For information on the /p switch, see "xx" later in this section.

Restoring Files to a Directory

In Brief

To restore files backed up with the **backup** command, use the **restore** command. For example, the following command restores the files on the backup disk in drive B to their original locations in the root directory of drive C:

```
restore b: c:\*.*
```

The **restore** command requires a source directory, but it does not require a destination directory. The first parameter tells MS-DOS where to get the files from, and the second parameter tells MS-DOS which files to restore. For example, to restore all the files from the backup disk in drive B to the C:\WORD\HOME directory you would enter the following command:

```
restore b: c:\word\home\*.*
```

B: is the backup disk and C:\WORD\HOME\*.* specifies the files you want to restore—every file on the backup disk that came from the C:\WORD\HOME directory.

When MS-DOS restores the files, it puts them in the directory they came from. If the directory they came from no longer exists, MS-DOS creates it.

When you enter the preceding **restore** command, MS-DOS prompts you to insert the disk with the backup files in drive B. When you press a key, MS-DOS displays the date of the backup and starts copying from the BACKUP.001 file to the C:\WORD\HOME directory. As files are restored, MS-DOS lists them on your screen.

If the files you want to recover were backed up on more than one disk, MS-DOS prompts you to insert the other disks. If the files you want to restore are not on the disk you specified, MS-DOS displays a "File not found" message.

To tell MS-DOS to prompt you before it replaces a file that is read-only or has been changed since the last backup, use the /p switch. For example, if you enter the following command, MS-DOS asks you to confirm your choice if any of the files that are backed up on drive B are read-only or have been changed since they were backed up:

```
restore b: c:\word\home\*.* /p
```

NOTE The /p switch depends upon directory time stamps to determine which file is the most recent. Because time stamps are set according to your system clock, if you depend on the /p switch, make sure your clock is always accurate.

To restore files to a directory and its subdirectories, use the /s switch. For example, the following command restores files from the backup disk in drive B to the C:\WORD\HOME directory and all of its subdirectories:

```
restore b: c:\word\home\*.* /s
```

MS-DOS reads the CONTROL.001 file to find out which directory a file came from and then stores it there. Any directories that don't exist are created.

To make sure that every file on a backup disk is restored, use the /s switch and start restoring at the root directory. For example, the following command restores every file on the backup disk in drive A to its original location on drive C:

```
restore a: c:\*.* /s
```

Restoring Selected Files

In Brief

To restore selected files from a directory, use the MS-DOS wildcards as in the following command:

```
backup a: c:\win\*.xls
```

To see a prompt before MS-DOS replaces files that have changed since the last backup, use the /p switch as in the following command:

```
restore b: c:\*.* /p
```

You can restore a subset of the files that were backed up by typing a single filename or by using the MS-DOS wildcards. For example, to restore only the C:\WORD\HOME\OUTGO.XLS file from the backup disk in drive B, you could enter this command:

```
backup b: c:\word\home\outgo.xls
```

To restore only the files with a .DOC extension, you would enter this command:

```
restore b: c:\word\home\*.doc
```

Using the Shell to Restore Files

When you restore files from the Shell you are running the MS-DOS **restore** command from Program Manager. Therefore, you use the same switches and wildcards and some of the same procedures to restore files from the Shell as you would from the command line.

Switches and wildcards allow you to restore subdirectories or selected files. For more information about the switches and wildcards you can use with the **restore** command, see the previous xx sections.

You can use the Backup Fixed Disk and Restore Fixed Disk commands only if you have MS-DOS Shell installed on a hard disk.

Shell ▶ **To restore files:**

1. Choose Restore from the DOS Utilities group on the Program Manager screen.

   MS-DOS displays the Restore Utility dialog box.
2. To restore all of the files on the backup disk in drive A, choose OK. To restore selected files or change the drive that the backup disk is in, type the appropriate drive letter, switches and wildcards in the Parameters box and then choose OK.

   From this point on, MS-DOS displays the same messages and prompts in the shell as it does at the command prompt. For more information about the information MS-DOS displays, see xx.
3. When the operation is finished, press any key to return the Program Manager screen.

Verifying All Save Operations

In Brief

To verify any data that MS-DOS writes to disk, use the following **verify** command:

```
verify on
```

To stop verifying the data use this command:

```
verify off
```

In general, MS-DOS reads and stores information on your disks with no problems. Occasionally, due to problems with the magnetic media of the disk, information is not stored properly on your disk. To tell MS-DOS to check that the information is stored accurately on your disks, you can use the following **verify** command:

```
verify on
```

While **verify** is on, MS-DOS makes sure that none of the sectors it tries to save information in are bad. To turn **verify** off, use this command:

```
verify off
```

To see whether the command is currently on or off, enter this command:

```
verify
```

When **verify** is on, data is written to disks more slowly than when it is off.

Recovering Files from Bad Disks

If you find that MS-DOS or another program can no longer read a file or group of files, there may be one or more bad sectors on the disk where the files or their directories are stored. To recover the parts of the file or files that are not damaged, you can use the **recover** command.

Recovering Files

In Brief

To recover as much information as possible from a file or files that have bad sectors, use the **recover** command. For example, the following command attempts to recover the COMB.TXT file from drive A:

```
recover a:comb.txt
```

You cannot get back the part of a file that is stored in a bad sector, but you can recover the rest of the file with the **recover** command. For example, if you have found that part of the GRAY.HIC file on drive A is no longer readable by the program that created it, you could use the following command to try to recover some of the information in the file:

```
recover a:gray.hic
```

MS-DOS reads the file one sector at a time. If any of the sectors are damaged, MS-DOS removes them from the file. MS-DOS marks the bad sectors so that it knows not to store any more information there.

When the operation is complete, MS-DOS stores the recovered file in the root directory of the disk it came from. MS-DOS names the files it recovers sequentially, beginning with FILE0001.REC (FILE0001, FILE0002, FILE0003, and so on).

NOTE Even if a file is successfully recovered, it might be unusable if the lost information is necessary to the proper use of the file.

Recovering Disks and Directories

In Brief

If the directory of a disk is unusable, you may be able to recover some of the information on the disk with the **recover** command. For example, the following command recovers files from drive A:

```
recover a:
```

If you find that a disk is unusable because of problems in its directory, you can use the **recover** command to recover as much of the information on the disk as possible. For example, to recover files from drive A, you would enter this command:

```
recover a:
```

All the files that MS-DOS recovers are stored in the root directory of the disk they came from.

CAUTION The root directory can hold only a limited number of entries. If you try to recover more files than the root directory can hold, some files will be lost. In general, you should only use the **recover** command when it is absolutely necessary.

Giving a Drive Letter to a Directory

In Brief

To give a drive letter to a directory, use the **subst** command. For example, the following command assigns the drive letter A to the C:\BAR directory:

```
subst a: c:\bar
```

To remove the letter, use the **/d** switch. For example, the following command removes the association between A and C:\BAR:

```
subst a: /d
```

Sometimes, the program you are using will not accept any drive letters other than A and B. In these situations, you can use the **subst** command to fool the program into thinking that it is getting files from drive A or B when in reality the files are coming from a directory on your hard disk.

For example, suppose you are using a communications program that only accepts files from drive A or drive B. To use the files in C:\COMM with this program, you could enter the following command before you start the program:

```
subst a: c:\comm
```

Then, when the program requests files from drive A, MS-DOS will look in C:\COMM instead.

The drive letter you specify in the **subst** command must not be higher than the letter specified in the **lastdrive** command in the CONFIG.SYS file. For more information about the **lastdrive** command, see the *MS-DOS 5.0 User's Reference.*

When you are finished using the program, remove the association between the drive and the directory with the **/d** switch:

```
subst a: /d
```

The following commands ignore any associations you make with the **subst** command: **backup, format, chkdsk, diskcomp, diskcopy, fdisk, label, recover, restore, sys.**

Partitioning Your Hard Disk

Each operating system has its own conventions for storing files on a hard disk. If you use only MS-DOS, then it is no problem if your entire hard disk is set up using the MS-DOS conventions. However, if you want to use your hard disk with another operating system in addition to MS-DOS, you have to *partition* your hard disk into MS-DOS sections and non-DOS sections.

If you use only MS-DOS, you can create a single MS-DOS partition that occupies your entire disk. This gives MS-DOS access to all of the disk's storage capacity, and in some cases enables MS-DOS to gain access to the disk more quickly. If you use only MS-DOS but you would like to separate the files you store on your hard disk, you can create a second MS-DOS partition. In this case, MS-DOS still has access to the entire hard disk. However, the files in the second partition appear to be on a different drive.

If you want to use your hard disk with another operating system (for example XENIX or OS/2) you tell MS-DOS to use only part of the disk for its files. Then, you can use the other operating system to set up the remaining disk space for its files. You switch to another operating system by making its partition *active.*

Partitioning your disk is different from formatting it. When you partition a disk, you tell MS-DOS which sections of the disk it can use. When you format a disk, MS-DOS prepares an existing partition to receive files. After you partition your disk, you must still format each partition before it can be used.

You create one or more MS-DOS partitions on a hard disk by using the Fdisk program included with MS-DOS.

Using Fdisk from the MS-DOS Command Prompt

You can partition you hard disk by entering **fdisk** at the command prompt.

The Fdisk program displays a series of menus that allow you to:

- Display information about your current partitions.
- Create, change, or delete MS-DOS partitions.
- Make a partition active.

CAUTION If you use Fdisk to change the existing partitions on a hard disk, you lose the information contained in those partitions. Be sure you have copies of all the data in a partition before you use Fdisk to change the partition.

Using Fdisk from the Install Program

You normally start Fdisk from the MS-DOS command prompt. If you want to partition your hard disk when you install MS-DOS, you can start Fdisk from the installation program. During the installation program, MS-DOS checks to see if your disk is partitioned. If is not partitioned, MS-DOS asks you how you want it partitioned.

If you choose to create a single MS-DOS partition, MS-DOS creates the partition and continues with the installation. If you choose to create more than one partition, MS-DOS starts Fdisk. You can then set up partitions, using the information in this section. When you are finished creating partitions with Fdisk, MS-DOS continues the installation process. For more information about installing MS-DOS, see "XX".

Understanding Hard Disk Partitions

You can create two MS-DOS partitions on a hard disk:

- The *primary MS-DOS partition* is the main storage area for MS-DOS. If you want to start MS-DOS using a hard disk, that disk must have a primary MS-DOS partition.
- An *extended MS-DOS partition* provides a way to further divide the files on a disk. You do not have to create an extended partition.

Whatever disk space you don't allocate to a primary and extended MS-DOS partition can be made into one or more non-DOS partitions by another operating system. The total number of partitions on a hard disk cannot exceed four. Thus, if you create only a primary MS-DOS partition and do not give it all the space on the disk, you can create three non-DOS partitions from another operating system. If you create a primary and extended MS-DOS partition, and you don't use all the disk's storage capacity, you can create two non-DOS partitions from another operating system.

The Primary MS-DOS Partition

If you want to start MS-DOS using a hard disk, it must have a primary MS-DOS partition. If you want to start MS-DOS from your hard disk, the primary MS-DOS partition must contain the three MS-DOS system files. In general, the primary partition contains all the files that are on drive C. Most users require only this one partition. Your hard disk must have a primary MS-DOS partition to start MS-DOS.

You use Fdisk to set up a primary MS-DOS partition that takes up all the room on your hard disk, as shown in the following illustration:

This artwork is not available
for this release.

dsk_3

If you reserve half of the disk's space for the primary MS-DOS partition, the other half will be available for other partitions, as shown in this illustration:

This artwork is not available
for this release.

dsk_4

The Extended MS-DOS Partition

You can create a second MS-DOS partition on a hard disk, called an extended MS-DOS partition. An extended partition lets you treat a single hard drive as if it were two or more disk drives. When you create an extended partition, you divide it into one or more *logical drives.* A logical drive is a section of a hard disk that behaves like a separate disk drive. In other words, you can treat each logical drive in the extended MS-DOS partition as though it were its own disk drive.

For example, suppose you create an extended MS-DOS partition that takes up 10 MB of hard disk space. If you create one logical drive in the extended MS-DOS partition, that part of the hard disk is named D. Your system would have a C drive that contains the files in the primary partition and a D drive that contains the files in the extended partition. You can change the current drive to D and work with files and directories stored in those 10 MB of the hard disk as though they were on a separate disk drive.

If you divide the extended partition into two logical drives, your system would have a C drive with the files in the primary partition, a D drive with some of the disk space in the extended partition, and an E drive with the remaining space in the extended partition.

There are 26 letters available for drives (A through Z). A and B are reserved for floppy disk drives. C is usually reserved for the primary partition (unless you have additional floppy drives). That leaves a maximum of 23 logical drives that you can create within an extended partition.

Suppose you create a primary MS-DOS partition and an extended MS-DOS partition containing two logical drives. A diagram of your hard disk might look like this:

This artwork is not available
for this release.

dsk_5

Logical drives do not give you more total disk space, but they do give you more flexibility in storing files.

Non-DOS Partitions

Non-DOS partitions are partitions for other operating systems (such as XENIX). You cannot use Fdisk to create a non-DOS partition. For information about creating non-DOS partitions, see the documentation for the system you want to use.

If you want to use non-DOS partitions, be sure you leave room on your hard disk. For example, if you want to use 40 percent of a disk for a non-DOS partition, you should create MS-DOS partitions that take up a total of 60 percent of the disk space. You can create two or three non-DOS partitions on a disk, depending on the number of MS-DOS partitions you have.

If you use half your disk for MS-DOS and the other half for another operating system, a diagram of your disk looks like this:

This artwork is not available
for this release.

dsk_6

The Active Partition

To tell your computer to use the operating system stored in a partition, you make it active. To use MS-DOS, you make your primary MS-DOS partition active (because an extended MS-DOS partition does not contain the MS-DOS operating system it cannot be active). A hard disk can have only one active partition at a time.

If you have only a primary MS-DOS partition that occupies the entire disk, it is automatically the active partition. For more information about setting the active partition, see "XX" later in this chapter.

Using Fdisk

You use Fdisk to display partition information, create partitions and logical drives, set the active partition, and delete partitions and logical drives. This section introduces the basic information you need in order to use Fdisk.

CAUTION Fdisk destroys all existing files in any partitions you modify. If you are using Fdisk to change the partitions on a disk with files on it, be sure you have backup copies of all files you want to keep in the partitions to be modified before you begin. If you want to create smaller partitions on a hard disk that only has a large MS-DOS partition, you need to back up all files on the disk that you want to keep.

To start using Fdisk, enter the following command at the command prompt:

```
fdisk
```

The **fdisk** command has no parameters or switches. When Fdisk starts, it displays the main Fdisk menu:

This artwork is not available for this release.

dsk_7

Fdisk uses a series of menus to display options, help messages, and warnings. To choose a menu option, type the number next to the option you want and press ENTER. To return to a previous menu, press ESC. To exit Fdisk, return to the main Fdisk menu and press ESC.

Each Fdisk menu displays a "Current fixed disk drive" message, followed by a number. The term *fixed disk* means *hard disk*. If you only have one hard disk drive, this number will always be 1. If you have more than one hard disk drive, the number will show which disk Fdisk is currently working with. The first hard disk drive in your system is 1, the second disk drive is 2, and so on. Changing the current drive in Fdisk does not change the current drive when you return to the system prompt. The Fdisk current drive refers only to physical disk drives, not logical drives.

Displaying Partition Data

You can display information about the status, type, and size of the partitions on a hard disk by choosing Display Partition Information (option 4) from the Fdisk main menu. The Display Partition Information screen looks like this :

```
Display Partition Information

Current hard disk drive: 1

Partition Status Type Size in Mbytes Percentage of Disk Used
 C: 1    A  PRI DOS   21     50
% 2        EXT DOS   21     50

%Total disk space is 42 Mbytes (1 Mbyte = 1048576 bytes).

The Extended DOS partition contains
logical DOS drives. Do you want to
display logical drive information?  [Y]

Press ESC to continue
```

The information varies, depending on the number, size, and type of partitions on your hard disk. The "Partition" column shows the drive letter of the disk Fdisk is currently working with, and the number of each partition. The "Status" column displays the letter A next to the active partition. The "Type" column shows whether a partition is a primary MS-DOS partition (PRI DOS), an extended MS-DOS partition (EXT DOS), or a non-DOS partition (non-DOS). The "Size in Megabytes" column shows the size of each partition, and the "Percentage of Disk Used" column shows the percentage of the current disk that each partition occupies.

If there is an extended MS-DOS partition that contains logical drives, Fdisk asks if you want to see information about that partition's logical drives. Enter y to display this information.

The screen looks like this :

```
Display Logical DOS Drive Information

Drv Volume Label Mbytes System Usage
D:  BACKUPA     18  DOS   90
%E:  BACKUPB     2  DOS   10

%Total Extended DOS Partition size is 20 Mbytes (1 Mbyte = 1048576
bytes)

Press ESC to continue
```

The information on the screen varies depending on the number and size of the logical drives. The "Drv" column displays the drive letter of each logical drive. The

"Volume Label" column shows the label assigned to each drive. "Mbytes" is the size of each logical drive and "Usage" shows the percentage of the available space in the extended MS-DOS partition that each logical drive takes up.

Creating a Primary MS-DOS Partition

The first hard disk you use with MS-DOS must have a primary MS-DOS partition. You can create a primary MS-DOS partition that reserves all of a hard disk's storage space or only a part of it. If you want to create an extended MS-DOS partition with logical disk drives, or if you want to leave room for a non-DOS partition, you need to create a primary MS-DOS partition that is smaller than the total size of your disk.

To change the size of an existing primary MS-DOS partition, delete it and recreate a new one that is the size you want. You will lose any information stored in the existing primary MS-DOS partition, so be sure you have backup copies of any files you want to save. See "XX" later in this chapter for information about deleting a partition.

If your hard disk already has a partition on it, you must delete it before you can create a primary MS-DOS partition that occupies the entire disk. If your hard disk does not already have a partition you can use the following procedure to create a primary MS-DOS partition that occupies the entire disk.

▶ **To create a primary MS-DOS partition that occupies the entire the hard disk:**

1. Start Fdisk and then press ENTER to choose Create MS-DOS Partition or Logical Drive (option 1) from the Fdisk main menu.

   The Create DOS Partition menu appears.

2. Press ENTER to choose Create Primary DOS Partition (option 1).

   Another menu appears, with this message:

   ```
   "Do you wish to use the maximum size for a Primary DOS Partition
   and make the partition active (Y/N).........? [Y]
   ```

3. Enter **y**. If you enter **n**, Fdisk prompts you to create a smaller primary partition. See the following procedure for more information.

   Fdisk creates a primary partition that occupies the entire hard disk. If you have only one hard disk MS-DOS displays the following message:

   ```
   System will now restart

   Insert DOS Install disk into drive A:
   Press any key when ready
   ```

4. Insert an MS-DOS system disk and press any key. You still need to format your hard disk before you can use it. For more information about formatting, see "XX".

▶ To create a primary MS-DOS partition that occupies part of the hard disk:

1. Press ENTER to choose Create DOS Partition or Logical DOS Drive (option 1) from the Fdisk main menu.

   The Create DOS Partition menu appears.

2. Press ENTER to choose Create Primary DOS Partition (option 1).

   Another menu appears, with this message:

   ```
   "Do you wish to use the maximum available size for a Primary DOS
   Partition
   and make the partition active (Y/N).........? [Y]
   ```

3. Enter **n.**

   A second Create Primary DOS Partition menu appears.

4. Press ENTER if you want the default size (100%). Otherwise, enter the number of megabytes or percentage of the disk to use. If you enter a percentage, follow the number with a % sign.

   The following message appears:

   ```
   Primary DOS Partition created, drive letters changed or added.
   ```

5. Press ESC to return to the Fdisk main menu.

   When you leave Fdisk you will need to format the new partitions on your hard disk. For more information see "XX", later in this chapter.

Any part of the disk that you do not use for the primary MS-DOS partition can be used for an extended MS-DOS partition, or for non-DOS partitions.

NOTE When you create a primary MS-DOS partition that does not take up all the room on a hard disk, you must make the primary MS-DOS partition active before you can use the hard disk with MS-DOS. For information about making a MS-DOS partition active, see "XX" later in this chapter.

Creating an Extended MS-DOS Partition

To divide your hard disk into two MS-DOS storage areas, you create an extended MS-DOS partition. Within the extended MS-DOS partition you can create one or more logical drives. Logical drives are parts of your hard disk that MS-DOS treats as separate disk drives. You can create only one extended MS-DOS partition on a hard disk, but you can create up to 23 logical drives in extended MS-DOS partitions on your hard disks.

If you have one hard disk, there must be a primary MS-DOS partition on the disk before you can create an extended MS-DOS partition. If you have more than one hard disk, the first disk in your system must have a primary MS-DOS partition, but you can create extended MS-DOS partitions on your additional disks without creating a primary partition on them.

▶ **To create an extended MS-DOS partition:**

1. Choose Create DOS Partition or Logical DOS drive (option 1) from the main Fdisk menu.
2. Choose Create Extended DOS Partition (option 2) from the Create DOS Partition menu.

   Fdisk displays a menu that shows the total number of megabytes available for an extended partition. The default for the partition size is the maximum available space on the hard disk drive minus the size of the primary partition. If there is no space available, you must delete and recreate the primary MS-DOS partition so it is smaller, or reduce the size of any non-DOS partitions that exist.
3. Press ENTER to choose the default size. Otherwise, type the number of megabytes or the percentage of the disk to be used for the extended MS-DOS partition. If you enter a percentage, follow the number with a % sign.

   The Create Logical Drive(s) in the Extended DOS Partition menu appears.

   When you create the extended MS-DOS partition, you can set up one or more logical drives. See the following section for more information.

NOTE If Fdisk finds any defective tracks at the start of an extended MS-DOS partition, it adjusts the partition boundaries to avoid bad tracks.

Creating Logical Drives in the Extended MS-DOS Partition

To store information on the portion of the hard disk assigned to the extended MS-DOS partition, you need to create one or more logical drives. Logical drives are sections of the extended MS-DOS partition that MS-DOS treats as disk drives. You can use logical drives to organize the information on your hard disk. When you create logical drives in an extended MS-DOS partition, they are assigned drive letters. For example, if you create one logical drive on a hard disk with the drive letter C, the logical drive is given the drive letter D. For more information about how drive letters are assigned to logical drives, see the next section, "XX".

You can store and retrieve information on your logical drives as though they were physical disk drives. For example, you can use logical drive D to store files for a particular application, and work with them by specifying drive D rather than a directory location.

▶ **To create or modify logical drives:**

1. Create an extended MS-DOS partition. See the preceding section for more information.
2. On the Create Logical Drive(s) menu, enter the number of megabytes or the percentage of the partition space for the first logical drive you want to create. If you enter a percentage, follow the number with a % sign. If you want one logical drive to take up the whole extended MS-DOS partition, press ENTER to accept the default (the entire disk).
3. Continue entering the sizes of partitions until you have used up the entire partition or until you have created all the logical drives you want.

   If the entire partition is assigned to logical drives, Fdisk exits the menu automatically. To exit from the menu before all the space has been allocated, press ESC.

After you create logical drives, you must format them before MS-DOS can use them. For more information about formatting logical drives, see "XX", later in this chapter.

How Drive Letters Are Assigned

The primary partition of the first hard disk in your system is normally drive C. The drive letters of additional hard disks and logical drives depend on the number of disks and how they are partitioned.

NOTE If you have three or four floppy disk drives in your system, the primary partition of the first hard disk may be D or E. In the examples in this section, it is assumed that the primary partition of the first hard disk is C.

If you have only one hard disk, any logical drives you create in the extended MS-DOS partition are given letters beginning with D. For example, if you create five logical drives in the extended partition, they are named D, E, F, G, and H.

If you have more than one hard disk, your first hard disk must have a primary partition (this partition is normally drive C). If you do not have any other primary partitions in your system, all the logical drives you create in the extended MS-DOS partitions on your hard disks are assigned letters consecutively.

Suppose you have two hard disks in your system: the first hard disk has a primary partition and an extended partition with two logical drives, and the second hard disk has an extended partition with two logical drives and no primary partition. The primary partition on the first disk is drive C, the two logical drives on the first hard disk are drives D and E, and the two logical drives on the second hard disk are drives F and G as shown in the following illustration.

This artwork is not available
for this release.

dsk_8

If you have a primary MS-DOS partition on more than one of the hard disks in your system, drive letters are first assigned consecutively to all the primary partitions, and then assigned consecutively to the logical drives in the extended partitions.

Suppose you have two hard disks in your system: both hard disks have a primary partition and an extended partition with two logical drives. The primary partition on the first disk is drive C, the primary partition on the second disk is drive D, the logical drives on the first disk are drives E and F, and the logical drives on the second disk are drives G and H as shown in the following illustration.

This artwork is not available
for this release.

dsk_9

If you have existing logical drives on a hard disk and you add an additional hard disk, the logical drives are given new letters if you create a primary MS-DOS partition on the new disk. If you create only an extended MS-DOS partition, the drive letters on the first disk are unchanged. For example, suppose you have a logical drive on drive C that is named D. If you add a hard disk to your system and create a primary MS-DOS partition on it, the primary partition on the new disk becomes drive D and the logical drive on the old disk becomes drive E.

On the other hand, if you add a hard disk to your system and create only an extended MS-DOS partition on it, the drives on the old disk are unaffected. Any logical drives that you create on the new disk are given the next available drive letter. For example, suppose you have one hard disk with a primary partition (drive C) and two logical drives (drives D and E). If you add a second hard disk and create an extended partition with two logical drives, these drives are named F and G. The drive letters on the first disk remain unchanged.

Setting the Active Partition

The active partition contains the operating system your system loads when you start or reset it. Unless you create a single MS-DOS partition that takes up all the room on your hard disk, you need to set the active partition with Fdisk. If you are using a non-DOS partition, you need to reset the active partition when you want to switch between MS-DOS and the non-DOS operating system. Only one partition can be active at a time.

▶ To set the active partition:

1. Choose Set Active Partition (2) from the Fdisk main menu.

   A menu appears that displays the status of each partition. The active partition is indicated with the letter A.
2. Enter the number of the partition you want to activate. The default setting is the current active partition number.
3. Press ESC to return to the Fdisk main menu.

If your hard disk drive contains only MS-DOS partitions, only the primary MS-DOS partition can be active. If you try to activate an extended MS-DOS partition, **fdisk** displays the following message:

```
Partition selected (3) is not startable, active partition not changed.
```

Deleting an MS-DOS Partition or Logical Drive

You might need to change the size of the partitions that currently exist on a hard disk for several reasons. You might need to make room for a non-DOS operating system, or you might want to change the size and number of the logical drives. You cannot reduce or enlarge an existing partition. If you want to change a partition's size, you need to delete it and recreate it again in the size you want. When you delete a partition, all the information in the partition is erased and cannot be recovered. Before you delete a partition, be sure you have backup copies of the information you want to save in the partition. When you delete a partition, you will not lose the information in the other partitions on a disk.

For example, if you just want to delete the extended MS-DOS partition but not the primary MS-DOS partition, the files in the primary MS-DOS partition will not be affected. If you want to delete the primary MS-DOS partition on your first hard disk, you also need to delete the extended MS-DOS partition and all logical DOS drives.

You can delete one or more of the logical drives in the extended MS-DOS partition of a hard disk. All the information on a logical drive is lost when you delete it. Deleting one logical drive does not affect the information on other logical drives. However, if there are logical drives on a disk that have higher drive letters than the drive you delete, these letters will be changed. Suppose you have logical drives D, E and F on a disk. If you delete drive D, drive E becomes drive D and drive F becomes drive E.

▶ **To delete an MS-DOS partition or logical DOS drive:**

1. Choose Delete DOS Partition or Logical Drive (option 3) from the Fdisk main menu.

   The Delete DOS Partition menu appears.
2. Enter the number of the option you want.

   Fdisk displays the status of the partition or logical drives, along with a message warning you that the data in the partition or logical drive will be lost.
3. Enter the drive letter and volume label of the logical drive you want to delete.

   Fdisk displays a message confirming the information you entered.
4. Enter y to delete the partition or drive.

NOTE To continue using MS-DOS after you have deleted the primary MS-DOS partition, you must restart your system with an MS-DOS system disk in drive A.

Working with Additional Hard Disks

If your system has more than one hard disk drive, you can use Fdisk to create and modify partitions on any of the drives you have. Only the first disk needs to have a primary MS-DOS partition. Your other disks can have primary MS-DOS partitions or extended MS-DOS partitions.

When you start Fdisk, it assumes you want to work with the first hard disk on your system. To work with a different disk drive, you must choose Change Current Fixed Disk Drive (option 5) from the Fdisk main menu, and specify the number of the drive you want to partition. If you only have one hard disk drive, the Change Current Fixed Disk Drive option does not appear on the Fdisk main menu.

You do not need to create a primary MS-DOS partition on additional hard disks in your system. Instead, you can create extended MS-DOS partitions with one or more logical drives.

When you finish creating partitions on an additional hard disk, you must format the partitions. See "XX" for more information. You do not need to use the /s switch or the sys command to transfer the system to additional hard disks.

Formatting Your Hard Disk After Using Fdisk

Fdisk reapportions the space on your hard disk, but it does not reformat it. After using Fdisk, you must use the **format** command to prepare any partition that you have created or changed.

You must format each new partition separately. For example, if you made your primary MS-DOS partition (drive C) smaller and created two logical drives in an extended partition (drives D and E), you must use the **format** command three times:

```
format c:
format d:
format e:
```

The first command formats the primary partition. The second and third commands format the logical drives.

When you leave Fdisk after you have changed the sizes of any of the MS-DOS partitions on any of your hard disks, Fdisk displays this message:

```
System will now restart
```

If you changed the size of your primary MS-DOS partition, Fdisk prompts you to insert an MS-DOS system disk in drive A and press any key. Otherwise, you do not need to insert an MS-DOS system disk. In either case, you return to the command prompt. You can now format one or more of the partitions on your hard disks with the **format** command.

If you are formatting the primary MS-DOS partition of the hard disk from which you will start your system, be sure to transfer the MS-DOS system with the /s switch or the sys command. For more information, see "XX".

CAUTION If you have made changes to some of the partitions or logical drives in your system, but not all of them, be very careful when you format the partitions or drives you have changed. Since Fdisk may assign different letters to drives after you change partitions or logical drives, you may inadvertently format a drive that has information stored on it.

Before you format a drive in this situation, use the **chkdsk** command to check the contents of the drive. If you see the message "Probable non-DOS disk" before the disk information is displayed, the drive is not formatted. If the disk information is displayed without this message, the drive is formatted.

You may also want to give descriptive labels to your logical drives when you create them so that you can be sure which drive has what information when you make changes to your system.

For more information about formatting, see "XX".

Changing to Non-DOS Partitions

Fdisk can only create MS-DOS partitions. For information about creating non-DOS partitions, see the documentation for the non-DOS system you want to use. However, you can use Fdisk to make a non-DOS partition the active partition on your system by choosing Set the Active Partition from the Fdisk main menu and re-starting your system. If you decide later that you want to make the primary MS-DOS partition the active partition, start your system from drive A with an MS-DOS system disk, then run Fdisk to make the primary MS-DOS partition active.

Chapter 8

Advanced Command Techniques

8

This chapter describes three MS-DOS features for working with commands: redirection, command editing keys, and the Doskey program. *Redirection* means changing the place that a command gets information from or sends information to. Redirection is useful when you want MS-DOS to save information in a file rather than displaying it on the screen. You can also redirect the information that a command usually sends to the screen through a *filter command.* Filter commands help you sort, display, and select parts of a command's output.

MS-DOS provides a number of command editing keys that let you quickly redisplay and edit your last command rather than retyping it.

The Doskey program records the commands you enter so that you can use them again rather than retype them. Doskey uses the MS-DOS command editing keys plus a number of other keys that make it even easier to edit commands. In addition, Doskey lets you create macros that contain a series of commands that you can run like a batch file.

Redirecting Command Output and Input

Unless you tell it otherwise, MS-DOS gets input for a command from the keyboard and sends the output of the command to the screen. Sometimes it is useful to redirect the input or output to a file. For example, you might want to redirect a directory listing from the screen to a file. Or you might want the information that the **more** command displays to come from a file.

To tell MS-DOS to redirect the input or output of a command, you use one of the following *redirection characters*:

- The greater-than sign (>) tells MS-DOS to redirect the output of a command to a file or printer.

- The less-than sign (<) tells MS-DOS to get the input for a command from a file rather than from the keyboard.

The following sections describe some of the uses of the redirection characters.

NOTE If you use a program such as MANS or LINK that prompts you for any parameters you don't enter with the program name, you can use input redirection to tell the program to look in a file for the parameters.

Redirecting the Output of a Command

Almost all commands send some output to your screen. Even the commands that send output to a disk drive or printer also display messages and prompts on the screen.

To redirect the output from the screen to a file or to your printer, use the greater-than sign (>). You can use the greater-than sign with any MS-DOS command. For example, the following command redirects the directory listing that the **dir** command produces to the DIRLIST.TXT file:

```
dir > dirlist.txt
```

If the DIRLIST.TXT file doesn't exist, MS-DOS creates it. If DIRLIST.TXT exists, MS-DOS replaces the information in the file with the output from the **dir** command. For example, the following command creates a file called CHECKDSK.TXT that contains the output of your most recent **chkdsk** command:

```
chkdsk a: > checkdsk.txt
```

MS-DOS replaces the contents of CHECKDSK.TXT with the output that the **chkdsk** command usually sends to the screen.

To add the output from a command to the end of a file without losing any of the information already in the file, use a double greater-than sign (>>). For example, the following command appends the directory listing that the **dir** command produces to the DIRLIST.TXT file:

```
dir >> dirlist.txt
```

To send the output of a command to the printer, use the greater-than sign (>) with the name of the port to which the printer is connected. For example, the following command redirects the output of the **dir** command from the display to the printer attached to the LPT1 port:

```
dir > lpt1:
```

Redirecting the Input to a Command

Just as you can tell MS-DOS to send the output of a command to a file or printer rather than to the screen, you can tell it to take the input for a command from a file rather than from the keyboard. To tell MS-DOS to take input from a file, use the less-than sign (<). For example, the following command tells MS-DOS to take the input for the **sort** command from the LIST.TXT file:

```
sort <list.txt
```

MS-DOS alphabetizes the lines of the LIST.TXT file and displays the result on your screen. If you enter the **sort** command without a less-than sign, MS-DOS assumes that you will enter the lines you want alphabetized from the keyboard.

Passing Information Through Filter Commands

Filter commands divide, rearrange, or extract portions of the information you pass through them. MS-DOS has three filter commands:

- The **more** command displays files and command output one screen at a time.
- The **find** command searches through files and command output for the words that you specify.
- The **sort** command alphabetizes files and command output based on the first character of each line.

To tell MS-DOS that a filter command should get its input from a file, use the less-than sign (<). To tell MS-DOS that the filter command should get its input from another command, use the pipe character (|). The following sections describe how you use redirection characters with these the three filter commands.

Controlling the Screen Display with the More Command

The **more** command tells MS-DOS to display the contents of a file or the output of a command one screen at a time.

For example, to display the contents of the LIST.TXT file one screen at a time, you would use the following command:

```
more < list.txt
```

Between each screen of information, MS-DOS displays a "--More--" message. To continue to the next screen of information, press any key. To end the command without displaying any more information, press CTRL+C.

The **more** command is particularly helpful if you are working with a command that produces a lot of output. For example, suppose you want to display a directory tree for your hard disk. If you have more directories than will fit on a single screen, you can follow the **tree** command with a pipe and a **more** command as in the following example:

```
tree c:\ | more
```

MS-DOS displays the first screen of output from the **tree** command and then pauses until you press a key.

Searching for Text with the Find Command

The **find** command searches one or more files for the text you specify. When it finds a line with text that matches the text you specified, it displays the whole line on your screen. The **find** command can be used as a filter command or as a regular MS-DOS command. For information about using **find** like a regular MS-DOS command, see **xx**.

To use the **find** command as a filter command, include a less-than sign and a filename to tell MS-DOS which file to search for the text. For example, the following command finds occurrences of the text *Pacific Rim* in the file TRADE.TXT:

```
find "Pacific Rim" <trade.txt
```

To tell MS-DOS to save the output of the **find** command rather than simply displaying it, use a greater-than sign (>) and a filename where you want the output stored. For example, the following command finds occurrences of the "Pacific Rim" in the TRADE.TXT file and saves them in the NWTRADE.TXT file:

```
find "Pacific Rim" <trade.txt >nwtrade.txt
```

To print the output rather than displaying it, use a greater-than sign and the name of the port your printer is attached to, as in the following command:

```
find "Pacific Rim" <trade.txt >LPT1:
```

Sorting Text Files

The **sort** command alphabetizes a text file or the output of a command based on the first letter of each line. In the previous section, you saw how the following command sorts the LIST.TXT file and displays the result on your screen:

```
sort <list.txt
```

The command tells MS-DOS to sort the lines of the LIST.TXT file and display the result without changing the file. If you don't include a less-than sign, MS-DOS assumes you want to enter the lines to be sorted directly from the keyboard.

To save the output of the **sort** command rather than displaying it, include a greater-than sign (>) and a filename in the command. For example, the following command alphabetizes the lines of the LIST.TXT file and stores the result in the ALFLIST.TXT file:

```
sort <list.txt >alflist.txt
```

To sort the output of a command, enter the command followed by a pipe character (|) and the **sort** command. For example, the following command sorts the output of the **find** command:

```
find "Jones" maillst.txt | sort
```

When you enter this command, rather than simply finding and listing the occurrences of the word *Jones* in the file MAILLST.TXT, MS-DOS alphabetizes these occurrences and then displays them.

Combining Commands with Redirection Characters

You can combine filter commands, regular commands and filenames to make custom commands. For example, you can use the following command to store the names of files in the current directory that contain the characters *log*:

```
dir | find "log" > loglist.txt
```

MS-DOS passes the output of the **dir** command through the **find** filter command. MS-DOS stores the lines that contain the characters *log* in the LOGLIST.TXT file. The result is a stored list of filenames that contain *log* (for example, A.LOG, LOGDAT.SVD, and MYLOG.BAT).

To use more than one filter in the same command, separate the filters with a pipe character. For example, the following command finds the files in every directory on drive C whose names include the characters *log* and displays them one screen at a time:

```
dir c:\ /s /b | find "log" | more
```

This **dir** command would normally display every file on the disk. However, in this case, MS-DOS sends the output of the **dir** command through the **find** command. The find command selects only the files that contain the characters *log*. The **more** command displays the files that are selected by the **find** command one screen at a time.

Using Command Editing Keys

MS-DOS provides several command editing keys that you can use to edit the last command you typed. For example, if you misspell the name of a file in a copy command, rather than retyping the whole command, you can use the command editing keys to redisplay the command and change the part that is misspelled. Or if your next command is similar to your previous command, you can include parts of the previous command in the command that you are typing.

When you enter a command, MS-DOS runs the command and also saves it in a temporary location called the *template*. For example, suppose you type the following command:

```
type ada.txt
```

When you press ENTER, MS-DOS runs the command and also copies it to the template. Whatever command was in the template is replaced by the command you entered. Thus, the template saves only the very last command you entered. For information about saving and reusing all the commands that you enter, see xx later in this chapter.

You can use the command stored in the template as a resource when you type your next command. To copy characters from the command in the template to the command you are typing, you use the MS-DOS command editing keys.

NOTE MS-DOS might clear the template if you load large programs.

The following table shows the command editing keys and how they help you use the command that is stored in the template. Each key can affect two commands: the command that you are typing and the command in the template.

To do this	Press this key
Delete the character before the cursor on the display and move back one character in the template.	LEFT ARROW or BACKSPACE
Delete the character above the cursor on the display and move forward one character in the template.	DEL
Insert new characters on the display without overtyping the existing characters on the display or moving forward in the template.	INS

To do this	Press this key
Copy the next character in the template to the display.	F1 (or RIGHT ARROW)
Copy characters up to the character you specify from the template to the display.	F2
Copy all of the remaining characters from the template to the display.	F3
Move forward in the template to the character you specify without copying characters to the display.	F4

The trick to using the template is knowing where you are in the template as you type a new command. In general, as you type a character on the display you move forward one character in the template. For example, if you type three new characters, the next character in the template is character number four from your last command.

This artwork is not available
for this release.

com_1

Redisplaying and Reentering a Command

Suppose you enter the following command

```
copy c:\word\*.doc a:
```

When you press ENTER, MS-DOS runs the command, copies it to the template and displays a new command prompt. Before you have typed any new characters, the cursor is at position one on the display, and the next character in the template is the first character of your previous command (C).

This artwork is not available for this release.

com_2

At this point, you can copy the entire previous command by pressing F3. For example, suppose you wanted to copy the *.DOC files onto two disks. After you have entered the command once, you can press **F3** to redisplay the command from the template:

```
copy c:\word\*.doc a:_
```

MS-DOS redisplays the command and places the cursor at its end. The result is exactly as if you had retyped the command. To copy your files to a second disk, you would insert a different disk in drive A and press ENTER. MS-DOS executes the **copy** command again.

Fixing a Misspelled Command

Once you are familiar with using the command editing keys, it is often easier to edit your last command than to type it again. The key to editing your previous commands is to remember where you are in the template.

With just the F3 and LEFT ARROW keys you can quickly fix a command that you mistyped. For example, suppose you typed *.DIC when you meant to type *.DOC as in the following command:

```
copy c:\word\*.dic a:
```

Rather than retyping the command, you can edit the old one. To edit the command, first press **F3** to redisplay the command from the template:

```
copy c:\word\*.dic a:_
```

MS-DOS redisplays the command with the cursor at its end. To change DIC to DOC, press the LEFT ARROW key five times to move the cursor back to the first incorrect letter:

```
copy c:\word\*.d_
```

MS-DOS deletes the letters from the display and moves back five characters in the template. Then, to correct the command, retype the remaining portion:

```
copy c:\word\*.doc a:_
```

To run the corrected command, press ENTER.

You can use this method to correct a command or even change one command into another. For example, if you want to copy the *.DOC files and then copy the *.BAK files, you can enter the **copy** command once for the *.DOC files and then change it so that it copies the *.BAK files.

Editing a Command

Rather than having to retype information that follows the part of the command you changed, you can use command editing keys to work with the characters stored in the template.

For example, in the previous section, you saw how to use the LEFT ARROW key to delete characters from the display. Instead of correcting the command by retyping the remaining part, you can take advantage of the fact that as you backspaced on the display you also moved back in the template. After pressing LEFT ARROW five times, the command line and the template look like this:

This artwork is not available
for this release.

com_3

After you type the correct character, the display has one more character, and you have moved forward one character in the template:

This artwork is not available for this release.

com_4

To copy the rest of the characters from the template to the display, press F3.

```
copy c:\word\*.doc a:_
```

Notice that when the command line is blank, F3 redisplays the entire previous command from the template. After you have typed part of a new command, F3 redisplays the template from the next character on.

Editing with F1 and INS

Suppose you have the following command stored in the template:

```
dir c:\word\finals\*.bak
```

To redisplay the command one character at a time you press the F1 key. For example, if you press F1 11 times, you will display the first 11 characters of the command:

```
dir c:\word_
```

To insert text in the command without changing where you are in the template, press INS before typing the new letters. For example, to change WORD to WORD-NEW, press INS and then type NEW:

```
dir c:\wordnew_
```

To continue copying characters from the template to the command line, you press F1 again. For example, to copy \FINAL from the template to the command line, press F1 six times:

```
dir c:\wordnew\final_
```

Editing with DEL

To avoid copying characters from the template to the command line, you press DEL. For example, to skip the next character in the template (the S in FINALS) you press DEL once:

```
dir c:\wordnew\final_
```

Because there is no character above the cursor, DEL has no effect on the display. However, in the template, the DEL moved forward one character. If you press F3 to copy the rest of the line from the template to the display, you'll notice that the S from FINALS is not copied:

```
dir c:\wordnew\final\*.bak_
```

Editing with F2 and F4

Often it is most convenient to use only part of the command that is stored in the template. To tell MS-DOS to copy characters from the template up to a certain position, you press F2 and type the character you want to copy up to. For example, suppose you have the following command in the template:

```
dir c:\wordnew\final\*.bak
```

To change the command so that it gives you a directory listing for C:\WORD-NEW, you can use the F2 key. To copy the characters in the template up to the first backslash (\), press F2 and type a backslash:

```
dir c:_
```

MS-DOS copies the characters up to but not including the backslash. The next character in the template is the first backslash in the command. To copy the characters up to the next backslash, press F2 and type backslash once more:

```
dir c:\wordnew_
```

Now you have the command that you want. To run it, you press ENTER.

To move forward in the template without copying characters to the display, you press F4 and type the character you want to move forward to. For example, suppose the following command is in the template:

```
dir c:\wordnew\final\*.bak
```

You can use the F4 key to transform the command in the template into the following command:

```
dir c:\wordnew\*.bak
```

First, to copy the characters up to the second backslash you can press F2, press W, and press F1 twice. Afterwards, the display and the template look like this:

This artwork is not available for this release.

com_5

To move to *.BAK in the template without copying any characters to the display, press F4 and type asterisk (*). At this point, the template and the display look like this:

This artwork is not available for this release.

com_6

Finally, to copy the rest of the characters from the template to the display, press F3.

Using Doskey to Work with Commands

The **doskey** command starts the Doskey program, which you can use to display, edit, and execute MS-DOS commands you have entered previously. Doskey uses the MS-DOS command editing keys and adds other keys that make it very easy to reuse commands. With the template that was discussed in the previous section you can reuse the last command you entered. With Doskey, you can reuse many of your previous commands (the exact number depends on how you set up Doskey).

In addition, you can create, run and save command *macros.* A command macro is a series of MS-DOS commands that are stored in RAM and run like a batch file. The first time you use the **doskey** command, MS-DOS loads the Doskey program into RAM. Thereafter, MS-DOS will save your previous commands and any macros you create.

Finally, Doskey allows you to enter several commands on one line. This enables you to speed up your command entry.

Although Doskey gives you more editing power than the MS-DOS editing keys, it does take up a small amount of your computer's temporary memory. If you need the maximum amount of memory for other purposes, you might want to use the editing keys instead of installing Doskey.

Installing Doskey

To begin recording commands, load the Doskey program into RAM with the following command:

```
doskey
```

Unless you tell it otherwise, MS-DOS reserves 512 bytes of temporary memory for the commands and macros you record. If your average command is 15 characters long, you can store about 35 commands with the amount of memory MS-DOS automatically reserves. The Doskey program itself occupies about 3K of memory.

If you want to reserve more or less memory, you can include the **/bufsize=** switch in the command. For example, to reserve 206 bytes of memory for recorded commands, enter the following command:

```
doskey /bufsize=206
```

If you enter more commands than can fit in the buffer you reserved, the oldest commands are removed from the buffer to make room for the newer ones.

To tell MS-DOS to reinstall Doskey after it has been installed once, use the **/reinstall** switch as in the following example:

```
doskey /reinstall
```

This option is useful if you want to use Doskey with a Microsoft Windows ® command prompt.

Typing More than One Command On a Single Line

Normally, you type a single command on a line and then press ENTER to run the command. However, once you install Doskey, you can type as many commands on a single line as you like. To type more than one command on a line, you separate the commands by pressing CTRL+T. The CTRL+T character appears on your screen as a paragraph mark (¶).

For example, to delete all the files in the \TMP directory and then remove the directory you could enter the following two commands on the same line:

```
del \tmp\*.* ¶ rd \tmp
```

MS-DOS runs the **del** command and prompts you as usual to confirm the deletion. When the first command is complete, MS-DOS runs the second command.

You can type as many commands as you like on a line as long as the total line length does not exceed 128 characters.

Displaying Previous Commands

Once Doskey is loaded, it keeps a list of your commands as you enter them. To run a command again, you redisplay it and press ENTER. When you reuse a command, Doskey adds it to the end of the list again. You redisplay commands using the following keys:

To do this	Press this key
Display the list of commands Doskey has recorded.	F7
Display the previous command in the list.	UP ARROW
Display the next command in the list.	DOWN ARROW
Search the list of stored commands for one that begins with the text you type.	F8
Choose the number of the command you want to display.	F9
Display the oldest command in the list.	PAGE UP
Display the newest command in the list.	PAGE DOWN
Erase the command that is displayed.	ESC

For example, suppose you enter the following three commands after you load Doskey:

```
copy c:\word\*.doc c:\bakup\
dir c:\bakup\*.doc
del c:\word\*.doc
```

The three commands are saved and you can redisplay them using the keys in the preceding list. To display the full list of commands, press F7. Doskey displays a numbered list of the commands it has saved:

```
1: copy c:\word\*.doc c:\bakup\
2: dir c:\bakup\*.doc
3:>del c:\word\*.doc
```

The command that you most recently used or displayed is marked with a greater-than sign (>). This command is called the current command. When you move through the list, you start from the current command.

NOTE If there are more commands in the list than can fit on a single screen, Doskey pauses at the bottom of each screen of commands.

Displaying the Next or Previous Command

The first time you press the UP ARROW key, Doskey displays the current command. For example, if Doskey has stored the three commands shown in the previous section, and command number 3 is current, Doskey displays the following command the first time you press the UP ARROW key:

```
del c:\word\*.doc_
```

Doskey redisplays the command and positions the cursor at its end. You can reuse the command by pressing ENTER or you can edit the command using the keys described later in this chapter.

If you press the UP ARROW key more than once, Doskey displays commands further back in the list. To move backward in the list and display command number 2, you would press the UP ARROW key twice. Now, the current command is number 2. If you press the UP ARROW key again, Doskey displays command number 1 (the **dir** command). If you press the UP ARROW key once more, for a total of four times, Doskey jumps from the oldest command to the newest and displays the **copy** command again.

Likewise, to move forward in the list, press the DOWN ARROW key. For example, if the current command is number 2, the first time you press the DOWN ARROW key Doskey displays this command:

```
dir c:\bakup\*.doc_
```

If you press the DOWN ARROW key again, Doskey displays command number 3. If you press the DOWN ARROW key a total of three times, Doskey jumps from the newest command to the oldest and displays command number 1.

Displaying the First or Last Command

To display the most recent command on the list, press PAGE UP. To display the oldest command press PAGE DOWN. For example, suppose Doskey has saved the following list of commands:

```
1: copy c:\word\*.doc c:\bakup\
2: dir c:\bakup\*.doc
3: del c:\word\*.doc
```

If you press PAGE UP, Doskey displays the following command:

```
del c:\word\*.doc_
```

If you press PAGE DOWN, Doskey displays this command:

```
copy c:\word\*.doc c:\bakup\_
```

Displaying Other Commands in the List

To choose a specific command from the list that Doskey has saved, use the F9 or F8 key. Use F9 to display a line by typing its number. For example, suppose Doskey has saved the following list of commands:

```
1: a:
2: dir
3: c:\myuts\figdsk a: time=30 space=35.8
4: dir 5: del *.tmp
```

If you want to use command number 3 again, you can display it using the ARROW keys or by pressing F9. When you press F9, Doskey asks you which line you want to display:

```
Line number:
```

To redisplay line 3, enter 3.

You can use F8 to display a line by typing the first few characters of the command. For example, to display the command that begins with C:\, type C:\ at the command prompt and then press F8.

When you press F8, Doskey searches through the list for a command that begins with the letters you typed. If it doesn't find a matching command, nothing happens. If Doskey finds a command that matches, it displays it.

If you press F8 again, Doskey finds the next oldest command that begins with the characters you typed.

Editing and Reusing Previous Commands

As you type a new command, or after you display a previous command, you can use command editing keys to change the command. You can use the same command editing keys with Doskey as you use with the command template described in Using Command Editing Keys, earlier in this chapter. When you use some of these keys with Doskey, however, you see slightly different results. Doskey also provides a number of other command editing keys that make it very easy to change a command you typed previously.

The following table describes the additional Doskey editing keys as well as the effect you see when you use some of the standard MS-DOS command editing keys:

To do this	Press these keys
Move the cursor to the beginning of the displayed command.	HOME
Move the cursor to the end of the displayed command.	END
Move the cursor back one character in the displayed command.	LEFT ARROW
Move the cursor forward one character in the displayed command.	RIGHT ARROW
Move the cursor back one word in the displayed command.	CTRL+LEFT ARROW
Move the cursor forward one word in the displayed command.	CTRL+RIGHT ARROW
Delete the character before the cursor.	BACKSPACE
Delete the character above the cursor.	DEL
Delete all characters from the cursor to the end of the line.	CTRL+END
Delete all characters from the cursor to the beginning of the line.	CTRL+HOME
Add characters at the position of the cursor.	INS
Erase the displayed command.	ESC

Most of these keys function exactly as you might expect. The editing keys affect only the command that is displayed. They do not change any of the commands that Doskey has already stored.

If you hold down the CTRL key while you press the RIGHT ARROW or LEFT ARROW key, the cursor moves to the beginning of the next or previous word. A word in this case is a group of characters that are separated from other characters by a space. For example, the following command has three words:

```
copy c:\games\suzz.exe a:_
```

If the cursor is at the end of the line, as is shown in the example, you can move it to the C in C:\GAMES\SUZZ.EXE by pressing CTRL+LEFT ARROW twice.

With the cursor anywhere in the word C:\GAMES\SUZZ.EXE you can move the cursor to the last letter by pressing CTRL+RIGHT ARROW. If you press CTRL+RIGHT ARROW again, the cursor moves to the end of the line.

By pressing the INS key, you can add characters at the position of the cursor. If you don't use the INS key, the new characters you type will overtype any characters that follow the cursor. After you press INS the characters that follow the cursor are moved right as you type. For example, suppose you have the following line displayed:

```
copy c:\games\suzz.exe a:
```

The cursor is under the S in SUZZ.EXE. To change the line so that C:\GAMES\SUZZ.EXE becomes C:\GAMES\NEW\SUZZ.EXE you press the INS key and type NEW\. The line now appears like this:

```
copy c:\games\new\suzz.exe a:
```

To turn off insert mode, pressthe INS key again. The characters you type will once again overtype any characters in front of the cursor. Insert mode is automatically turned off when you press ENTER to run a command.

Saving and Clearing the List of Stored Commands

Anytime you want to delete the whole list of stored commands and begin a new list, you can press ALT+F7. The list of commands is also cleared when you reinstall Doskey or reset your computer.

To save the list in a file, you can enter the **doskey** command with the **/dhist** switch, the output redirection character (>), and the name of the file where you want the list stored. For example, to store your list of commands in the SAV-COMMS.TXT file, enter the following command:

```
doskey /dhist >savcomms.txt
```

This is particularly useful for creating a batch file. To create a batch file using Doskey, press ALT+F7 to clear the list of commands from Doskey, enter the commands you want to save, and then use the **/dhist** switch to save the commands in a file with a .BAT extension. For more information about batch files, see "XX".

Using Doskey to Create Macros

A macro is a set of commands that you can quickly run by typing the name of the macro. Macros are very much like batch files. Both contain sets of commands that you run by typing a name. However, macros differ from batch files in the following ways:

- Macros are stored in RAM, while batch files are stored on disk. Because macros are stored in RAM, they run much faster than batch files. Unlike batch files, you can run a macro from any directory without worrying about where the macro is located. On the other hand, when you reset or restart your computer, the macros you have in RAM are lost, while the batch files on disk are unharmed.
- You create a batch file by saving commands in a file. You create a macro by putting commands inside a macro definition. In a batch file, you separate commands by putting them on separate lines. There is no limit to the number of commands you can put in a single batch file. In macros, you type all the commands on the same line and separate them with special characters. The total length of a macro cannot exceed 127 characters.
- You can stop a batch file by pressing CTRL+C once. In a macro, you must press CTRL+C repeatedly. Each time you press CTRL+C in a macro, MS-DOS skips one command. To stop the entire macro, you must press CTRL+C once for every command left in the macro.
- In both batch files and macros you can use replaceable parameters. However, in batch files the parameters are **%0** through **%9**, and in macros they are **$0** through **$9**. In addition, the redirection characters you use in macros are different from those you use in batch files and MS-DOS commands.
- In batch files, you can use **goto** commands to jump to another place in the file. In addition, you can start other batch files from inside a batch file. You cannot use **goto** commands in a macro or start other macros inside a macro.
- You can run a batch file from inside a macro, but you can't run a macro from inside a batch file (however, you can include a command that *creates* a macro inside a batch file).

- You can't use the **echo off** command inside a macro to stop the commands from being displayed as MS-DOS runs them.

Macros work well *inside* batch files. Because you have to run a macro definition command to load a macro into RAM, the most useful way to use macros is to put the definition commands for the macros that you commonly use inside a batch file. Then, to make the macros available, you run the batch file.

Creating Macros

To create a macro, you enter a **doskey** command that includes the name of the macro and the commands that are in the macro. For example, the following command creates a macro called DDIR that displays a directory in wide format:

```
doskey ddir=dir /w
```

When you type DDIR at the command prompt, MS-DOS runs the macro, displaying a five-column list of the files in the current directory. Because the macro is stored in RAM, it doesn't matter which directory is current when you run it.

To include more than one command in a macro, separate the commands with a dollar sign ($) and the letter T. For example, the following command creates a macro called CMP that alphabetizes and lists the .DOC files and then the .BAK files in the current directory:

```
doskey cmp=dir *.doc /o:n $t dir *.bak /o:n
```

While you are creating and testing a macro, it is easiest to enter the command that defines the macro at the command prompt. Then, you can use the Doskey command editing keys to quickly change and redefine the macro.

Because the macros you define are stored in RAM, when you turn your computer off or reset it, the macros you defined are lost. Thus, if you create a macro that you will commonly use, instead of entering the command to define the macro each time you start or reset your computer, you can put the command that defines the macro in your AUTOEXEC.BAT file to automatically define the macro each time MS-DOS starts.

Running Macros

To run a macro, you enter its name at the command prompt. For example, to run the DDIR macro, enter the following command at the command prompt:

```
ddir
```

If the macro takes parameters, separate them from the name of the macro with a space. For example, suppose you create a macro called MOVE that needs to know which file you want to move and the name of the directory to which you want to move the file. To move all *.TXT files from the current directory to the C:\TFILES directory, you would enter a command like this:

```
move *.txt c:\tfiles
```

You cannot run a macro from inside a another macro or from inside a batch file.

When you enter the name of a macro, there must be no space between the command prompt and the macro name. Otherwise, MS-DOS will not recognize the name and will display a "Bad command or file name" message.

You can use this requirement to your advantage if you want to create a macro that has the same name as a command. For example, suppose you generally like your directories displayed in five-column format. you can use the following command to create a macro called DIR that replaces the MS-DOS **dir** command:

```
doskey dir=dir /w
```

When you have a macro with the same name as a command, MS-DOS will always run the macro rather than the command. So when you enter DIR at the command prompt, MS-DOS will always run the DIR macro rather than the **dir** command.

Whenever you want to use the **dir** command instead of the DIR macro, you can type a space between the command prompt and the word DIR. Now, MS-DOS does not recognize DIR as a macro name, but it does recognize it as a command. If you have a macro and command with the same name, you can type one or two spaces before the name to run the command instead of the macro.

Displaying Macros

To display the names and contents of the macros you have created, press F10. Doskey displays a list of the macros you have created. The redirection characters are displayed as their regular characters. Command separators are displayed as paragraph marks.

Editing Macros

To edit a macro, edit the command that created the macro and run it again. If the macro is inside a batch file, you can edit the command and then run the batch file again. If the macro command is in the list of commands that Doskey saves, you can redisplay the command, edit it with the Doskey command editing keys and then press ENTER to create the new macro. For information about the Doskey command editing keys, see xx, earlier in this chapter.

Saving Macros

There are two ways to save macros that you created at the command prompt: you can save the entire command that created the macro or you can save just the macro's name and contents. The first method uses the **/hist** switch and saves all the commands in the Doskey list. The second uses the /dmacs switch and saves all of the macros currently stored by Doskey in memory.

If the commands that created the macros are in your Doskey list, you can save them by using the **doskey** command with the **/hist** switch, the greater-than sign (>), and a filename. For example, the following command saves your Doskey list of commands in the COMLIST.TXT file:

```
doskey /dhist >comlist.txt
```

Once the commands are saved, you can run them again in a batch file. For example, suppose you want to run three of the macro commands stored in COMLIST.TXT. You could use a text editor to move the three commands into a batch file. Then whenever you want to use the macros, you run the batch file. If you develop a set of macros that you use often, you could put the commands that create them in your AUTOEXEC.BAT file so that they are run each time you restart or reset your computer.

To save the content of the macros stored in memory, use the **doskey** command with the **/dmacs** switch, a greater-than sign (>), and a filename. This also translates some of the special macro characters into regular MS-DOS characters. For example, the following command stores the name and content of the macros that are currently in RAM in the MACS.TXT file:

```
doskey /dmacs > macs.txt
```

Once the macros are in the file, you can easily turn one or more of them into a batch file by putting each command on its own line and changing the replaceable parameters to match the batch file format.

Deleting Macros

To delete a macro, enter a **doskey** command with the name of the macro you want to delete. For example, to delete the DDIR macro, you would enter the following command:

```
doskey ddir=
```

Doskey removes the macro from memory. To delete all macros, press ALT+F10.

Using Replaceable Parameters

You can use replaceable parameters in your macros in much the same way you use them in batch files. In macros, the replaceable parameters are called **$0** through **$9** rather than %0 through %9.

For example, the following command creates a macro called FINDIT that searches through the directories on drive C for files that match the one you specify:

```
doskey findit=dir c:\$1 /s
```

To run this macro, enter FINDIT and a filename at the command prompt. For example, to locate all files on drive C that have the extension .OLD, you would enter the following command:

```
findit *.old
```

Doskey substitutes the filename you type for the **$1** in the macro. The resulting command looks like this:

```
dir c:\*.old /s
```

The /s switch tells MS-DOS to display files from all the directories on drive C (including the current directory) that match *.OLD.

You can use the same parameter more than once in a macro. For example, the following command creates a macro called DDEL that moves a file to a directory called DELETED rather than actually deleting it:

```
doskey ddel=copy $1 c:\deleted $t del $1
```

When you start the DDEL macro, you also type the name of the file you want to delete. Doskey replaces the $1 in the macro with the filename. It copies the file to another directory and then deletes it from its original directory. This gives you a second chance to recover the file if you discover later that you really needed it.

Periodically, you can clean out the C:\DELETED directory with this macro:

```
doskey cleanup=dir c:\deleted $t del c:\deleted\*.*
```

The macro displays the DELETED directory and then starts the command that will delete all the files in the directory. Because the **del** command automatically prompts you to confirm that you want to delete all the files, you have a chance to review the list of files and decide whether you want to delete them.

Using the $0 Replaceable Parameter

The **$0** parameter allows you to assign all of the text following the command that starts a macro to a single parameter. Normally, MS-DOS distinguishes parameters by looking for a space. It assumes that the text between the first two spaces is the first parameter, the text between the second and third spaces is the second parameter, and so on. If you use the **$0** parameter, Doskey ignores spaces and assigns all of the text to the **$0** parameter.

The **$0** parameter is most useful when the macro you create should function with a variable number of parameters. For example, you can use the following command to create macro called D that allows you to abbreviate the **dir** command:

```
doskey d=dir $0
```

This macro will work exactly like the **dir** command regardless of the number of parameters you specify. For example, all the following commands will run the same with the D macro as they do with the **dir** command:

```
d *.txt
d *.txt /s
d *.txt /s /b
```

If you used any parameter other than **$0** with the macro, MS-DOS would assign *.TXT to the parameter and ignore the rest of the command line.

Redirecting Input and Output

You redirect input and output in macros the same way you redirect it in MS-DOS commands. The only difference is that macros require different characters:

- **$L** (or $l) is equivalent to the less-than sign (<). It redirects the input to a command.
- **$G** (or $g) is equivalent to the greater-than sign (>). It redirects the output of a command.
- **GG** (or gg) is equivalent to the double greater-than sign. It appends output onto the end of a file.
- **$B** (or **$b**) is equivalent to the pipe character (|). It redirects output from one command to another.

For example, the following command creates a macro called PDIR that prints your directory listings:

```
doskey pdir=dir $g Lptl:
```

The following command creates a macro called MTYPE that displays the contents of the file you specify and pauses between each screen of information:

```
doskey mtype=type $1 $b more
```

The following command creates a macro called ASORT that alphabetizes the file you specify and stores it in a different file:

```
doskey asort=sort $L $1 $g $2
```

When you start this macro, you type the name of the file you want to sort. Doskey replaces the **$1** replaceable parameter with the first filename you enter. The **$L** redirects the file into the **sort** command. The **$g $2** redirects the output of the sort command into the second filename you enter.

Because the dollar sign ($) is such a significant character in macros, you have to mark it when it occurs anywhere other than in parameters, command separators, and redirection characters. You mark extra dollar signs by typing two of them rather than one. For example, suppose your macro copies a file to the $&CENTS directory. When you type the name of the directory you must type $$&CENTS. That way when Doskey runs the command, it knows to take the dollar sign as a regular character rather than a marker or parameter.

Chapter 9

The Shell Program Manager

The Shell Program Manager is designed to arrange and run programs. It enables you to display the programs you use, add and delete programs, subdivide programs into smaller groups, and run a program with the click of a mouse or a few keystrokes.

You can organize your programs into groups. Groups are collections of programs somewhat like directories. They allow you to gather your programs so you can easily access them. For example, you might put three different financial programs into a single group, so you can work easily with any of them. When you add a group, you give it a title, which appears on the File Manager screen. To work with the group you choose its title.

Before you can run a program with the Program Manager, you must add it to a group. When you add a program to a group, you give it a title and type the command that starts the program. If you want, you can create a dialog box that prompts you for information each time you start the program. To run the program, you choose its title from the group list it appears on.

Working with Groups

Groups are similar to directories in that they allow you to gather together similar files. Groups are a way to organize your programs without worrying about where they are stored. The files in a group can be stored in any directory on any of your computer's disks. The programs themselves are not moved or in any way changed when you add them to a group. All commands for managing groups are in the Group menu of the Program Manager screen.

When you first start the Shell, the Program Manager screen displays the Main group. The Main group contains three programs: DOS Command Prompt, File Manager, and Change Colors. The Main group also contains any other groups and programs that you have added to it.

This artwork is not available
for this release.

spm_1

The Program Manager's Main group is like the root directory of a disk. Every other group, including the DOS Utilities group, is a subgroup of the Main group. You can create a group as a subgroup of the Main group or as a subgroup of another group. For example, you can create two subgroups of the DOS Utilities Group; one that contains file utilities, and one that contains directory utilities. The following figure illustrates this relationship:

This artwork is not available
for this release.

spm_2

A group can contain any number of other groups and programs. The titles of the programs and subgroups in a group are displayed on the *group list*. If the Shell is in text mode, groups are shown with three dots (. . .) after their names to distinguish them from programs. If the Shell is in graphics mode, groups are labeled with a group icon.

To work with a group, choose the group's title. To move backward from a subgroup to a group, press ESC.

In addition to a title, you can assign each group a password and a help message:

- The group help message contains information about using the group and its programs. When you select a group title, you can display the group's help message by choosing xx from the Help menu or pressing F1. MS-DOS displays help messages in a dialog box.

- The group password restricts access to the group. If a group has a password, MS-DOS displays a dialog box in which you must enter the password before you can display or change the group.

NOTE If you decide to assign a password to a program, be sure to write it down and keep it in a safe place. There is no way to display a password once it has been assigned.

Adding a Group

You can add groups to the Main group, the DOS Utilities group, or a group you have created. For example, you might add a group called ACCOUNT if you have three different programs that you use to keep track of your finances (such as a program to maintain your checkbook, a program to estimate taxes, and a program to track your monthly bills). By putting the three programs in a group, you can move between them efficiently.

When you create a group, you give it a title. If you want, you can also give it a password that a user must know to display the group, and a help message that provides information about the group and its programs.

▶ To add a group to an existing group:

1. Display the group to which you want to add the new group.
2. Choose Add from the Group menu.
   MS-DOS displays the Add Group dialog box.

This artwork is not available for this release.

spm_3

3. Type a title for the new group.

   The title can be up to 74 characters long.

4. If you want the group to have a help message, type up to 256 characters (including blanks) in the Help text box. For example, you might type a help message that says, "Use the programs in this group to perform statistical operations." MS-DOS displays the message exactly as you type it and formats it to fit in the Help dialog box. If you want to begin a new line, type an ampersand (&) at the point where you want the new line to start.

   If you don't want the group to have a help message, leave the box blank.

5. If you want the group to have a password, type it in the Password box. Passwords must be less than 256 characters long.

   If you don't want to assign a password, leave the box blank.

6. Choose OK to add the group and return to the Program Manager screen.

Changing a Group

You can change the title, password, or Help message of any group except the Main group.

▶ **To change group information:**

1. Select the group title.
2. Choose Change from the Group menu.

   If the group has a password, MS-DOS displays the Password dialog box. Type the password and choose OK.

   MS-DOS displays the Change Group dialog box.

This artwork is not available
for this release.

spm_4

3. Change whatever information you want.
4. Choose OK to save the changes and return to the Program Manager screen.

   You cannot save the changes if the Title box is not filled in.

Deleting a Group

When you delete a group, the group name is removed from the list it appeared on and the group's password and help message are deleted. The programs in the group remain in the directories where they are stored, but they are no longer accessible from this group of the Shell.

▶ **To delete a group:**

1. Select the title of the group you want to delete.
2. Choose Delete from the Group menu.

   If the group has a password, MS-DOS displays the Password dialog box. Type the password and choose OK.

   MS-DOS displays the Delete Item dialog box.

This artwork is not available for this release.

spm_5

3. Choose option 1 to delete the group and return to the Program Manager screen.

   Or choose option 2 to return to the Program Manager screen without deleting the group.

Reordering a Group List

To move a program or group title from one position in a group list to another, use the Reorder command.

▶ **To reposition a title:**

1. Select the title you want to reposition.
2. Choose Reorder from the Group menu.
3. Move the selection cursor to the new location and press ENTER.

   MS-DOS repositions the title and returns you to the Program Manager screen.

Working with Programs

You can use Program Manager to make programs easy to start. By adding your programs (including MS-DOS commands and batch files) to a group, you can avoid having to remember where they are stored, and what special commands or parameters you need to start them. To run a program from a group list, you choose its title. The Program menu contains the Shell commands that manage programs. For more information about running programs from the Shell, see xx.

Adding a Program to a Group

When you add a program to a group, you give it a title and type the command you need to start the program. Along with the command, you can specify MS-DOS commands or batch files to run before or after the program, and you can have MS-DOS display a custom dialog box.

When you add a program, you can also specify the following information:

- A directory that MS-DOS should make current before starting the program.
- A help message that presents information about the program and how to run it. Help messages appear when you select a program from a group list and choose xx or press F1.
- A password to restrict access to the program. If a program has a password, you must enter it before you can run the program or change any of the information you assigned to it.
- An option that tells MS-DOS to pause before returning to the Shell when the program finishes.

NOTE If you decide to assign a password to a program, be sure to write it down and keep it in a safe place. There is no way to display a password once it has been assigned.

You can add programs to the Main group, the DOS Utilities group, or a group you have created.

▶ To add a program to a group:

1. Display the group list you want to add the program to.
2. Choose Add from the Program menu.
   MS-DOS displays the Add Program dialog box.

This artwork is not available
for this release.

spm_6

3. Type the name that you want to appear on group lists in the Title box.

   Titles can be 74 characters long, including blank spaces.

4. Type the command that starts the program in the Commands box.

   The command you type in this box is called the startup command. Before you can add the program to a group, the startup command must contain at least a single command to start the program.

   You can assume that the command you type in the Commands box starts from the Startup directory you specify.

   In addition to the command that starts the program, you can add other MS-DOS commands to the startup command and you can create a custom dialog box that prompts you for parameters. For more information, see xx.

5. If you want MS-DOS to change the current directory before it starts the program, type the name of the directory you want to change to in the Startup directory box.

   This option is useful if the program you are running assumes that a certain directory is current.

6. If you want the program to have a help message, type up to 256 characters (including blanks) in the Help Text box.

   MS-DOS displays the message exactly as you type it and formats it to fit in a Help dialog box. If you want to begin a new line, type an ampersand (&) at the point where you want the new line to start.

   If you don't want the program to have a help message, leave the box blank.

7. If you want the program to have a password, type it in the Password box. Passwords must be less than 256 characters long.

   If you don't want to assign a password, leave the box blank.

8. If you want MS-DOS to pause and prompt you to press a key before returning to the File Manager screen, select the Pause After Exit option.

   This option is useful if you want to read any text that your program displays after it finishes running.

9. Choose OK to add the program to the group you selected.

   If there are any replaceable parameters in your startup command, MS-DOS displays a dialog box for each parameter. Choose OK to accept the information that is displayed, or edit the information in these boxes and then choose OK.

Changing a Program

After you add a program to a group, you can change the name of the program, the startup command, the help text, or the program's password.

▶ **To change program information:**

1. Select the title of the program you want to change.
2. Choose Change from the Program menu.

   If the group has a password, MS-DOS displays the Password dialog box. Type the password and choose OK.

   MS-DOS displays the Change Program dialog box.

This artwork is not available
for this release.

spm_7

3. Make the changes you want to the information in the box then choose OK to save the changes.

   MS-DOS will not save the changes if the title box is not filled in, or if you press ESC. If there are any replaceable parameters in your startup command, MS-DOS displays a dialog box for each parameter. Choose OK to accept the information that is displayed, or edit the information in these boxes and then choose OK.

Removing a Program from a Group

If you remove a program from a group, MS-DOS no longer displays its title, and the information you entered for the program is deleted. Removing a program from a group does not delete the program from the directory that contains it.

▶ **To delete a program title from a group:**

1. Select the title of the program you want to delete.
2. Choose Delete from the Program menu.

   If the group has a password, MS-DOS displays the Password dialog box. Type the password and choose Ok.

   MS-DOS displays the Delete Item dialog box.

This artwork is not available
for this release.

spm_8

3. Choose option 1 to remove the program from the group and return to the Program Manager screen.

   Or choose option 2 to return to the Program Manager screen without removing the program.

Copying a Program

To move a program from one group to another or to have a program in more than one group, you use the Copy command. For example, if you use Microsoft Excel with your accounting programs and with your tax programs, you might copy the program to your Accounting group and your Tax group. You can copy a program to as many groups as you like.

▶ To copy a program from one group to another:

1. Select the title of the program you want to copy.
2. Choose Copy from the Program menu.

   If the group has a password, MS-DOS displays the Password dialog box. Type the password and choose OK.
3. Display the group list of the group you want to copy the program to.
4. Choose OK to copy the program and return to the Program Manager screen.

   Or choose Cancel to return to the Program Manager screen without copying the program.

Creating Startup Commands for Your Programs

When you add a program to a group on the Program Manager screen, you type a startup command in the Commands box of the Add Program dialog box. A startup command contains the command you use to start a program plus any other commands you want to run before and after your program. The startup command runs just like a batch file. MS-DOS runs each command in sequence. When all of the commands in the startup command are finished, MS-DOS returns you to the Program Manager screen.

For example, suppose you want to add the program C:\EDIT\WORD.EXE to the Editor group. First, you display the Editor group and choose Add from the Program menu. Then you give the program a name and type the following command in the Commands box:

```
c:\edit\word
```

The command includes the pathname of the file as well as its name, so that MS-DOS can find the file.

The startup command can contain any number of valid MS-DOS commands. The maximum length of the text in the Commands box is 500 characters. To include more than one MS-DOS command in a startup command, you separate the commands with semicolons.

For example, to change the current directory to the root of drive C after you exit from WORD, but before MS-DOS returns you to the Program Manager screen, you would include a **cd** command in the startup command as in the following example:

```
c:\edit\word ; cd \
```

You cannot include spaces, but you can include one or more spaces before and after the semicolons. MS-DOS runs the commands you type in sequence. In this case, it runs WORD.EXE first. When WORD.EXE ends, MS-DOS runs the **cd** command. When the **cd** command ends, MS-DOS returns you to the Shell Program Manager screen.

Using the Startup Directory Box

Sometimes it is preferable to change the current drive and directory before you start a program. To accomplish this task in a startup command, you would have to type three commands: one to change the drive, one to change the directory, and one to start the program. Alternatively, you can set the directory by typing the directory in the Startup directory box and the startup command in the Commands box as in the following example:

This artwork is not available
for this release.

spm_9

Running Batch Files in a Startup Command

You can run batch files by including call commands in the startup command. For example, to run the PREP.BAT batch file before you enter WORD, and the POST.BAT batch file after you exit the program, enter the following startup command:

```
call prep ; word ; call post
```

For more information about batch files, see Chapter 11, "Working with Batch Files".

Using Parameters

If the programs you add to a group take parameters, MS-DOS displays a dialog box that prompts you for them when you start the program. For example, suppose you often start Word by changing to its directory and then typing a filename along with the command that starts the program as in the following example:

```
c:\edit\word turn.doc
```

By adding a replaceable parameter to your startup command, you can have MS-DOS display a dialog box that prompts you for the filename. Replaceable parameters are the symbols **%0** through **%9**. You use them to stand in for the parameter that you enter in the dialog box. For example, if you want MS-DOS to prompt you for a filename when you start C:\WORD\WORD.EXE, you could use the following startup command:

```
c:\edit\word %1
```

The **%1** substitutes for the filename you normally enter when you run the program. The replaceable parameter tells MS-DOS to display a dialog box and use what you type in the dialog box as the filename of the file you want to edit.

When you enter a startup command that includes a replaceable parameter, MS-DOS displays the following dialog box where you can enter a title, an informational message, and a prompt for the parameter's dialog box:

This artwork is not available for this release.

spm_10

Type the information that you want MS-DOS to display in the parameter's dialog box. If you don't type any information, MS-DOS displays this generic dialog box when you start the program:

This artwork is not available
for this release.

spm_11

Using the Same Parameter Twice

You can use a single replaceable parameter more than once in a startup command. For example, suppose you always back up new Word files to the A drive after you have created them. To start Word with URN.DOC and then back up the file after you have edited it, you would enter the following commands:

```
c:\edit\word patko.doc
copy patko.doc a:
```

To make a single startup command that performs these tasks, you can use the %1 replaceable parameter in two places as in the following example:

```
c:\edit\word %1 ; copy %1 a:
```

Using More Than One Parameter

You can have as many dialog boxes in a startup command as you like. For each dialog box, you use a different replaceable parameter. For example, suppose you want MS-DOS to prompt you for a file to edit with Word and for a backup directory when you are finished editing. You would include two replaceable parameters in the startup command as in the following example:

```
c:\edit\word %1 ; copy %1 %2
```

When MS-DOS runs the command to start Word, it displays a dialog box to prompt you for a file to edit (%1). After you exit from Word, MS-DOS runs the **copy** command, which tells it to display another dialog box to prompt you for a backup directory (%2).

When you enter a startup command with two different parameters, MS-DOS first displays a dialog box where you can type information for the first parameter's dialog box. When you exit this dialog box, MS-DOS displays a second dialog box where you enter information for the second parameter's dialog box.

Chapter 10

Working with the MS-DOS Editor

10

Edit is an MS-DOS text editor that is useful for creating memos, letters, and special files that customize MS-DOS. You can use Edit to create and edit text, save the text in a file, modify existing files, and print files.

Files you create with Edit are unformatted text files. Since MS-DOS batch files and files such as AUTOEXEC.BAT and CONFIG.SYS must be unformatted text files, Edit is a particularly useful tool for customizing your system.

You can use a keyboard or a mouse to work with Edit.

Those who have used the Edlin line editor in previous versions of MS-DOS will still find it in MS-DOS 5.0. For information about specific Edlin commands, see the alphabetical command listing in part I of the *MS-DOS 5.0 User's Reference.*

When you use Edit, you can:

- Choose commands from menus and specify information and preferences in dialog boxes.
- Move around in a text file using cursor movement keys, view the text of a file at all times, and see changes as you make them.
- Select text in a file and move, copy, or delete it.
- Use Edit's extensive on-screen help system to find information about Edit techniques and commands.

Getting Started with Edit

You start Edit by entering the **edit** command at the command prompt. If you want to open a file in Edit when you start it, you can include the filename in the command. For example, if you want to open the file named PROGRAM.TXT, you enter:

```
edit program.txt
```

When you enter the **edit** command alone, the following screen appears:

This artwork is not available for this release.

edt_1

To get information about Edit, press ENTER when this first screen appears, or click the area between the angle brackets that reads "Press Enter to see the Edit Survival Guide." This is an introduction to Edit and the Edit help system.

To start working with Edit, press ESC or click the area between the angle brackets that reads "Press Esc to clear this dialog box."

You see an Edit screen with a blank Edit window. You create a text file by typing text in the window. If you specify a filename when you start Edit, this screen appears immediately.

This artwork is not available for this release.

edt_2

Working with Menus

To perform tasks in Edit, you choose from lists of commands called *menus*. The names of the Edit menus are displayed at the top of the Edit screen. Before you choose a command, select the menu that has the command you want.

▶ **To select a menu:**

Mouse ■ Click the name of the menu you want to select.

Keyboard 1. Press ALT.

The first menu name, File, is highlighted.

2. Press the LEFT and RIGHT ARROW keys to move the selection cursor to the menu you want to select, and press ENTER.

   Or press the first letter of the menu name.

▶ To choose a command:

Mouse ■ Click the command you want to choose.

Keyboard ■ Press the UP and DOWN ARROW keys to select the command you want to choose, and press ENTER.

Or press the highlighted letter in the command name. If a command does not have a highlighted letter, you need to select text, open a file, or perform some other action before you can use it.

You can also choose some commands using keyboard shortcuts. If a command has a keyboard shortcut, it is displayed next to the command in the menu.

▶ To cancel a selected menu without choosing a command:

■ Click anywhere outside the selected menu

Or press ESC.

Working with Dialog Boxes

Some commands require you to give Edit more information. When you choose these commands, a *dialog box* appears. You enter additional information or choose options in the dialog box.

For example, when you choose the Open command from the File menu, a dialog box appears:

This artwork is not available for this release.

edt_3

Most dialog boxes have several areas in them, in which you can enter information or select options. You can move from one area of a dialog box to another by pressing the TAB key, or by clicking the area you want to use. You can also press the ALT key along with the highlighted letter of the area's name.

There are several types of areas in a dialog box:

- Text boxes in which you can enter text. If the box already contains text, you can clear it by typing any character or pressing the SPACEBAR.
- A series of options, with a bullet next to the currently selected option. You choose an option by clicking between the parentheses next to the option you want, or by moving the bullet with the ARROW keys.

- A check box that you can select or clear. To select a check box, click the check box, or move the cursor to the check box with the TAB key and press the SPACEBAR.
- A list of files or drives and directories.

▶ To select an item from a list in a dialog box:

Mouse ■ Click the item you want.

If the item you want does not appear, click the scroll arrows until the item you want comes into view. You can also drag the small box that is in the scroll bar, or click the shaded areas of the scroll bar.

Keyboard ■ Use the ARROW keys to move the selection cursor to the item you want.

You can use HOME, END, PAGE UP, and PAGE DOWN to move the selection cursor more quickly.

Dialog boxes have three *buttons* at the bottom. The OK button performs the command you chose. The Cancel button cancels the command. The Help button displays information about the command.

You can choose a button in a dialog box by clicking it with the mouse, or by moving the selection cursor to it with the TAB key and pressing ENTER. You can also cancel a dialog box by pressing the ESC key, and get help by pressing F1. When the angle brackets around the OK button are highlighted, you can press ENTER to perform the command.

Getting Help

Edit's online help enables you to display information about:

- Edit menus and commands
- Getting started with Edit
- Using a keyboard with Edit

When you select an Edit command, the line at the bottom of the screen displays information about the command. For further help, select the menu or command you want to know more about, and press F1. Edit displays a help window that describes the menu or command and what it does. You can also display a help window by choosing the Help button at the bottom of each Edit dialog box. To clear help from the screen, press ESC.

You can display information about getting started with Edit, and about the Edit keyboard, by using the Edit help system. The Edit help system contains information about editing keys, cursor movement, dialog boxes, selecting text, opening and saving files, and other basic information. You can enter the Edit help system in several ways:

- By pressing ENTER at the Edit startup screen
- By pressing F1 at the Edit window when no commands or menus are selected
- By choosing Getting Started or Keyboard from the Help menu

When one of the help windows appears, you can choose any of the related topics enclosed in triangular brackets. To display information, you move the cursor to one of the bracketed topics and press ENTER, or double-click the topic with the mouse.

When you choose a topic, the help system displays a window with information about that subject. Each screen contains additional topics that you can choose to get more information. When you are through using the help system, press ESC to return to the Edit window.

You can use these keys to move the cursor around from one topic to the next, or from one help window to the next:

To do this	Use this key
Move the cursor to next topic.	TAB
Move the cursor to previous topic.	SHIFT+TAB
Move the cursor to next topic starting with that character.	character
Move the cursor to previous topic starting with that character.	SHIFT+character
Look at help window for the previous topic (repeat up to 20 times).	ALT+F1
Look at the next help topic stored in the Help file.	CTRL+F1
Look at the previous help topic stored in the Help file.	SHIFT+CTRL+F1

Exiting Edit

When you have finished working with Edit, choose Exit from the File menu. You return to the command prompt.

If you are working with a file that has been changed but not been saved, Edit asks you if you want to save the changes. For more information about saving files, see "XX" later in this chapter.

Creating a Text File

You create a text file by entering text on the Edit screen. When you reach the end of a line, you must press ENTER to tell Edit to move to the next line. Edit does not wrap words to the next line automatically. Lines in Edit can be up to 256 characters long.

The following table shows keys that make entering text easier. These keys are similar to the keys you use in word processors such as Microsoft Word and WordStar ®.

To do this	Use this key
Delete a character to the left of the cursor.	BACKSPACE or CTRL+H
Delete the character at the cursor.	DEL or CTRL+G
Delete the word at the cursor.	CTRL+T
Overwrite existing characters (press INS or CTRL+V again to stop overwriting).	INS or CTRL+V

Moving the Cursor

You can use the mouse or the keyboard to move the cursor around existing lines of text. You can also scroll the text to see parts of a file that are not displayed in the Edit window.

To move the cursor using the mouse, click the place on the screen where you want the cursor to be. If you are working with a file that is longer or wider than the Edit screen, you can use the scroll bars at the right and bottom of the screen to scroll text into view. To scroll text, point to one of the arrows on either end of the scroll bars and press the mouse button, drag the small box in the scroll bar, or click the shaded areas of the scroll bar.

The following keys allow you to move the cursor around in existing lines of text :

To do this	Use this key
Move the cursor one character or one line.	ARROW KEYS
Move the cursor one word left.	CTRL+LEFT ARROW
Move the cursor one word right.	CTRL+RIGHT ARROW
Move the cursor to beginning the of the line.	HOME
Move the cursor to the end of the line.	END
Move the cursor to the beginning of the next line.	CTRL+ENTER
Move the cursor to the top of the screen.	CTRL+Q+E
Move the cursor to the bottom of the screen.	CTRL+Q+X

You can also use the keyboard to scroll the text that is on the screen:

To do this	Use this key
Scroll one line up.	CTRL+UP ARROW or CTRL+W
Scroll one line down.	CTRL+DOWN ARROW or CTRL+Z
Scroll one screen up.	PAGE UP
Scroll one screen down.	PAGE DN
Move to the beginning of the file.	CTRL+HOME or CTRL+Q+R
Move to the end of the file.	CTRL+END or CTRL+Q+C
Scroll one screen left.	CTRL+PAGE UP
Scroll one screen right.	CTRL+PAGE DN

You can also scroll by moving the cursor to the edge of the window and holding down an ARROW key.

Selecting Text

You perform various editing operations by *selecting* a block of text. For example, if you want to delete several lines at once, you can select the lines and press the DEL key. You can select any amount of text, from a single character to an entire file.

▶ **To select text:**

Mouse
1. Point to the first character you want to select.
2. Drag the cursor to the last character you want to select.
3. Release the mouse button.

   To cancel a selection, click anywhere in the Edit window.

Keyboard
1. Use the ARROW keys or other cursor movement keys to move the cursor to the first character you want to select.
2. Hold down SHIFT, and use the ARROW keys to move the cursor to the last character you want to select.
3. Release the keys.

   To cancel a selection, press any cursor movement key.

After the text is selected, you can delete it with the DEL key or edit it using one of the commands in the Edit menu.

Editing Text

The Edit and Search menus list commands you can use to work with the text in a file. Using the commands in the Edit menu, you can move or copy a block of text from one part of a file to another. You do this by transferring text from the file to a storage area called the *clipboard.* You can locate any occurrences of a set of characters in a file or replace the characters you are searching for with a different set.

Moving Text

You can move a block of text using the Cut and Paste commands. This is useful if you want to rearrange the order of the text in a file.

▶ **To move a block of text:**

1. Select the block of text you want to move.
2. Choose Cut from the Edit menu, or press SHIFT+DEL.

   The block of text is removed from the file and placed on the clipboard.

You can delete all the text in the current line by pressing CTRL+Y. You can delete the text from the cursor location to the end of the current line by pressing CTRL+Q+Y.

3. Move the cursor to the place in the file where you want the text to appear.
4. Choose Paste from the Edit menu, or press SHIFT+INS.

   The text on the clipboard is inserted into the file at the cursor location.

When you choose the Paste command, the text is not removed from the clipboard. The text remains on the clipboard until you move or copy another block of text to the clipboard. You can insert the text from the clipboard into the file as many times as you want by repeating the Paste command.

Copying Text

You use the Copy and Paste commands to copy a block of text.

▶ **To copy a block of text:**

1. Select the block of text you want to copy.
2. Choose Copy from the Edit menu, or press CTRL+INS.

   The block of text is copied to the clipboard. It is not removed from its original location.
3. Move the cursor to the place in the file where you want the copy of the text to appear.
4. Choose Paste from the Edit menu, or press SHIFT+INS.

   A copy of the text on the clipboard is inserted into the file at the cursor location.

When you choose the Paste command, the text is not removed from the clipboard. The text remains on the clipboard until you move or copy another block of text to the clipboard. You can insert the text from the clipboard into the file as many times as you want by repeating the Paste command.

TIP To replace a block of text with the text in the clipboard, highlight the block you want to replace, and choose Paste from the Edit menu. The text in the clipboard replaces the highlighted text.

Clearing Text

You can clear text from the Edit window, without copying it to the clipboard, by choosing the Clear command from the Edit menu. You can also use the DEL key to clear text.

▶ **To clear text:**

1. Select the text you want to clear.
2. Choose Clear from the Edit menu.

   Or press DEL.

   The text is cleared from the screen.

Choosing the Clear command has no effect on text that is on the clipboard.

Finding Text

To find text in a file, choose the Find command from the Search menu. The text can be a word, a phrase, or any other combination of characters and spaces. When Edit searches for text, it starts at the current cursor position and highlights the first occurrence of the text.

▶ **To find text:**

1. Choose Find from the Search menu.

   The Find dialog box appears.

   If you select text before you choose the Find command, the selected text appears in the Find What box. Otherwise the word at the current cursor location appears in this box. If you want to search for different text, type the text you are searching for in the Find What box.
2. Select The Match Upper/Lowercase option if you want to match capitalization exactly.

   Otherwise Edit finds, for example, both *Brown* and *brown*.
3. Select the Whole Word option if you want to find only separate occurrences of the search text.

   Otherwise Edit finds, for example, *main* in *remainder*.
4. Choose OK to start the search.

Edit begins searching at the cursor position. The first occurrence of the text is highlighted.

If no occurrences of the text are found, Edit displays a Match Not Found dialog box.

5. To search for the next occurrence of the search text, choose Repeat Last Find from the Search menu (or press F3).

   The Find command continues to search through the file each time you choose Repeat Last Find. When it reaches the end of the file, it returns to the beginning and continues searching.

6. Choose Cancel to close the Find dialog box.

Replacing Text

You can use the Change command to find text and replace it with new text. The Change command begins at the cursor location, and continues to the end of the file. When it reaches the end of the file, it returns to the beginning and continues searching and replacing.

▶ **To replace text:**

1. Position the cursor where you want to start replacing text.
2. Choose Change from the Search menu.

   The Change dialog box appears.
3. In the Find What box, type the text you want to replace, and in the Change To box, type the text you want to replace it with.
4. Select the Match Upper/Lowercase option if you want to match capitalization exactly.
5. Select the Whole Word option if you want to replace only separate occurrences of the search text.
6. Choose either the Find and Verify option or the Change All option to start searching and replacing.

   If you choose the Find and Verify option, each occurrence of the original string is highlighted, and a dialog box appears that asks if you want to make the change, skip this occurrence, or cancel the command. If you choose the Change All button, Edit replaces all occurrences of the search text.
7. Choose OK (or press ESC) when Edit displays the Change Complete dialog box.

Managing Files

Use the commands in the File menu to open an existing file, start working on a new file, and save or print the file you are working with.

Creating a New File

After you finish working on a file, you can create a new file by choosing the New command from the File menu.

▶ **To create a new file:**

1. Choose New from the File menu.

   If you have made changes to the file you are working on but have not saved it, Edit displays a dialog box asking you if you want to save the changes. Choose Yes to save the changes, or No to close the file without saving the changes.

   Edit closes the open file and displays an empty Edit window labeled "Untitled." You can begin typing text in this window.
2. When you want to save the new file, choose the Save As command from the File menu.

When you start Edit by entering the **edit** command without specifying a filename, Edit assumes that you want to create a new file, and displays an empty "Untitled" window.

Opening a File

You can open several types of files using the Open command in the File menu:

- Files you created previously with Edit
- Other unformatted text files (including files such as AUTOEXEC.BAT and CONFIG.SYS)
- Formatted text files created with a word processor. These files will include some formatting characters, and they will lose their formatting if you save them using the Save As command.

▶ **To open a file:**

1. Choose Open from the File menu.

The Open dialog box appears.

2. Type the name of the file you want to open, or select the filename in the file list.

   If the file you want is not on the current drive or directory, you can type the pathname as part of the filename.

   To list the files in a different drive or directory, move to the Drives/Dirs list, select the drive or directory you want to list, and press ENTER. Depending on your directory structure, you might need to move up or down through several levels of subdirectories to get to the directory you want. The currently listed drive and directory are shown on the left side of the dialog box, below the words File Name.

3. When the file you want to open is displayed in the File Name box, choose OK.

   If you have made changes to the file you are working on when you choose the Open command, Edit displays a dialog box asking you if you want to save the changes. If you want to save the changes, choose Yes. If not, choose No. If you decide not to open the new file, choose Cancel.

Saving a File

After you have created a new file, or made changes to an existing file, you can save it on a disk by choosing the Save As command from the File menu. It is a good idea to save the file you are working on periodically, in case you experience a power loss or some other equipment failure.

▶ To save a file:

1. Choose Save As from the File menu.

2. When the Save As dialog box appears, enter a name for the file and choose OK.

   If you want to save the updated version of the file without changing its name, choose OK without entering a new name.

   If you want to save the file on a different drive or in a different directory, select the drive or directory you want in the Drives/Dirs list, or include a pathname when you enter the filename.

   If you try to save a file in a directory that already has a file of the same name, Edit displays a dialog box asking if you want to overwrite the existing file. Choose Yes to overwrite the file. Choose No to cancel the Save As command.

If you save a file that has not been given a name, Edit saves the file with the name UNTITLED. If there is already an existing UNTITLED file, the new file overwrites the old one.

TIP You can use the Save As command to save a modified version of a file without losing the original version. For example, if you have a file named MEMO, you can keep this file and save the modified version as MEMO_A.

Printing a File

You can use the Print command in the File menu to print part or all of the open file. This command works only if you have a printer attached to the LPT1 (parallel) printer port.

▶ **To print a file:**

1. Be sure the file you want to print is open. If you want to print a portion of a file, select the text you want to print.
2. Choose Print from the File menu.

   The Print dialog box appears.
3. If you want to print only the text you selected before you chose the command, select the Selected Text option. Otherwise, select the Entire Document option.
4. Choose OK.

Customizing Edit

You can use the commands in the Options menu to set the screen display and help options for Edit.

Controlling the Screen Display

Use the Display command in the Options menu to change the colors or shades in the Edit window, display or hide scroll bars, and set the number of tab stops.

▶ **To change the screen display:**

1. Choose Display from the Options menu.

   The Display dialog box appears.

2. Select a foreground color and a background color. The screen colors that are listed depend on the type of display you have. The words "Set colors for the text editor window" display a sample of the currently selected colors or shades.
3. Select the Scroll Bars option if you want scroll bars to appear in the Edit window. If you want to see more of the Edit window, clear the check box to hide the scroll bars. If you do hide them, you can still scroll text using the keyboard.
4. Set the number of spaces between tab stops. The default setting is 8 spaces.
5. Choose OK.

You can also control the screen display by specifying one or more switches when you enter the **edit** command at the command prompt. For more information about these switches, see the *MS-DOS 5.0 User's Reference.*

Specifying the Help Path

If you started Edit from a directory other than the directory that contains the EDIT.HLP file, Edit will not be able to display help information unless you specify the path to this file using the Help Path command in the Options menu.

▶ **To specify the Help path:**

1. Choose Help Path from the Options menu.
2. Enter the path to the directory that contains EDIT.HLP.

If you use the MS-DOS **path** command to set a path to the directory that contains EDIT.HLP, you do not need to use the Help Path command.

Part 3
Customizing MS-DOS

Chapter 11

Working with Batch Files

11

As you work with MS-DOS, you'll find yourself entering identical sequences of commands repeatedly. For example, you might enter the same three commands to change the current drive, change the current directory, and start a program. With MS-DOS you can store the three commands in a special file called a batch file. Instead of entering the commands individually, you can run the whole sequence by entering the name of the batch file. MS-DOS performs these "batches" of commands as if you had entered them from the keyboard.

Understanding Batch Files

Batch files are unformatted text files that contain one or more MS-DOS commands. For example, a batch file might contain the commands you use to change directories and start a word processor. Or it might contain the following commands that back up your files to a floppy disk:

```
cd \word\docfiles
copy *.doc a:
cd \excel\xfiles
copy *.xls a:
```

To make these four commands into a batch file, you store them in an unformatted text file and assign the file a .BAT extension. Each time you want to back up your files, you enter the name of the batch file at the command prompt.

Any task you can perform with a batch file, you can also perform at the MS-DOS command prompt or in the Shell. However, there are several good reasons to use batch files:

- Batch files speed up your work. By using a batch file, you only have to remember one command instead of several. You do not have to take the time to retype multiple commands or to look up commands you can't remember.

- Batch files customize MS-DOS. Using batch files, you can create personalized commands that perform exactly the task you need. Batch files also let you design commands, prompts, and messages that add a personal touch to your system.

Each MS-DOS command has rules that you must follow whether you enter it at the MS-DOS command prompt or include it in a batch file. In addition, there are rules that govern how the batch file is created and used.

Batch File Commands

Any MS-DOS command that you can enter at the MS-DOS command prompt you can also put in a batch file. In addition, there are eight MS-DOS commands that are specially designed for batch files. These commands let you customize your batch files so that they act like programs, displaying messages on the screen and performing different tasks, depending on what the user of the batch file requests.

The following lists these special batch commands and describes what they do:

echo	Displays messages on the screen.
pause	Stops running the batch file until the user presses a key.
rem	Annotates your batch files so that you can read them more easily.
if	Runs different commands under different conditions.
goto	Jumps to the commands in different parts of your batch file.
shift	Increases the number of parameters you can use with a batch file.
call	Runs a second batch file and then returns to the first one.
for	Performs a command for a group of files.

The **echo, pause, rem, call, if,** and **goto** commands are explained later in this chapter. For information about the **shift** and **for** commands, see the *MS-DOS 5.0 User's Reference.*

Because batch commands are internal commands, you don't need to specify the directory that contains them.

Tools for Creating Batch Files

You can create a batch file using Edit, described in Chapter 10, "Working with the MS-DOS Editor," or the **copy** command. If you want to use a word processor other than Edit to create batch files, be sure it can save files as unformatted (ASCII) text. Most popular word processors have an option for saving files this way.

The **copy** command is a convenient way to create brief batch files. You can find more information about using the **copy** command to create batch files later in this chapter.

Naming Batch Files

You can give a batch file any valid name, but you must assign it a .BAT extension so that MS-DOS can identify it as a batch file.

It is generally not a good idea to give a batch file the same name as an existing MS-DOS command. Suppose, for example, that you create a batch file for a customized format command. If you name this batch file FORMAT.BAT, you have to be careful about how you start the batch file. If your FORMAT.BAT file is in the same directory as the MS-DOS FORMAT.EXE file, the batch file will never be used because MS-DOS gives precedence to files with the .EXE extension. In fact, if the MS-DOS FORMAT.EXE file is anywhere ahead of your batch file in the directory search path, your file will never be used. The only way you could be sure that FORMAT.BAT runs is to make its directory current or type its full pathname. You can avoid these problems by using names that are not already assigned to MS-DOS commands (for example, you could name the file MYFMT.BAT).

Running Batch Files

To run a batch file, you type its filename without the extension. For example, to run a batch file called FILES.BAT, enter the following command:

```
files
```

If the batch file has parameters, separate them from the name with a space. For example, if the FILES.BAT batch file expects the pathname of a directory as a parameter, you would enter a command like this:

```
files c:\excel\data
```

Unless you tell it otherwise, MS-DOS displays the commands of the batch file as they are executed. At the end of the batch file, MS-DOS might display two command prompts because it treats the end-of-file character in the batch file as an extra carriage return.

Stopping Batch Files

If you want to stop a batch file before all its commands have run, press CTRL+C or CTRL+BREAK. MS-DOS asks you to confirm that you want to terminate the batch file. Press Y to stop running the file or N to continue.

You can stop a batch file temporarily by pressing CTRL+S or PAUSE. This key combination freezes the display and pauses the batch file until you press any key.

If you remove the floppy disk that contains a batch file being run, MS-DOS prompts you to reinsert the disk so that it can continue running the file.

Debugging Batch Files

It is generally best to create large batch files in steps. This ensures that one part of the batch file works before you create the next part.

When you run a batch file, if one of the commands isn't correct, MS-DOS cancels it and proceeds to the next command. If the batch file is set up to display commands as MS-DOS runs them, you'll see an error message when a command is incorrect. If you do not see commands and error messages displayed, the batch file has an **echo off** command in it. To see commands and error messages displayed, make sure your batch file has no **echo off** commands.

If necessary, you can press CTRL+S to freeze the display while the batch file is running. When you freeze the display, the batch file also stops.

Undoing What You Have Done

The changes that occur because of the commands you execute in a batch file remain in effect until you specifically reverse them. Therefore, it is sometimes worthwhile to make your batch file undo the effects of its commands before the batch file ends. For example, if your batch file changes the current directory to perform a series of tasks, it is best to reset the current directory to what it was when the batch file began. If your batch file creates any temporary files, you may want the batch file to delete them before it ends so that they don't take up space on your disk.

The AUTOEXEC Batch File

There is a special batch file, called AUTOEXEC.BAT, that contains commands you want MS-DOS to run each time you start your system. For example, if you want MS-DOS to set the search path, configure your printer, and change the MS-DOS command prompt each time you start your system, you can put a **path** command, a **mode** command, and a **prompt** command in your AUTOEXEC.BAT file.

The AUTOEXEC.BAT file is described in Chapter 12, "Customizing Your System."

Creating Small Batch Files

Creating a small batch file can be as easy as typing an MS-DOS command. Using the **copy** command, you can create a batch file directly from the keyboard. Suppose, for example, that you want to create a batch file that formats a 360K disk in your high-density disk drive. To create the file and name it MYFMT.BAT use the following **copy** command:

```
copy con c:\myfmt.bat
```

MS-DOS moves the cursor to the next line and displays it without the command prompt. At this point the file is empty.

To add the **format** command to the file, type the following line of text:

```
format a: /f:360
```

With the command in the batch file, you are ready to close the file and return to the MS-DOS command prompt by pressing CTRL+Z followed by ENTER.

Once you have created the batch file, you need only enter the name of the batch file to format a 360K floppy disk in your high-density disk drive as in the following example:

```
myfmt
```

MS-DOS displays the format command in the batch file, and then runs it (assuming the directory that contains this batch file is either current or in the directory search path). The result is the same as if you had entered the **format** command yourself.

The next batch file is also short enough to create using the **copy** command. The file backs up .DOC and .XLS files from two directories onto a floppy disk. As when creating the MYFMT.BAT file, you begin by entering the copy command and naming the file, as follows:

```
copy con c:\bakit.bat
copy c:\word\may\*.doc a:
copy c:\excel\may\*.xls a:
dir a:
```

If there are more than a few lines to your batch file, you'll want to use an editor such as Edit to create the file.

Displaying Messages with the Echo Command

In Brief

To display a line of text on the screen, use the **echo** command in your batch file as in the following command:

```
echo Batchworks Backup Builder
```

To stop MS-DOS from displaying commands as it runs them, use the following command:

```
echo off
```

A useful tool for customizing MS-DOS is the ability to display messages. When you want to tell the user what the batch file will do, or ask for more information, you can display a message.

To display messages on the computer screen, use the **echo** command. For example, the following command tells MS-DOS to display the message "Put a disk in drive A":

```
echo Put a disk in drive A
```

MS-DOS displays the message at the beginning of the next line of the display. If you want the message shifted to the right, you must add the extra space as part of the message. For example, to center the message on the screen, add a number of spaces to the command:

```
echo          Put a disk in drive A
```

If **echo** is off, you can skip a line on the display by entering **echo** without a message. If **echo** is on, enter **echo** followed immediately by a period (.) as in this command:

```
echo.
```

Because MS-DOS displays commands in a batch file as it runs them, the "Put a disk in drive A" message of the preceding example is displayed twice: first as part of the batch file command and secondly as a result of the command itself. To display a message only once, use the following command before displaying messages:

```
echo off
```

For example, you can add an **echo off** command to the beginning of the BAKIT.BAT file described in the previous section:

```
echo off
copy c:\word\may\*.doc a:
copy c:\excel\may\*.xls a:
cls
echo Here are the files on the backup disk:
echo.
dir a:  /p
```

All commands following the first are not displayed, including the two **echo** commands that introduce the directory listing. To control the display of the message and directory listing, the file clears the screen and adds the **/p** switch to limit the amount of directory information displayed at a time.

TIP To keep a single command in your batch file from being displayed put an ampersand (@) in front of it. For example, to supress display of the **echo off** command, enter **@echo off**.

To begin displaying commands again use this command:

```
echo on
```

Pausing a Batch File

In Brief

To stop a batch file from running momentarily, use the **pause** command, as in the following example:

```
pause
```

When MS-DOS finds a **pause** command in a batch file, it displays the following message:

```
Press any key to continue . . .
```

MS-DOS stops executing the file until the user presses a key.

For example, adding a **pause** command to the BAKIT.BAT batch file will enable you to momentarily stop running the batch file while you put a disk in drive A:

```
echo off
echo Put a backup disk in drive A then
pause
copy c:\word\may\*.doc a:
copy c:\excel\may\*.xls a:
cls
echo Here are the files on the backup disk:
echo.
dir a:  /p
```

A new message reminds you to put a backup disk in drive A before the batch file pauses.

As a result, MS-DOS displays the following:

```
Put a backup disk in drive A then
Press any key to continue . . .
```

The **pause** command is particularly helpful if you want to give the user of the batch file a chance to consider the results of a potentially dangerous operation.

Including Remarks in a Batch File

In Brief

To put a remark in a batch file, use the **rem** command, as in the following example:

```
rem This part of the batch file copies files to the backup disk.
```

If your batch files are longer than a few lines, it is helpful to include remarks that explain what the batch file is doing. Remarks are generally useful for two reasons: to comment on the commands in a batch file, and to make the file easier to read by separating it into sections.

After you type REM and a space, MS-DOS ignores any other text on the line. Therefore, you can type any characters you want on a remark line, or you can type REM and then leave the rest of the line blank to add space to the file.

For example, the following remarks divide and explain sections of BAKIT.BAT:

```
rem *****Backup the MAY dirs*****
rem
echo off
echo Put a backup disk in drive A then
pause
copy c:\word\may\*.doc a:
copy c:\excel\may\*.xls a:
rem
rem ********************************************************
rem Clear the screen and display the files that were backed up:
rem
cls
echo Here are the files on the backup disk:
echo.
dir a:  /p
```

Remarks don't affect the way a batch file runs; they simply explain the commands to anyone who reads through the file.

Starting Batch Files Inside Batch Files

In Brief

To start a batch file from inside another batch file, either include the name the file you want to start or use the **call** command. The first method permanently exits the file that includes the name and starts the second batch file. The second method exits the file, runs the second batch file, and returns to the original file when the second ends. For example, the following command starts the SUBTASK.BAT batch file from inside another batch file and returns to the original file:

```
call subtask
```

If you don't need to return to the original batch file, you can start SUBTASK.BAT without the **call** command.

If you want to execute two batch files one after another, enter the name of the second batch file as the last command in the first batch file. For example, the following batch file runs four commands. Rather than returning you to the MS-DOS command prompt when it ends, the program starts the NEXTONE.BAT batch file:

```
a:
cd \tmp
copy c:\*.sys
cd \perm
nextone
```

After it runs the four commands, MS-DOS starts the NEXTONE.BAT batch file. When NEXTONE.BAT ends, MS-DOS displays the command prompt.

If you want to return to the original batch file after running a second batch file, invoke the second batch file with a **call** command. When the second batch file ends, rather than displaying a system prompt, MS-DOS returns to the original batch file and runs the next command. For example, the following batch file runs two commands, invokes the NEXTONE.BAT batch file, and then runs two more commands.

```
a:
cd \tmp
call nextone
copy c:\*.sys
cd \perm
```

After the first two commands, MS-DOS starts the NEXTONE.BAT file. When NEXTONE.BAT ends, MS-DOS returns to the batch file and runs the fourth and fifth commands.

Using Replaceable Parameters

In Brief

Replaceable parameters (%0 to %9) are placeholders for parameters entered at the command prompt. For e: .mple, the following command in the BAKIT.BAT file includes two replaceable parameters:

```
copy %1 %2
```

At the command prompt the user types the two correspondi.:g parameters, as in the following command:

```
bak c:\comm\*.* a:
```

MS-DOS replaces %1 with C:\COMM\*.* and %2 with A:.

Suppose you want to create a batch file that moves a file from one directory to another. In its simplest form, the batch file consists of a **copy** command and a **del** command. The **copy** command requires two parameters to specify the source and destination files; the **del** command requires one to specify the file to be deleted. For example, the following moves the CASTILE.EXE file from the root directory of drive A to the GAMES directory of drive C:

```
copy a:castile.exe c:\games
del a:castile.exe
```

As a batch file, the example is very limited because you can use it to move only a specific file, CASTILE.EXE. To create a batch file useable for any file, you need a way to replace MS-DOS parameters in the batch file with information the user enters when invoking the file. Replaceable parameters provide it.

MS-DOS includes 10 replaceable parameters numbered %0 through %9. The %0 replaceable parameter substitutes for the first piece of information the user types, %1 substitutes for the second, %2 for the third, and so on. If you want to specify more than 10, use the **shift** command described in the *MS-DOS 5.0 User's Reference.*

If you use the percent sign as part of a filename *within a batch file*, you must type it twice. The first % tells MS-DOS that the second % is part of a name, rather than a placeholder.

You don't have to start with **%0**. For instance, in the preceding example, you could use **%1** to substitute for A:CASTILE.EXE and **%2** to substitute for C:\GAMES.

MS-DOS uses the parameters in the order they are entered at the command prompt. In this case, if the user enters the command MOVE A:CASTILE.EXE C:\GAMES, MS-DOS substitutes A:CASTILE.EXE for parameter **%1** and C:\GAMES for parameter **%2**.

If the batch file requires a parameter that the user does not provide, MS-DOS displays an "Invalid parameter" message.

In addition to replaceable parameters, you can use environment variables in a batch file. For more information about environment variables, see the set command in the *MS-DOS 5.0 User's Reference.*

Controlling When Commands Run

To increase the flexibility of your batch files, you can use the **if** command to run different commands under different conditions, and the **goto** command to jump to different parts of the batch file. In conjunction with replaceable parameters, conditional and jump commands allow your batch files to perform complex tasks that exceed what you can do by entering commands at the command prompt.

Using Goto

In Brief

To jump to another part of the batch file, use the **goto** command and a label, as in the following command:

```
goto runword
```

The label *runword* (preceded by a colon) must appear on its own line somewhere else in the batch file as in the following example:

```
:runword
```

The **goto** command tells MS-DOS to jump to another line of the batch file. The line of the batch file that MS-DOS should jump to is marked with a label preceded by a colon (:). The same label appears in the **goto** command as is shown in the following illustration:

This artwork is not available for this release.

bat_1

Using If Commands

In Brief

To execute a batch file command only when one or more conditions are met, use the if command. For example, the following command will start Microsoft Word when the user types W as a parameter:

```
if "%1" == "W" c:\word\word
```

The parameter and the text it is compared to must be enclosed in quotation marks and they must match exactly. In this case the user must enter an uppercase W.

The **if** command lets you specify a condition that must be true in order for MS-DOS to run a command. For example, suppose that you want to create a batch file named RUNIT.BAT that starts your chess program, CMATE.EXE, when the user enters the following command:

```
runit A
```

To perform this task, include the following **if** command:

```
if "%1" == "A" cmate
```

When MS-DOS runs the preceding command, it checks to see if **%1** is an A (the double equal sign (==) means the parameter must equal the value). If **%1** is an A, MS-DOS runs the command that follows (start CMATE.EXE). When the user exits from CMATE.EXE, MS-DOS runs the command on the next line of the batch file.

If **%1** is not an A, MS-DOS skips the command that runs CMATE.EXE and moves to the next line of the batch file. The parameter and the letter that it is compared to must be enclosed in quotation marks to tell MS-DOS that they are text.

You can also use the **if** command with negative signs, filenames, and error values. For more information, see the **if** command in the the *MS-DOS 5.0 User's Reference.*

Using If and Goto Together

In Brief

To jump to a different line of a batch file when one or more conditions are met, use an **if** command with a **goto** command, as in the following example:

```
if "%1" == "W" goto runword
```

If you use the **goto** command with an **if** command, you can make MS-DOS run different sections of a batch file under different conditions.

For example, the following command tells MS-DOS to jump to the line labeled *chess* if the user enters an A:

```
if "%1" == "A" goto chess
```

If the user enters anything other than an A as a parameter, including no parameter, MS-DOS moves to the next line of the batch file. With a series of if commands you can create a batch file that will allow you to run any number of programs.

For example, suppose you want the batch file to change to the C:\GAMES\CHESS directory and run CMATE.EXE if the user enters A, or change to the C:\GAMES\CHECK directory and run CHECKERS.EXE if the user enters anything but an A.

You can use the preceding if command with the following other commands to run other games, depending on what the user enters:

```
if "%1" == "A" goto chess
rem
rem**************************************************
rem If the user doesn't type A, run checkers.
cd \games\check
checkers
rem skip over chess by going to the line labeled :end
goto end
rem
rem ******************************************
rem If MS-DOS goes to this label, the user wants chess.
:chess
cd \games\chess
cmate
rem The following line marks the end of the batch file.
:end
```

Creating a Batch File Menu System

One way to customize MS-DOS is to create a batch-file menu system that lets you type simple commands to start the programs you normally use. Menus are particularly helpful if novice users will be using your system.

For example, suppose that you use Microsoft Word and a number of computer games. In addition, you use a batch file to back up Word document files to a floppy disk. Using a series of batch files, you can create a menu system that lets you or anyone else use these programs without having to know where they are or how MS-DOS starts them.

The customized menu you create might look this:

```
        Start Menu


    Here's what you can do:

    1. Back up the Word document files

    2. Start Microsoft Word
```

```
3. Play a game

4. Use MS-DOS


Type the number you want and press ENTER:
```

You can create this menu and customized command prompt with very little effort. With the following **echo** and **prompt** commands you can tell MS-DOS to display the menu and prompt:

```
echo off
cls
echo      Start Menu
echo.
echo.
echo Here's what you can do:
echo.
echo 1. Back up the Word document files
echo.
echo 2. Start Microsoft Word
echo.
echo 3. Play a game
echo.
echo 4. Use MS-DOS
echo.
echo.
prompt Type the number you want and press ENTER:
```

The **cls** command clears the screen before MS-DOS displays the menu. The **prompt** command changes the command prompt so that it asks for a menu option. Notice, however, that the MENU.BAT file doesn't do anything but clear the screen and display a number of messages. The real work is done by other batch files that perform the tasks on the menu. The menu itself simply tells the user what to enter to start the batch file that performs a task.

The batch files that perform each menu option are named so that the user enters a single number to start them:

- The batch file that the backs up Word document files is called 1.BAT
- The batch file that starts Microsoft Word is called 2.BAT.
- The batch file that allows you to play a game is called 3.BAT.
- The batch file that returns you to MS-DOS is called 4.BAT.

Because the name of the batch file that performs an option is identical to the number of the option on the list, when users enter a number at the menu prompt they are actually starting a new batch file.

Making a Menu Item with Pause

Menu option 1 on the Start Menu is "Back up the Word document files." The batch file that performs this option must be called 1.BAT so that the user can start it by entering 1 at the menu prompt.

This batch file copies .DOC files from their usual directory to a floppy disk. The batch file includes a **pause** command that waits for the user to put a backup disk in the disk drive. Because the user might want to return to the menu to perform some other function, the 1.BAT batch file starts the MENU batch file when it ends.

If the Word document files are in the C:\WORD\DOCFILES directory, and the backup drive is A, 1.BAT would contain the following commands:

```
echo off
cls
echo To back up the word files, put the backup disk in drive A
echo and press ENTER
pause
echo These files are being backed up:
copy c:\word\docfiles\*.doc a:
echo.
echo When you are ready to return to the Start Menu, press ENTER
pause
menu
```

By choosing 2 from the Start Menu, the user tells MS-DOS to run this batch file. The batch file clears the screen and prompts the user to insert the backup disk. After the files are copied, the batch file pauses to give the user a chance to read the display. As its last command, the batch file runs the MENU.BAT batch file to return the user to the Start Menu.

Making a Menu Item with Replaceable Parameters

Menu option 2 on the Start Menu is "Start Microsoft Word." The easiest way to perform this option is to have a batch file called 2.BAT that starts Microsoft Word. If the Word program file is C:\WORD\WORD.EXE, the following command would start the program:

```
c:\word\word
```

One way to make the option more useful is to have it display available Word document files and then give the user the choice of editing an existing file or opening a new file. The two tasks require two batch files. The first displays the existing files and prompts the user for a file to edit; the second starts Word with an existing file or a new file.

The first batch file is like a menu. It displays a list of files to choose from and then asks the user to choose one. This batch file must be called 2.BAT so the user can start it by entering a 2 at the menu prompt.

The 2.BAT batch file would contain the following commands assuming that the Word files have a .DOC extension and are in the C:\WORD\DOCFILES directory, and that the Word program file is C:\WORD\WORD.EXE:

```
echo off
cls
cd c:\word\docfiles
echo          Start Microsoft Word
echo.
echo.
echo Here are the Word files that already exist:
echo.
dir *.doc /p
echo.
echo To edit a new file, type W and press ENTER.
echo.
echo To edit an existing file, type W and the filename
echo (separated by a space) then press ENTER.
echo.
prompt What'll it be?:
```

The batch file makes the Word document directory current and then shows the user which files are available. It sets the current directory to C:\WORD\DOCFILES so the user doesn't have to enter any directory pathnames. The **dir** command uses the **/p** switch so that MS-DOS will pause if there is more than one screenful of files. The batch file displays messages that explain how to start Word with a new or existing file. The final command in the batch file changes the prompt.

The user runs this batch file by entering 2 at the prompt that the Start Menu displays. The batch file produces a screen that looks like this:

```
Start Microsoft Word


Here are the Word files that already exist:

CH1 DOC
CH2 DOC
CH3 DOC
CH4 DOC

To edit a new file, type W and press ENTER.

To edit an existing file, type W and the filename
(separated by a space) and then press ENTER.
```

```
What'll it be?:
```

To start Word, you need a second batch file. To create such a batch file, you can take advantage of the fact that Word, like most word processing programs, allows you to specify a file to edit when you start it. For example, to start Word and edit C:\WORD\DOCFILES\HONCH.DOC you would enter this command:

```
c:\word\word c:\word\docfiles\acct.doc
```

C:\WORD\WORD starts the program; C:\WORD\DOCFILES\ACCT.DOC is a parameter that tells Word which file you want to edit. If you don't enter a filename, Word assumes that you want to edit a new file.

You can use a replaceable parameter to start Word and tell it which file to edit (if any). According to the instructions that 2.BAT displays, the user enters W to edit a new file or a W plus a filename to edit an existing file. You can execute these instructions by creating a batch file called W.BAT that includes the following command:

```
c:\word\word %1
```

When the user enters W, MS-DOS starts the W.BAT batch file. To ensure that the user doesn't have to type a pathname, W.BAT must be in the current directory (C:\WORD\DOCFILES) or in the directory search path.

If the user enters a filename after the W, MS-DOS assigns it to the **%1** parameter, and Word starts with that file. If the user does not enter a filename, MS-DOS ignores **%1**, and Word starts with a new file.

W.BAT needs two more commands to get the user back to the Start Menu. When the user exits from Microsoft Word, MS-DOS returns to the batch file and runs the next command. To return to the menu, the batch file should change the current directory back to the root directory of drive C and display the menu. The following shows the command that starts Word, and the two commands that return the user to the Start Menu. Together, these commands constitute W.BAT:

```
c:\word\word %1
cd \
menu
```

Making a Menu Option with If and Goto

Option 3 of the Start Menu is called "Play a game." To create its batch file, use the same sort of **if** and **goto** commands that were used in the examples in "xx" earlier in this chapter.

Like the batch files that start Microsoft Word, there are two batch files for this option: one that displays a games menu and one that starts the appropriate game. The first must be named 3.BAT so the user can start it from the Start Menu by entering 3. If there are three games you want to list (for example, chess, checkers, and backgammon), 3.BAT would contain the following:

```
echo off
cls
echo        Let's Play a Game!
echo.
echo.
echo Here are your choices:
echo.
echo A. Play chess.
echo B. Play checkers.
echo C. Play backgammon.
echo.
echo To start a game, type "game" and the letter of the
echo game. Then press ENTER (for example, game A).
echo.
prompt What'll it be?:
```

Like the batch file that displays the Start Menu, this batch file simply presents the alternatives. You could create a separate batch file for each option. However, by using **if** and **goto** commands, you can cover all the options in a single batch file.

The instructions in the menu tell users to enter *game* and the letter of the game they want to play. If the chess game is \GAMES\CHESS\CMATE.EXE, the checkers game is \GAMES\CHECK\CHECKERS.EXE, and the backgammon game is \GAMES\BACK\BACKGAMM.EXE, then a batch file called GAME.BAT containing the following commands will perform all options:

```
rem ***** This batch file runs a game *****
echo off
cls
rem See which game the user chose
if "%1" == "A" goto chess
if "%1" == "B" goto check
if "%1" == "C" goto back
rem
rem **********************************************
rem If MS-DOS comes here, the user didn't choose a game.
echo.
echo Enter game A, game B, or game C.
pause
rem If no game, skip over all of the game commands.
goto end
:chess
rem
rem **********************************************
rem if MS-DOS comes here, the user chose chess
```

```
cd \games\chess
cmate
rem If the user chooses chess, skip checkers and backgammon
goto end
:check
rem
rem ***********************************************
rem If the user chooses checkers, go here
cd \games\check
checkers
rem If the user chooses checkers skip backgammon
goto end
:back

rem
rem ***********************************************
rem If the user choose backgammon, go here
cd \games\back
backgamm
rem
rem ***********************************************
rem When the games are finished change the
rem directory back and display the Start Menu.
:end
cd \
menu
```

MS-DOS jumps to one of three locations in the batch file, depending on the parameter the user types. If the user omits a parameter or types an incorrect parameter, the batch file displays an error message and returns the user to the Start Menu. The batch file GAME.BAT must be in the root directory or in the directory search path so that the user need only enter **game** to start it.

The batch file changes to the directory that contains a game in case the game program and its data files must be in the current directory. However, before the batch file finishes, it resets the directory to the root. The last statement of the batch file starts the batch file that displays the Start Menu.

Exiting from the Menu System

To provide a way for users to get back to the familiar command prompt, the 4.BAT batch file contains the following commands:

```
prompt $p$g
cls
```

The Start Menu prompts the user to enter a number. If the user enters 4, MS-DOS runs the 4.BAT batch file. The batch file clears the screen and sets the command prompt to show the current drive and directory.

Chapter 12

Customizing Your System

12

You can customize how MS-DOS uses hardware, memory, and files to get different results or to meet special conditions. For instance, you can customize the way MS-DOS interacts with your keyboard and monitor and change the amount of memory MS-DOS reserves for itself and for storing files.

A piece of hardware that enables you to communicate with the computer is called a *device*. You use devices such as a keyboard or mouse to give the computer information (*input*), and devices such as a monitor or printer to receive information from the computer (*output*). Each device has its own characteristics that can be customized.

When MS-DOS starts it looks for a special batch file that defines the characteristics of devices on your system. The file, named AUTOEXEC.BAT, can also include any MS-DOS commands you want to run each time you switch on your system. For example, you can define the port to which your printer is connected, control the rate that MS-DOS repeats a keystroke when you hold a key down, define the path that MS-DOS uses to find files, and clear your screen of startup messages.

For every device, there is a program that MS-DOS uses to control it. Such a program is called a *device driver*. For instance, MS-DOS uses a device driver to control how information is read to and from your floppy disk drive. MS-DOS has built-in device drivers for your keyboard, monitor, hard and floppy disk drives, and ports. MS-DOS includes other device drivers that are installable. Unlike built-in drivers, which are part of MS-DOS, installable device drivers are stored on disk. When you want to use them, MS-DOS must install them by transferring the drivers from disk to memory.

You tell MS-DOS what devices to install, and which built-in and installable device drivers to use, with a file named CONFIG.SYS. This file contains special CONFIG.SYS commands that also determine how MS-DOS uses memory and controls files. (For more information on MS-DOS and memory, see Chapter 13, "Optimizing Your System.") When you start your system, MS-DOS runs commands in the

CONFIG.SYS file before running AUTOEXEC.BAT. First CONFIG.SYS determines the devices you want on your system; then AUTOEXEC.BAT defines their characteristics.

If you have more th[illegible]n two floppy disk drives on your system, use CONFIG.SYS to install the DRIV[illegible]R.SYS driver. DRIVER.SYS controls how MS-DOS uses additional floppy disk [illegible]rives. If you want to change characters that a key displays or the way your monitor displays characters, you'll need the installable driver named ANSI.SYS. This chapter concludes with a description of the ANSI.SYS commands.

Creating a Startup Procedure

A startup procedure is one or more commands that MS-DOS runs automatically each time you start your system. The commands can set the characteristics of your devices, customize information that MS-DOS displays, and start batch files and programs.

MS-DOS comes with a startup procedure that is defined in a batch file named AUTOEXEC.BAT. The file is located in the root directory of your system disk. Each time you switch on your system, MS-DOS looks for this AUTOEXEC.BAT file and automatically runs the commands stored in it. You can also run AUTOEXEC.BAT anytime without restarting your system by entering **autoexec** at the command prompt.

You can make your own startup procedure in the AUTOEXEC.BAT batch file using the techniques described in Chapter 11, "Working With Batch Files." Be sure to store your startup procedure in a file named AUTOEXEC.BAT so that MS-DOS runs your procedure automatically. To avoid loosing the original MS-DOS AUTOEXEC.BAT startup file, change its name before you create your own.

Startup Commands

Every command in an AUTOEXEC.BAT file can also appear in any other batch file. Each is fully described elsewhere in this *User's Guide* or in the *MS-DOS 5.0 User's Reference*. The following list briefly describes the most common AUTOEXEC.BAT commands and indicates where to find more information about them:

- The **mode** command sets the characteristics of your keyboard, display, parallel ports, and serial ports. For more information about the **mode** command, see xx later in this chapter.

- The **date** and **time** commands prompt you to enter the correct date and time. These commands are important to include in your AUTOEXEC.BAT file only if your system does not have a clock that stays current when the power is off. For more information about the **date** and **time** command, see "xx".
- The **path** command indicates the directories where MS-DOS will search for a program file that you want to run. For more information about the **path** command, see xx
- The **echo off** command tells MS-DOS not to display the commands of the AUTOEXEC.BAT file as they are executed. For more information about the **echo** command, see xx.
- The **set** command creates an environment variable that can be used by programs and batch files. For more information about the **set** command, see the *MS-DOS 5.0 User's Reference.*

AUTOEXEC.BAT files often contain commands that run other batch files or programs. These commands allow you to start a program or batch file automatically without having to enter its name. For example, to start the Shell each time you switch on your system, you would include the following command at the end of your AUTOEXEC.BAT file:

```
dosshell
```

After MS-DOS finishes running all the batch files and programs in the AUTOEXEC.BAT file, it displays the command prompt (or the Shell, if the AUTOEXEC.BAT file is set up to start the Shell).

Instead of the command that runs the Shell, you might want to include the command that installs Doskey and any Doskey macros you commonly use. For more information, see xx.

Sample Startup Procedures

If you are satisfied with the way MS-DOS starts and sets up your devices, you need not create your own startup procedure in an AUTOEXEC.BAT file. On the other hand, some commands are best suited for an AUTOEXEC.BAT file.

For example, a system with two floppy-disk drives and a clock that does not run when the computer's power is off might use an AUTOEXEC.BAT file with the following commands:

```
date
time
path a:
```

The **date** and **time** commands prompt you to set the date and time when you start MS-DOS. Comm::nds such as **xcopy**, **backup**, and **restore** may not work correctly if your cl: is not accurate. The command **path a:** tells MS-DOS to look for commands or programs on drive A in addition to the current directory.

For a system with one floppy-disk drive, one hard drive, a clock that does not need to be set, and the MS-DOS Shell installed, an AUTOEXEC.BAT file might contain the following commands:

```
path=c:\;c:\utility;c:\batch;c:\word;c:\excel
prompt=$p$g
doskey shell
```

The **path** command tells MS-DOS to look for program files in the current directory, and then in the following directories: the root directory of drive C, C:\UTILITY, C:\BATCH, C:\WORD, and C:\EXCEL. Semicolons separate the directories.

The **prompt pg** command displays the current drive and directory, followed by a greater-than sign (>), as the MS-DOS prompt. This is one of the more common forms of the MS-DOS command prompt.

The **doskey** command loads DOSKEY.COM into memory. Because the **doskey** command comes after the **path** command, DOSKEY.COM can be located in any of the directories listed in the **path** command. Likewise, the **shell** command tells MS-DOS to start the program called DOSSHELL.EXE.

For a system with one floppy disk drive, one hard drive, a clock that does not need to be set, a laser printer attached to port COM1, and a menu batch file that runs automatically, an AUTOEXEC.BAT file might contain these commands:

```
echo off
path=c:\;c:\utility;c:\batch;c:\word;c:\excel
prompt=$p$g
mode lpt1=com1
mode com1:9600,n,8,1,p
set tmp=c:\temp
menu
```

The **echo off** command tells MS-DOS not to display the AUTOEXEC.BAT commands as it runs them. The first **mode** command redirects printer output from LPT1, its default port, to the serial port COM1. The second **mode** command sets up the COM1 port to use with a printer such as the Apple ™ LaserWriter ™ printer.

The **set** command creates an environment variable called TMP. Many programs use this particular variable to find out where they should store their temporary files.

The last command in the AUTOEXEC.BAT file starts another batch file called MENU.BAT. When MENU.BAT and any other programs or batch files that it starts end, MS-DOS displays a command prompt.

Configuring MS-DOS for Your System

Before looking for the AUTOEXEC.BAT file, MS-DOS runs a group of com-mands that load installable device drivers and reserve space in system memory for information processing. The file that contains these commands is named CON-FIG.SYS. Like the AUTOEXEC.BAT file, a version of CONFIG.SYS comes with MS-DOS and is located in the root directory of your system disk.

You can add and change CONFIG.SYS commands in the file to configure your system as needed. To edit the CONFIG.SYS file, use Edit or a word processor that saves files as unformatted (ASCII) text. For more information on using Edit, see Chapter 10, "Working with the MS-DOS Editor".

Because the CONFIG.SYS file controls how MS-DOS starts, MS-DOS runs it only when you start your system. If you change the file, you must start your sys-tem again to have the changes take effect.

An Overview of the Configuration Commands

There are 13 commands you can use in a CONFIG.SYS file. Except for the **break** command, you can't enter these commands at the MS-DOS command prompt; they must be included in the CONFIG.SYS file. The following table briefly de-scribes the purpose of each CONFIG.SYS command:

Command	Purpose
buffers	Sets the amount of RAM that MS-DOS reserves for informa-tion transfer to and from the disk.
files	Sets the number of files that MS-DOS will allow to be open at any one time.
device	Tells MS-DOS to include an installable device driver in the operating system.
break	Controls how often MS-DOS checks to see if you pressed CTRL+C or CTRL+BREAK.
rem	Allows you to include descriptive comments in your CON-FIG.SYS file.

Command	Purpose
lastdrive	Sets the number of disk letters MS-DOS recognizes as valid.
install	Tells MS-DOS to run a terminate-and-stay-resident (TSR) command while reading CONFIG.SYS.
country	Tells MS-DOS which country's language conventions to use.
shell	Tells MS-DOS to work with a command processor other than COMMAND.COM or to change the way COMMAND.COM is set up.
drivparm	Sets the characteristics of a disk drive.
fcbs	Sets the number of file control blocks (FCBs) MS-DOS can open concurrently over a network.
stacks	Sets the amount of RAM that MS-DOS reserves for processing hardware interrupts.
dos	Sets the area of RAM where MS-DOS will be located.

The **install, drivparm, fcbs,** and **stacks** commands are described only in the *MS-DOS 5.0 User's Reference*. The **country** command is described in Chapter 14, "Customizing for International Use". For information on using the **dos** command, see Chapter 13, "Optimizing Your System".

Installing Device Drivers

In Brief

To install an installable device driver, include a **device** command in your CONFIG.SYS file as in the following example:

```
device=c:\keyboard.sys
```

To use an installable device driver, you include a **device** command in your CONFIG.SYS file. This loads, or *installs*, the driver in memory. For example, to tell MS-DOS to use the device driver MOUSE.SYS in the C:\MOUSE directory, you would include the following command in your CONFIG.SYS file:

```
device=c:\mouse\mouse.sys
```

When MS-DOS reads this command, it loads MOUSE.SYS into memory. The driver becomes part of the MS-DOS system software and remains in memory until you switch on your system without the **device** command in your CONFIG.SYS file. In many cases, when you install a program that works with an installable device driver, it automatically adds the proper command to your CONFIG.SYS file. The **device** command is described fully in the *MS-DOS 5.0 User's Reference*.

Increasing the File Transfer Area

In Brief

To increase the memory that MS-DOS reserves for file transfer, use the **buffers** commands as in the following example:

```
buffers=30
```

When MS-DOS starts, it reserves an area in your main memory for temporarily holding information from disks. It divides the memory into units called *buffers* that are the same size as the sectors on the disk (usually .5K long). Each buffer can hold a sector of information from a disk. The buffers hold parts of files that are waiting to be stored or used by a program, as well as information from the disk's directories and file table.

As your directory structure becomes more complex, MS-DOS works more efficiently with additional buffers. The performance of MS-DOS increases up to about 50 buffers. However, the more buffers there are, the less space there is for other programs and data in memory.

To set the number of buffers that MS-DOS reserves for file transfers, use a **buffers** command in your CONFIG.SYS file. The amount of space actually reserved by the **buffers** command depends on the size of the disk sectors. For example, the following command tells MS-DOS to reserve 20 buffers for file transfer operations (10K of space for .5K sectors):

```
buffers=20
```

NOTE Disk caching programs such as SMARTDRIVE.SYS perform much of the work of buffers. If you use a disk caching program, you need fewer buffers. For more information about SMARTDRIVE.SYS, see Chapter 13, "Optimizing Your System".

Increasing the Number of Files You Can Have Open

In Brief

To increase the number of files that MS-DOS will allow you to have open simultaneously, use the files command as in the following example:

```
files=30
```

When MS-DOS starts, it reserves space in memory for a table that contains information about files that are currently open. The more files you expect to have open at a time, the more space MS-DOS needs to reserve for its table. You can have up to 255 files open at a time. To specify the maximum number of files you expect to have open, use the **files** command.

For example, to tell MS-DOS to reserve enough space for 30 files, enter the following command:

```
files=30
```

Choose a number based on the number of files you expect to have. If you run database or spreadsheet programs, or if you run MS-DOS with Microsoft Windows® or with network software, you are likely to need a value of 30. On the other hand, the larger the number you specify, the more space MS-DOS will take in memory and the less space you will have for programs and data.

If you do not include a **files** command in your CONFIG.SYS file, MS-DOS assumes a value of 8. If you specify more than the limit of 255 files, you'll receive an error message when MS-DOS runs the CONFIG.SYS file.

Increasing CTRL+C Checking

In Brief

To increase the number of times MS-DOS checks for CTRL+C, use the **break** command as in the following example:

```
break=on
```

Unless you tell it otherwise, MS-DOS checks to see if you pressed CTRL+C or CTRL+BREAK only while it is reading from the keyboard or writing to the display. For example, if you press CTRL+C while MS-DOS is saving a file to disk, MS-DOS will not respond until the next time it displays something on the screen.

To tell MS-DOS to check more frequently for these key combinations, include the following **break** command in your CONFIG.SYS file:

```
break=on
```

After MS-DOS runs this command it will check for a CTRL+C or CTRL+BREAK almost anytime the program or command you are using calls on MS-DOS to perform a task.

Some programs ignore or redefine CTRL+C and CTRL+BREAK. Pressing these keys while you are using such a program may have another effect or no effect at all.

The **break** command can also be used in a batch file or at the command prompt.

Increasing the Number of Logical Drives

In Brief

To increase the number of logical drives that your system recognizes, use the **last-drive** command as in the following example:

```
lastdrive=z
```

The default value is one more than the number of logical drives currently on your system.

When MS-DOS starts, it reserves space in memory for a table that contains information about each logical drive that you expect your system to use. Regardless of how many physical disk drives your system has, you can tell MS-DOS to leave space for up to 26 logical drives. You might need to increase the number of logical drives if your system is part of a network, or if you use the **subst** command to equate a directory with a disk drive. For more information about physical and logical drives, see xx, later in this chapter.

To change the number of logical drives, include a **lastdrive** command in your CONFIG.SYS file. For example, the following command reserves space for 10 logical drives (A through J):

```
lastdrive=j
```

The **lastdrive** command prepares MS-DOS to recognize extra logical drives. However, you must still assign these drives (to physical devices, network shares, existing directories and so on) before you can use them. You cannot reserve space for fewer drives than you actually have. Unless you tell it otherwise, MS-DOS assumes that you have one drive more than the logical drives currently on your system.

Sample Configuration Files

If you have a mouse and you use spreadsheet or database programs, you might have the following commands in your CONFIG.SYS file:

```
buffers=20
files=30
device=c:\dos\mouse.sys
break=on
```

The **buffers** command reserves 30 buffers (15K for .5K sectors) for transferring information to and from disks. The **files** command reserves enough room to have 20 files open at the same time.

The **device** command tells MS-DOS to load the device driver MOUSE.SYS from the C:\DOS directory. The **break** command tells MS-DOS to check frequently for CTRL+C or CTRL+BREAK.

If you use a network, and your system is a 386<F2P8MJ251™ machine with expanded memory and 1K of extended memory, your CONFIG.SYS file might look like this:

```
buffers=20
files=30
rem   The following commands install the mouse, network ram cache,
and
rem   expanded memory drivers. The /a switch tells smartdrive to use
rem   expanded memory.
device=c:\mouse.sys
device=c:\net\network.sys
device=c:\bin\himem.sys
device=c:\bin\emm386.exe 1024
device=c:\bin\smartdrive.sys /a
break=on
rem The following command reserves space for 26 drives.
lastdrive=z
```

In addition to the commands contained in the previous CONFIG.SYS file, this file has commands to load additional device drivers and reserve space for additional drives.

A **rem** command indicates a comment in the CONFIG.SYS file. MS-DOS does not run **rem** commands.

The **device** commands tell MS-DOS to include the following four installable device drivers in the operating system: the program that lets MS-DOS communicate with the mouse, a program for managing the computer's network link, the program for creating memory caches, the program for managing extended memory, and the program for managing disk caches. For more information about the last two installable drivers, see xx.

The **lastdrive** command reserves space for 26 logical drives. In other words, on this system, letters from a to z are available as labels for drives.

Configuring Your Ports

To configure your parallel and serial ports, you use the **mode** command. Using the **mode** command, you can define how MS-DOS uses the printer ports when you execute commands such as **print**. You can also change settings on your serial ports. You can enter the **mode** command alone or with a device name and one or more options.

Other programs that you run may change the configuration of certain devices. For example, most word processing programs have their own printer drivers that may supersede the one you configure with the **mode** command.

Configuring Your Printer

In Brief

To connect your printer to a serial port, use the **mode** command as in the following example:

```
mode lptl = coml
```

MS-DOS redirects the output it would normally send to LPT1 to COM1.

To change the characters per line and lines per inch that MS-DOS assumes for your printer, use the **mode** command with the name of your printer's port. For example, the following command tells MS-DOS that a printer supporting Epson® compatible escape sequences has 132 characters per line and 8 lines per inch (the defaults are 80 and 6 respectively):

```
mode lptl:132,8
```

MS-DOS assumes your printer is attached to the LPT1 port unless you tell it otherwise. To tell MS-DOS that the printer is connected to a different port, use the **mode** command with the names of the two communication ports. For example, the following command tells MS-DOS to redirect its printer output from LPT1 to COM1:

```
mode lptl = coml
```

You can only redirect a parallel port to a serial port. You cannot redirect a parallel port to another parallel port and you cannot redirect a serial port to any other port.

For printers that support Epson compatible escape sequences, MS-DOS assume 80 characters across a line and 6 lines per inch of paper. If your printer prints 132 characters per line or 8 lines per inch, use the **mode** command with an LPT parameter to set MS-DOS to the new values.

For example, the following command configures MS-DOS for a printer attached to port LPT1 with 132 characters per line and 8 lines per inch:

```
mode lptl:132,8
```

You can add a retry option to tell MS-DOS how to behave when the printer does not accept information. For example, the following command tells MS-DOS to keep trying if the printer won't accept information:

```
mode lpt1:,,b
```

The two commas are placeholders for the first two options (characters per line and lines per inch), which will not change. MS-DOS will keep trying to send information to the printer until the printer responds or until you press CTRL+BREAK to stop printing.

If you use the **mode** command with no parameters, MS-DOS gives you information about your LPT, COM and CON ports. It tells you whether any of your LPT ports are re-routed. For LPT1, it also lists the retry option.

You can ask for information about a specific port by typing its name with the **mode** command. For example, the following command displays information about LPT1:

```
mode lpt1
```

Configuring a Serial Port

In Brief

To change the way MS-DOS communicates with a serial port, use the **mode** command with the name of the serial port as in the following example:

```
mode com1:9600,n,8,1,b
```

This command sets COM1 to communicate with printers such as an Apple LaserWriter.

To set up a serial port, you enter a **mode** with the name of the port. To specify exactly how MS-DOS should set up the port, you use one or more of the following switches:

- The **baud=** switch sets the rate at which MS-DOS communicates with the port. The default is 1200 baud.
- The **parity=** switch sets the error checking mode of the port. The default is even parity.
- The **data=** switch tells MS-DOS how many data bits the port expects. The default is 7.
- The **stop=** switch tells MS-DOS how many stop bits the port expects. The default is 1.

- The **retry=** switch tells MS-DOS how to behave if the device attached to the port is not ready. The default is to stop sending data if the device is not ready.

To change these settings for any of your serial communications ports, you can use the **mode** command with the name of the port.

For example, to set your COM2 port to work with a 2400 baud modem with even parity, seven data bits and one stop bit, you would use the following **mode** command:

```
mode com2:  baud=2400
```

You do not have to specify the parity, data and stop bits because they are the same as the default values.

You can type the name of the switch you want to change, or list the values you want with commas between each value. For example, the following two commands are equivalent:

```
mode com1:  baud=9600  parity=n  data=8  stop=1  retry=b
mode com1:9600,n,8,1,p
```

For the complete list of the values you can use for each switch, see the *MS-DOS 5.0 User's Reference.*

To display information about a COM port, enter a **mode** command with the name of the port. For example, the following command displays information about COM1:

```
mode com1
```

Adding Disk Drives

MS-DOS has a built-in device driver that controls two floppy disk drives. To add a third external floppy disk drive, you must install a separate device driver named DRIVER.SYS. You can use this device driver to control up to four external floppy disk drives. DRIVER.SYS cannot be used to control hard disk drives. For more information about built-in device drivers, see xx.

To use DRIVER.SYS, it's important to understand the difference between physical drives and logical drives. Physical drives are actual hardware components. The physical drives are numbered beginning at 0. Your first floppy disk drive is always physical drive 0, the second is always physical drive 1, and so forth. If you have a hard disk drive, it is physical drive 2 whether or not you have a second floppy disk drive. MS-DOS can support 127 different physical drives, but only 26 drive letters at a time.

Logical drives are labels that MS-DOS uses to keep track of where it sends data. Logical drive labels are letters a to z. Every physical drive has a corresponding logical drive. However, every logical drive does not have a corresponding physical drive. Your first floppy disk drive (physical drive 0) is always represented by MS-DOS as drive A. The second (physical drive 1) is always represented as B. You will have logical drives that do not correspond to physical drives if you use network drives, extended MS-DOS partitions, or RAM drives, or if you assign a drive letter to a directory.

In general, you need only logical drive letters. The only time you need to know the physical drive number of a device is when you define the drive with DRIVER.SYS or when you redefine the drive with a **drivparm** command in your CONFIG.SYS file. For information about the **drivparm** command, see the *MS-DOS 5.0 User's Reference.*

Installing the Driver

In Brief

To install DRIVER.SYS, include a **device** command in your CONFIG.SYS file. For example, the following command installs DRIVER.SYS for a new 5.25-inch 1.2M floppy disk drive:

```
device=driver.sys /d:2 /c /f:1
```

The switches are defined in the following paragraphs.

When you define a new disk drive, you tell MS-DOS which physical drive it is. MS-DOS gives it the next available logical drive letter. You cannot use DRIVER.SYS to change the setup for a previously defined logical drive. However, you can assign a second logical drive letter to a drive to change its characteristics.

To add a new floppy disk drive to your system, include a **device=driver.sys** command in your CONFIG.SYS file. To give the drive a physical drive number and specify what kind of drive it is, include one or more of the following switches:

- The **/d:** switch assigns a physical drive number to the new drive. The numbers range from 0 through 127. Drive numbers 0 and 1 are reserved for the two floppy disk drives that the built in device driver controls. Each DEVICE=DRIVER.SYS command must have a **/d:** switch.

  For example, **/d:3** switch sets the physical drive number to 3 (drive D).

- The /c switch tells DRIVER.SYS that the drive can sense when the drive door is open. If you specify this switch, MS-DOS assumes that the disk drive supports change-line error detection. This improves MS-DOS performance. See the documentation for your disk drive to see if it supports change-line.
- The **/f:** switch specifies the storage capacity of the disk drive you want to add. You can use any of the following values:

  0 = 160K/180K/320K/360K
  1 = 1.2 MB
  2 = 720K bytes
  7 = 1.44 MB or 2.88 MB
- The **/h:** switch specifies how many heads the drive has. You can specify a number from 1 through 99. The default value is 2.
- The **/s:** switch specifies the number of sectors per track on the disk drive. You can specify values from 1 through 99. The default value is 9.
- The **/t:** switch specifies the number of tracks per side on the disk drive. You can specify values from 1 through 999. The default is 80.

The following table shows the switch values that go together:

Drive type	/f	/h	/s	/t
360K or less	0	1 or 2	8 or 9	40
1.2 MB 5.25-inch	1	2	15	80
720K 3.25-inch	2	2	9	80
1.44 MB 3.25-inch	7	2	18	80
2.88 MB 3.25-inch	7	2	36	80

Suppose you want to add a 720K drive to your computer. The drive has two sides (that is, two heads), nine sectors per track, and 80 tracks per side. If DRIVER.SYS in the C:\DOS directory, you can make the new disk drive physical drive 2 with the following CONFIG.SYS command:

```
device=c:\dos\driver.sys /d:2/f:2
```

Because the disk drive has the default values for heads, sectors, and tracks, you don't have to include the switches that specify these values.

Assigning Two Drive Letters to a Single Drive

In Brief

To give two logical drive letters to the same physical device, include a **device** command in your CONFIG.SYS file. For example, the following command assigns the next available drive letter to the 360K disk drive that has drive letter A:

```
device=c:\dos\driver.sys /d:0 /f:1
```

If you include in your CONFIG.SYS file a **device=driver.sys** command that uses the number of an existing drive, MS-DOS assigns an additional drive letter to the disk drive. If you assign two drive letters to a drive, MS-DOS will prompt you to switch disks when you copy from one of the drive's logical device letters to the other.

For example, the following command tells MS-DOS to assign the next available drive letter to drive A (physical device 0):

```
device=c:\dos\driver.sys /d:0 /f:2
```

After this command, drive A has two different drive letters associated with it: A and the next available letter (D, for example). If a disk drive has two letters associated with it, MS-DOS allows you to copy files to and from that disk drive. For example, if drive A is also called drive D, you can use the following command to copy files to a different disk using only drive A:

```
copy a:*.* d:
```

MS-DOS prompts you to switch between the source disk and the destination disk.

Redefining Drive A or B

In Brief

To redefine a drive that your computer doesn't recognize, include a **device** command in your CONFIG.SYS file. For example, the following command redefines drive A as a 720K 3.25-inch floppy disk drive:

```
device=driver.sys /d:0 /f:2
```

MS-DOS gives the drive the next available drive letter.

Some computers are designed to function only with a certain kind of disk drive. If you try to use a different kind of drive, the computer does not recognize it. If your A or B floppy disk drive is incompatible with your system's settings, you can use DRIVER.SYS rather than the built-in device driver to control the drive.

For example, suppose your system came with a 360K 5.25-inch disk drive as drive B. When you try to replace the drive with a 720K 3.5-inch drive, your computer still assumes that it is a 360K drive and experiences disk errors when it accesses the drive.

To remedy the problem, you can use the following command to install DRIVER.SYS to control the drive:

```
device=driver.sys /d:1/f:2/h:2/s:9/t:80
```

The **/d:1** switch tells DRIVER.SYS you want to assign a new logical drive letter to drive B. The other switches tell DRIVER.SYS what kind of disk drive it is.

As a result of this command, the 3.5-inch drive will have two letters: B and the next available letter (say, E). When you use drive B, the computer will still experience disk errors. However when you use drive E, the computer will recognize the drive and allow you to use it. By using an installable device driver to control the drive rather than the built-in device driver, you are able to change the characteristics that MS-DOS assumes for the drive.

Controlling Your Monitor and Keyboard

In addition to the device driver that normally controls your monitor and keyboard, MS-DOS provides another driver, called ANSI.SYS, that allows you to control the appearance of your screen and the functions of your keys. With the ANSI.SYS device driver, you can:

- Position the cursor anywhere on the screen.
- Change the color, background, and other qualities of the text on your screen.
- Erase text from the screen.
- Change the character that a key displays or assign a command to a key.

You modify your keyboard and monitor by running ANSI commands. For example, you can use ANSI commands to display a screen of multicolored text, or change a key so that you can type a character that your keyboard does not normally produce. The changes that you make using ANSI.SYS are overridden by many programs. This section describes the most common ANSI commands. For the full set of commands see the *MS-DOS 5.0 User's Reference.*

Understanding ANSI Commands

An ANSI *escape sequence* is a command that you send to your monitor or keyboard. The commands are called escape sequences because they always begin with the escape (ESC) character. They are called ANSI commands because they were developed by the American National Standards Institute (ANSI).

ANSI commands have a very different format from other commands you use with MS-DOS. ANSI commands begin with an ESC character and the square open bracket character ([). These two characters tell the ANSI.SYS driver that what follows is a command and not just a string of characters. Following these two characters are the parameters of the command. Lastly is a single letter command name (for example, the A command moves the cursor up a line). You have to type the command name in either uppercase or lowercase, whichever the command requires.

For example, the ANSI A command moves the cursor up. When you type the A command, you precede the name with a parameter that specifies how many lines to move up. For example the following command moves the cursor up 10 lines:

```
ESC[10A
```

ESC[tells the monitor that this is an ANSI command. The 10 is a parameter that specifies 10 lines, and the A is the command name. The A must be typed in uppercase. Notice that there are no spaces between any of the characters in the command. When ANSI.SYS finds a space in a command, it assumes that the space marks the end of the command.

If an ANSI command has more than one parameter, the parameters are separated by semi colons. For example, the following H command moves the cursor to line 3, character 12:

```
ESC[3;12H
```

The 3 specifies the line and the 12 specifies the character position to move the cursor to. H is the command name. The H must be typed in uppercase. Notice that the H follows the last parameter with no spaces and no semicolon.

Understanding ASCII Codes

Often ANSI commands require ASCII codes as parameters, so it is important that you understand what ASCII codes are. ASCII stands for the American Standard Code for Information Interchange and is a convention developed for representing characters. Originally, ASCII consisted of 127 different codes that represent the English alphabet and punctuation. Currently, most computers recognize 256 codes; the original 127 ASCII codes and an additional 127 codes called the IBM extended character set. The extended character set includes a number of European characters, graphics characters and scientific characters.

Each key on your keyboard has an ASCII code associated with it. When you press a key, MS-DOS determines which key it is and then assigns the ASCII code to it. There are different codes assigned to the key alone and to the key combined with the SHIFT key. In addition, most keys have codes for key combinations using CTRL and ALT combinations. The ASCII codes for keys that display a character (A, S, D, F and so on) are single numbers. For example, the ASCII code for uppercase A is 65. The ASCII code for keys that don't display a character (F1, DEL, UP ARROW, and so on) is a zero and semicolon (0;) followed by a number. For example, the ASCII code for F1 is 0;59.

Depending on the ASCII code MS-DOS assigns, the monitor will display a different character. For example, when you press SHIFT+2, MS-DOS sends an ASCII 64 to the display. The display has a table that tells it what character to display when it receives an ASCII 64. Normally, an ASCII 64 is an At sign (@). For a list of the ASCII codes MS-DOS normally assigns to each key, see xx.

NOTE The ASCII code that MS-DOS assigns to a key is affected by the current code page, and the keyboard and display driver you are using. For more information, see xx.

Running ANSI Commands

To run an ANSI command, you send it to the monitor. To send it to the monitor, you display it in the same ways that you display text. There are a number of ways to display text. The following ways are the best suited to displaying (that is, running) ANSI commands:

- You can use the **prompt** command to enter and test your ANSI commands quickly.
- You can put ANSI commands in an unformatted text file and then enter a type command to display them.
- You can put ANSI commands inside batch files in an **echo** command. When the batch file runs, the ANSI commands are displayed by the **echo** commands.

Running ANSI Commands from a Prompt Command

The **prompt** command is the most convenient way to enter single ANSI commands. It allows you to enter ANSI commands directly from the keyboard. If you enter ANSI commands from the keyboard, you can easily edit them using Doskey.

Doskey allows you to display commands you have previously entered. For more information about Doskey, see xx.

If you press ESC on the keyboard, MS-DOS cancels what you've typed on the command line. Because ANSI commands begin with ESC, you need a way to enter ESC from the keyboard without canceling your command. The prompt command provides a way to do this. To specify ESC with the **prompt** command, type a $e combination. You can type ein uppercase or lowercase.

For example, you can enter the ESC[10A command, which moves the cursor, with the following **prompt** command:

```
prompt $e[10A
```

If you enter this example on your system, you'll notice that it runs the ANSI command but deletes your command prompt. This will happen anytime you use the **prompt** command to run ANSI commands. There are two ways to solve this problem. You can restore your command prompt with another **prompt** command after you are finished entering ANSI commands. Or you can add the characters that restore your system prompt (pg) to the ANSI command. For example, to run the ANSI A command and restore the command prompt, enter the following **prompt** command:

```
prompt $e[10A$p$g
```

This moves the cursor and then creates a command prompt that shows the current drive and directory. Notice that there are no spaces in the command. MS-DOS takes any spaces in a prompt command literally. If there is a space between the $p and $g in a **prompt** command, MS-DOS will display the space as part of the command prompt. The new cursor position becomes the new command prompt position. For more information, see xx,later in this chapter.

Running ANSI Commands from a Text or Batch File

To put an ANSI command in an unformatted text file or batch file, you can use any editing program that saves files as unformatted text and provides a way to enter the ESC character. In Microsoft Word, you can enter an ESC character by typing ALT+27.

If you have a number of ANSI commands that you want to run, it is most convenient to store them in an unformatted text file or batch file. To run the commands, you either display the text file using the **type** command or run the batch file. For example, the following command runs the ANSI commands stored in the text file SCREEN.ANS:

```
type screen.ans
```

Because the monitor interprets everything it receives as either a command or text to be displayed, there must be no spaces or carriage returns between the ANSI commands in your file. If you put carriage returns between your ANSI commands, the monitor will move the cursor to the beginning of the next line and display the command prompt. If you put spaces between you ANSI commands, the monitor will display spaces.

Consequently, a text file containing a series of ANSI commands looks like one continuous line of characters. For example, a file with 10 ANSI commands looks like this:

```
ANSICommand1ANSICommand2ANSICommand3ANSICommand4ANSICommand5ANSICommand
6ANSICommand7ANSICommand8ANSICommand9ANSICommand10
```

If you want to run ANSI commands from a batch file, begin each line with an **echo** command. An **echo** command in a batch file with five ANSI commands would look like this:

```
echo ANSICommand1ANSICommand2ANSICommand3ANSICommand4ANSICommand5
```

NOTE When putting ANSI commands in an **echo** command, remember that after an **echo** command MS-DOS automatically moves the cursor down 2 lines.

In this form a file filled with ANSI commands is very hard to read and edit. One way to avoid this problem is to put each ANSI command on its own line and then remove the carriage returns before you run the file. To make the examples in this section easier to read, ANSI commands are shown one per line.

Changing the Character and Key Displays

In Brief

To change the character a key displays use the ANSI p command. For example, the following command changes SHIFT+4 ($) to an ASCII 172, which is a one quarter sign (1/4):

```
ESC["$";172p
```

This form of the p command has two parameters: the character or ASCII code assigned to the key you want to change, and the character or ASCII code you want the key to have.

You can change the ASCII code MS-DOS assigns to a key using the ANSI p command. Specify ASCII codes by typing their number or by typing their character enclosed in quotation marks.

For example, to change SHIFT+2 to display a plus sign (+), use the following **p** command.

```
esc["@";"+"p
```

Notice that there are no spaces in the command and that the command name (**p**) follows the last parameter with no semicolon. Rather than typing the characters you can also use their ASCII codes as in this command:

```
esc[64;43p
```

When you use ASCII codes instead of characters, you do not need quotation marks. After you enter this command, whenever you press SHIFT+2 MS-DOS displays a plus sign (+).

To return the key to its original code, type its original code as the first and second parameter as in this command:

```
esc[64;64p
```

Because the key has been reassigned, you have to type its ASCII code.

There are 256 ASCII codes. Many of the codes are not normally assigned to a key. To assign a normally unassigned code to a key, use the character's ASCII code in a **p** command.

For example, if you want the SHIFT+2 key to display a check mark (✓), which is ASCII 251, rather than its normal At sign (@), use the following ANSI **p** command:

```
esc["@";251p
```

Because the At sign (@) normally exists on the keyboard, you have two ways to specify it: by typing an At (@) or by typing its ASCII code. The check mark does not normally exist on the keyboard, so you must specify it by typing its ASCII code.

Once a key has a different ASCII code, you can no longer type its character to specify it. To return the SHIFT+2 key to its normal assignment, enter its normal ASCII code as both the first and second parameter as in this command:

```
esc[64;64p
```

Assigning Commands to a Key

In Brief

To assign a command or sequence of commands to a key, use the ANSI **p** command. For example, the following command assigns the command C:\WINDOWS\WIN386 to the CTRL+W key (ASCII 23):

```
esc[23;"c:\windows\win386";13p
```

This form of the **p** command can have any number of parameters. The first parameter always specifies the character or ASCII code of the key you want to change. The remaining parameters specify the operations you want the key to perform. The command in the above example assigns two separate operations to CTRL+W: Type the text C:\WINDOWS\WIN386, and issue an ASCII 13, which is the ENTER key.

In addition to assigning a single new character to a key, the **p** command also lets you assign any number of characters to a key. For example, you can assign the word *percent* to the percent key (SHIFT+5) with the following **p** command:

```
esc["%";"percent"p
```

After this command, when you press SHIFT+5 MS-DOS displays the word *percent*.

You can use this capability of the **p** command to assign one or more MS-DOS commands to a key. To run a command, you type the command and press ENTER. Likewise, when you assign a command to a key, you have to tell MS-DOS to type the command and then issue an ENTER. To type the command, enclose it in quotation marks; to issue an ENTER, specify the ASCII code for the ENTER key (13). For example, use the following **p** command to assign the command **dir /s /p** to the SHIFT+7 ampersand (&) key:

```
esc["&";"dir /s /p";13p
```

You include two parameters for each command you want to assign to the key: the text of the command and an ENTER character. Notice that you can type spaces inside a set of quotation marks.

To set the F1 key (ASCII code 0;59) so that it changes the current directory and starts the WORD.EXE program, enter the following **p** command:

```
esc[0;59;"cd \edit";13;"word";13p
```

The command has four parameters: two commands enclosed in quotation marks and two ENTER codes (ASCII 13).

NOTE Many applications override key assignments.

Moving the Cursor

In Brief

To move the cursor, use the **A**, **B**, **C**, **D**, **H**, or **f** command.

The following **A** command moves the cursor up two lines:

```
esc[2A
```

The following **B** command moves the cursor down 3 lines:

```
esc[3B
```

The following **C** command moves the cursor right 10 characters:

```
.::[10C
```

The following **D** command moves the cursor left 5 characters:

```
esc[5D
```

The **H** and f commands are equivalent. Both of the following commands move the cursor to line 6, character 8:

```
esc[6;8H
esc[6;8f
```

When a command has finished, the cursor always returns to the position immediately to the right of the command prompt. If you run ANSI commands with a **prompt** command, you can change the cursor's position at the end of a command. If you run ANSI commands with text or batch files, you can change the position of the cursor in order to display text at a certain location. At the end of your ANSI command, however, the cursor will return to the command prompt.

You use the following ANSI cursor movement commands inside or outside a **prompt** command:

- The A command moves the cursor up from its current position.
- The **B** command moves the cursor down from its current position.
- The C command moves the cursor right from its current position.
- The **D** command moves the cursor left from its current position.
- The **H** command moves the cursor to the line and character you specify (you can type f rather than **H** for the same result).

The A through **D** commands are relative. They move the cursor a specified number of characters or lines from its current position. The H and f commands are absolute. Regardless of where the cursor is currently, they move it to the location specified. If the cursor is already at the edge of the display, and you send a command that tells the monitor to move the cursor off the screen, the command will be ignored and the command prompt will remain where it is.

Positioning the Command Prompt

If you run an ANSI H command with the **prompt** command, the cursor position you specify becomes the permanent position of the command prompt. For example, the following command moves the cursor to the upper left corner of the display (the "home" position) and sets this to the new command prompt location:

```
prompt ESC[H$p$g
```

ESC[H moves the cursor to the home position, and pg displays the usual command prompt (current drive and directory and a greater-than sign). Until you enter another **prompt** command, MS-DOS will display the command prompt in the upper left most position of your screen.

If you position the cursor with the **H** command, you specify an absolute location where the command prompt will be displayed. If you position the command prompt with the other cursor movement commands, you specify where the command prompt will be displayed *in relation to its position at the end of a command.*

Recall that as MS-DOS sends command output to the monitor, it displays each line below the previous line. At the end of a command, MS-DOS moves the cursor down two lines and displays the command prompt. If your command prompt includes an H command, no matter where the cursor ended up at the end of the command, it will always move to the same position on the display. On the other hand, if your command prompt includes an **A**, **B**, **C**, or **D** command, MS-DOS moves the cursor up, down, or sideways from its current position when a command is finished.

For example, if you include the following C command in your command prompt, MS-DOS displays the command prompt beginning at the fourth character on a line:

```
prompt $e[3C$p$g
```

Before the prompt is displayed, MS-DOS always moves the cursor to the beginning of the line, two lines down. In this case, when MS-DOS displays the command prompt, the ESC[3C moves the cursor over three characters and the pg tells it to display the usual command prompt. The result is that the command prompt appears shifted over three characters.

Creating a Display

If you combine cursor movement commands with text, you can create custom displays. For example, the following ANSI commands in a batch file display an introductory screen to a program:

```
echo off
cls
```

```
echo
esc[7C*Welcome*
esc[2B*to the*
esc[2B
esc[2C*Database Utility*
esc[12;1H(For help press F1)
prompt $e[10;1HEnter a Command:cls
```

NOTE For clarity, each ANSI command is shown here on a new line. However, before you run this batch file there must be no spaces or carriage returns between the ANSI commands contained in the **echo** command. For more information, see xx, earlier in this chapter.

The **cls** command clears the screen and moves the cursor to the home position. The remaining lines position the cursor and display text. The screen produced by this batch file looks like this:

```
*Welcome*

  *to the*

    *Database Utility*



Enter a command:

(For help press F1)
```

The command prompt remains on line 10 of the display until you specifically reposition it.

Changing the Attributes of the Text

In Brief

To set the attributes of text, use the ANSI **m** command. For example, the following command makes text bold:

```
esc[1m
```

This command makes text magenta and bold:

```
esc[1;35m
```

The **m** command can have as many parameters as you want to specify. See the following paragraphs for a complete list of the attributes you can set.

You can use the ANSI **m** command to change the way text appears on your screen. After you set an attribute, any new text that is displayed has the attribute. For example, after you set the bold attribute any new text that is displayed is bold.

To turn all attributes off and display plain text on a black background, use a zero for the parameter as in the following example:

```
ESC[0m
```

You can set as many attributes as you like with a single **m** command. For example, the following command sets text to be blinking (5), and red (31) with a blue background (44):

```
ESC[5;31;44m
```

The order in which you type the parameters is not important. However, each parameter must be separated from the others with a semicolon.

There are three kinds of attributes you can give text: form, color, and background. The form attributes specify whether the text will be bold, underlined, blinking, or hidden. The color attributes specify which color the text will be. The background attributes specify the color behind the text. If you specify an attribute that your system doesn't support, MS-DOS ignores your request and no change occurs on the screen.

Changing the Kind of Text

The following form attributes control the kind of text displayed:

1	Bold text
2	Non-bold text
4	Underlined text
5	Blinking text
8	Hidden text

The underline attribute functions only with monochrome monitors. The hidden text attribute tells the monitor to move the cursor as text is displayed but not to actually display the text.

For example, the following ANSI commands display the word *WARNING* in bold, blinking letters at the current cursor position:

```
ESC[1;5mWARNING
ESC[0m
```

The second **m** command resets all attributes so that future text is not displayed bold and blinking.

The following **prompt** command makes the greater-than sign in your command prompt bold:

```
prompt $p$e[1m$g$e[0m
```

Before the $g, bold is turned on, and after the $g, bold is turned off.

Changing the Color of Text

The following attributes specify the color of text:

30	Black text
31	Red text
32	Green text
33	Yellow text
34	Blue text
35	Magenta text
36	Cyan text
37	White text

For example, the following command displays the word *WARNING* in red, bold, blinking letters:

```
esc[31;1;5mWARNING
esc[0m
```

As before, the **0m** command resets the attributes when the text has been displayed.

The following **prompt** command makes the drive and directory in your command prompt bold blue and the greater-than sign bold red:

```
prompt $e[1;34m$p$e[31m$g$e[0m
```

Before the **$p**, bold and blue are turned on, before the $g, red is turned on, and after the $g all the attributes are reset.

Changing the Background of Text

The following attributes control background color:

40	Black text
41	Red text
42	Green text
43	Yellow text
44	Blue text
45	Magenta text
46	Cyan text
47	White text
7	Reverse video text

For example, the following command displays the word *WARNING* in red, bold, blinking letters on a yellow background:

```
ESC[43;31;1;5mWARNING
ESC[0m
```

Reverse video text has a white background. When you set this attribute, the text background changes to white. The color of the text remains the same unless it was white (in which case it changes to black).

NOTE The color and background parameters meet the ISO 6429 standard.

Chapter 13

Optimizing Your System

13

Optimizing is customizing your system so that it uses its resources most efficiently for the tasks you usually perform. Typically, optimizing involves improving one or more aspects of your system's performance, but sometimes sacrificing something else. For example, you might improve your system's speed, but have less memory for applications as a result.

In the MS-DOS environment, optimizing your system usually means balancing speed against memory. Typically, you have one of two goals:

- To improve your system's speed as much as possible, while still retaining enough memory to run the applications you need.

  You might want to improve your system's speed if there is sufficient memory to run the applications you need, but you want those applications to run faster.

- To free as much memory as possible for applications, even if doing so slows down your system a bit.

  You might want to free up memory, even at the cost of some speed, if there is not enough memory available to run certain applications.

This chapter briefly describes your system's resources and how they relate to system performance. The chapter then explains the different utilities and procedures you can use to improve speed, or to free up memory for applications.

NOTE If you are using Microsoft Windows version 3.0 or later, you should follow the optimization procedures in the *Microsoft Windows User's Guide* instead of the procedures in this chapter, since Windows manages memory and other system resources very differently from the way MS-DOS does.

What Are System Resources?

Your system's resources determine its limitations. In the MS-DOS environment, the most important system resources are memory and disk space. The available resources can affect:

- Which applications you can run.
- How fast applications run.
- How much data an application can work with at a time.
- How much data you can store from one session to the next.

The sections that follow explain each resource and how MS-DOS uses it.

Understanding Memory

Memory provides temporary storage for applications and data. It exists on your computer's main system board or on add-in memory boards. All applications must be loaded into memory in order to run.

In general, the more memory you have, the more data you can store in memory at a time. Some applications require more memory than others. You can increase the amount of memory on your system by plugging a memory board into a slot inside the machine. For example, you might add a 2MB memory board to a system that already has 1MB of memory on its main system board; the system would then have a total of 3MB of memory.

Your system may have up to three different kinds of memory:

- Conventional memory
- Extended memory
- Expanded memory

To find out what kind of memory your system has, and how much, use the **mem** command. (See the *Reference* for additional information on **mem**.)

The sections that follow explain each type of memory.

Conventional Memory

Conventional memory exists on all 8086, 8088, 80286, 80386, and 80486-based computers. Most computers have at least 256K of conventional memory, and can accommodate up to 640K of conventional memory.

When you start MS-DOS, it uses up some of your computer's conventional memory. Next, MS-DOS starts the device drivers listed in your CONFIG.SYS file and the memory-resident utilities specified in your AUTOEXEC.BAT file. These device drivers and utilities use up additional memory. The memory that is left over is available for running applications.

All applications require some conventional memory.

Extended Memory (XMS)

One way to add more memory to your system is to install *extended memory*. Extended memory is general-purpose memory beyond 640K on 80286, 80386, and 80486 computers. In effect, it is a simple extension of your computer's conventional memory. (Many 80286 and 80386 computers come with 640K of conventional memory and 384K of extended memory, for a total of 1MB of memory.)

Extended memory is fast and efficient for applications to use. However, to take advantage of extended memory, an application must be designed to use it; many applications are not designed to do so. If you do not use applications that can take advantage of extended memory, you might want to use your system's extended memory to run utilities that improve its speed. For example, if you have extended memory, you can speed up your system by using some of that memory for the SMARTDrive utility.

To use extended memory efficiently, you should install a special program called an *extended memory manager*. An extended memory manager "administrates" the use of your system's extended memory, so that several programs do not try to use the same memory at the same time. The extended memory manager also makes it much easier for programs to use extended memory. MS-DOS includes the extended memory manager HIMEM.SYS. (You must install HIMEM.SYS if you want to run MS-DOS in extended memory. See "Running MS-DOS in Extended Memory," later in this chapter, for details.) HIMEM.SYS conforms to the Extended Memory Specification (XMS), which specifies a standard way for programs to use extended memory cooperatively.

Extended memory is the best choice for memory expansion if you are using or plan to use Microsoft Windows 3.0 or later, since Windows works best with extended memory. Another advantage of choosing extended memory is that MS-

DOS can run in extended memory, leaving more conventional memory available for applications.

NOTE If your 80386 or 80486 system has extended memory, but you use applications that can take advantage of expanded memory, you may want to install EMM386. EMM386 is an expanded memory emulator; basically, it uses extended memory to provide expanded memory for applications. See the following section for information about expanded memory. For information on EMM386, see "Using the EMM386 Expanded Memory Emulator," later in this chapter.

Expanded Memory (EMS)

Another way to add memory in excess of 640K to your system is to install *expanded memory*. Most personal computers can accommodate expanded memory. Expanded memory exists separately from your system's conventional and extended memory.

To use expanded memory, you must install a special program called an *expanded memory manager*, which comes with the expanded memory board. The expanded memory manager is necessary because, in order to gain access to expanded memory, applications must request it from the expanded memory manager. The expanded memory manager and the expanded memory board conform to the Expanded Memory Specification (EMS), which specifies how programs can make use of expanded memory.

Some applications are unable to use expanded memory because they were not designed to interact with an expanded memory manager. However, because expanded memory was introduced before extended memory, more applications are designed to use expanded than extended memory.

Because an expanded memory manager gives applications access to only a limited amount of expanded memory at a time, expanded memory can be slower and more cumbersome to use than extended memory.

Some expanded memory boards, such as the AST RAMpage! ® AT or Intel Above ™ Board/AT, can be set up with extended memory, with expanded memory, or with both.

Understanding Disk Space

Disks provide long-term and temporary storage for program files and data files. The most common types of disk media are floppy disks and hard disks. Applications and other files are stored on these media magnetically, much as information is stored on a cassette tape.

It's best if there is some disk space left after you have installed all the software and data files you need. There are two reasons you might need free disk space:

- You need disk space to save documents and other data files.
- Some applications use available disk space to store temporary files and data while they're running.

It's often useful to know how much disk space is available on your system, since the amount of free disk space can affect your ability to store files and data. With some applications, the amount of available disk space can even affect how well the application runs.

Use the MS-DOS commands **chkdsk** or **dir** to check the amount of free disk space. See Chapter ??, "What title," for information on how to use these commands.

Improving Your System's Speed

Your system's speed affects how fast applications can perform tasks such as reading and saving files, displaying information on the screen, and so on.

In the MS-DOS environment, there are two ways to improve your system's speed:

- By improving the efficiency of your hard disk.

  The organization of the files on your hard disk can affect how fast your computer runs. This chapter explains several procedures you can perform to improve your hard disk's organization. None of these procedures take up additional memory. For more information about these procedures, see "Improving the Efficiency of Your Hard Disk," later in this chapter.
- By installing one or more utilities that are designed to improve speed.

  Often, the most effective way to speed up your system is to install a utility such as SMARTDrive. MS-DOS comes with several such utilities. These can improve speed dramatically, but they use up some memory. For more information about these utilities, see "Using Speedup Utilities," later in this chapter.

NOTE If your system has additional extended or expanded memory that your applications do not need, you might want to use that memory for utilities that improve your system's speed.

Certain speedup methods work better for some systems than for others. The best way to speed up your system will depend on how much memory your system has. In addition, the type of applications you use can determine which speedup methods work best for you.

Database applications and language compilers use the hard disk differently from the way most applications do. Therefore, the best speedup methods for someone who normally uses database applications or language compilers will differ from the best methods for someone who uses mainly a word processing application.

The following table lists methods for speeding up your system, explains when you would want to use each method, and lists the types of applications that each method speeds up.

Method	When to Use It	What It Speeds Up
Use the **buffers** command.	On most systems.	All applications.
Use the **buffers** command to specify a secondary cache.	If you use applications that a secondary cache can speed up. Do not use in conjunction with SMARTDrive.	Compilers and other applications that read files in small pieces.
Install Fastopen.	If you use databases or compilers.	Databases and compilers.
Install SMARTDrive.	If your system has a hard disk and extended or expanded memory that is not needed by applications. Do not use in conjunction with a secondary cache.	All applications.
Install RAMDrive.	If your system has extended or expanded memory, and you use applications that RAMDrive can speed up.	Applications that use temporary files, or applications that you start often.
Structure the **path** command efficiently.	If your system takes a long time to respond when you type a command.	Starting applications.
Reorganize (compact) your hard disk.	Periodically.	All applications, to some extent. Can particularly improve application startup time.
Adjust your hard disk's interleave.	If you are using SMARTDrive and your hard disk still responds sluggishly	All applications.

Method	When to Use It	What It Speeds Up
Make more space available on your hard disk using the methods described in "Improving the Efficiency of Your Hard Disk" later in this chapter	If your system is low on disk space.	All applications, to some extent.

The following sections explain each method in detail.

Using the Buffers Command

The **buffers** command specifies the number of buffers that MS-DOS reserves for file transfers. (For a detailed explanation, see Chapter 12, "Customizing Your System.")

Basic Recommendations

The higher the number of buffers (up to about 50), the faster your system runs. However, past a certain value, increasing the number of buffers does nothing but use up memory.

When optimizing your system for speed, you would normally want to specify the highest number of buffers that is useful for your system. This number depends on the size of your hard disk. The following table lists the highest efficient buffer sizes for different hard disks:

Hard disk size	Buffer size
Less than 40 MB	20
40 to 79 MB	30
80 to 119 MB	40
More than 120 MB	50

The following command specifies 40 buffers — an optimum number for a system with a 110-MB hard disk:

```
buffers = 40
```

NOTE When calculating the default number of buffers, MS-DOS bases the number of buffers on how much conventional memory your system has, rather than on the size of the hard disk. The default that MS-DOS calculates is a minimum number of buffers. The numbers in the preceding table are larger in order to increase your system's speed.

Using the Secondary Cache

A secondary cache can be useful when you are not using SMARTDrive. It is more useful for database applications and language compilers than for other applications. The secondary cache can also make it faster to load programs.

You specify a secondary cache as part of the **buffers** command.

Basic Recommendations

- Do not specify a secondary cache if you have installed a disk-caching utility such as SMARTDrive.
- Normally, if you specify a secondary cache, you should set the secondary cache to 8.

The following command specifies 30 buffers and a secondary cache of 8:

```
buffers = 30,8
```

Using Fastopen

The **fastopen** program can speed up access to files and directories. **Fastopen** keeps track in memory of the location of files and directories you open. This makes subsequent access to those files much faster. **Fastopen** is particularly useful if you use applications such as database programs that repeatedly open and close files.

Basic Recommendations

- Use **fastopen** if you use databases or compilers and have some memory to spare.
- Give **fastopen** access to 1 file for every MB of hard disk space. For example, if you have a 40-MB hard disk, you would allow **fastopen** to work with up to 40 files at once.
- Experiment! If you do not notice any improvement in speed with **fastopen**, then the applications you are using probably do not perform the type of disk access that **fastopen** can speed up. In that case, stop using **fastopen** in order to free memory for other uses.

Advantages

- Improves speed for programs that repeatedly open and close files, such as database software and language compilers.
- Does not require expanded or extended memory.
- Can use expanded memory.

Disadvantages

- Uses up some conventional memory.
- Cannot use extended memory.
- Does not improve performance of applications that do not repeatedly open and close files. For applications other than database programs and compilers, **buffers** or SMARTDrive might improve speed more effectively than **fastopen.**

There are three ways to use **fastopen:**

- Add **fastopen** to your CONFIG.SYS file using the **install** command. (You must use this method in order to run **fastopen** in expanded memory.)
- Include the **fastopen** command in your AUTOEXEC.BAT file.
- Type the **fastopen** command at the MS-DOS prompt.

To install **fastopen** in your CONFIG.SYS file, you would include a command like the following:

```
install = c:\dos\fastopen.exe c:=40 /x
```

This command runs **fastopen** in expanded memory, and tells it to work with the files on drive C:. It allows **fastopen** to work with as many as 40 files at once — the optimum number for a 40-MB disk drive.

For more details on **fastopen,** see the *Reference.*

Using SMARTDrive

SMARTDrive is a disk-caching program for computers that have a hard disk and extended or expanded memory. Disk-caching programs can reduce the amount of time your computer spends reading data from your hard disk. (If you use Windows version 3.0 or later, SMARTDrive can provide a significant increase in speed without taking memory away from Windows. This is because SMARTDrive, unlike other disk-caching programs, cooperates with Windows to provide the most effective use of your system's memory.)

SMARTDrive saves information read from your hard disk in your computer's expanded or extended memory. This area of memory is SMARTDrive's *cache.* When an application tries to read that information from the hard disk, SMARTDrive supplies the information directly from its cache in memory instead. SMARTDrive always copies new or modified information to the hard disk, so there is no danger of losing information when you turn off your computer.

You install SMARTDrive by adding the SMARTDrive command line to your CONFIG.SYS file.

Basic Recommendations

Because the optimum settings for SMARTDrive depend on the applications you run and your system configuration, there is no single "best setting" for SMARTDrive. After installing SMARTDrive, you might want to experiment to find the optimum settings for your system.

The following are basic recommendations you can start with:

- You should use SMARTDrive if your system has a hard disk and at least 512K of extended memory or 256K of expanded memory. Many applications run much faster with SMARTDrive.
- If your system has extended memory, tell SMARTDrive to use extended memory.
- If your system has expanded memory, tell SMARTDrive to use expanded memory by adding the /A switch to the SMARTDrive command line.
- If your system has both extended and expanded memory, tell SMARTDrive to use whichever type of memory is more abundant on your system.
- To start with, give SMARTDrive as large a cache as possible, up to 2MB. (The larger the cache, the more memory SMARTDrive uses.) If an application won't run because there is not enough expanded or extended memory left over, gradually reduce the size of the cache until the application is able to run properly. "Specifying the Size of the SMARTDrive Cache," later in this chapter, explains more about figuring out the optimum size for SMARTDrive's cache.
- If possible, compact your hard disk regularly. SMARTDrive runs best if the files on your hard disk are not fragmented.

Advantages

- Improves speed on all systems that have expanded or extended memory.
- Fairly easy to adjust.

Disadvantages

- Uses up some conventional memory.
- Requires either extended or expanded memory.

NOTE Do not use SMARTDrive in conjunction with other disk-caching programs. In addition, Windows/386 ™ version 2.x will not run with SMARTDrive version 3.0.

Installing SMARTDrive

When you install MS-DOS, Install copies the SMARTDRV.SYS file to your MS-DOS directory. To install SMARTDrive, you add a **device** command for SMARTDrive to your CONFIG.SYS file. (This section refers to that **device** command as *the SMARTDrive command line.*)

The SMARTDrive command lines tells MS-DOS:

- Where to find the SMARTDRV.SYS file
- The size of SMARTDrive's memory cache
- Whether SMARTDrive should use extended or expanded memory (optional). By default, SMARTDrive uses extended memory.

The following is a typical SMARTDrive command line:

```
device = c:\dos\smartdrv.sys 1024
```

This command tells MS-DOS that SMARTDRV.SYS is in the \DOS directory on drive C. It runs SMARTDrive in extended memory, since SMARTDrive runs in extended memory by default. It specifies a cache size of 1024K (1 MB).

IMPORTANT In order for SMARTDrive to run, your CONFIG.SYS file must contain a command line for the appropriate memory manager.

If you want SMARTDrive to use extended memory, your CONFIG.SYS file must contain a command line for HIMEM.SYS. The SMARTDRV.SYS command line must come after the HIMEM.SYS command line.

If you want SMARTDrive to use expanded memory, your CONFIG.SYS file must contain a command line for the expanded memory manager that came with your memory board. The SMARTDRV.SYS command line must come after the command line for your expanded memory manager.

For additional information about setting up and optimizing SMARTDrive on your system, see the following sections.

Specifying the Size of the SMARTDrive Cache

The size of SMARTDrive's disk cache affects SMARTDrive's efficiency. In general, the larger the cache, the less often SMARTDrive needs to read data from the disk. The most efficient size for the cache size is around 2 MB; increasing the cache over 2 MB will probably not provide faster performance than a 2-MB cache.

The SMARTDrive cache size is the first numeric parameter on the SMARTDrive command line.

NOTE The SMARTDrive command line also has an optional second numeric parameter which limits how much Windows can reduce the cache size. This parameter matters only if you use Windows version 3.0 or later. For additional information, see the *Microsoft Windows User's Guide.*

The following SMARTDrive command line tells SMARTDrive to create a 1024K cache in extended memory:

```
device = c:\dos\smartdrv.sys 1024
```

When creating its cache, SMARTDrive rounds the specified cache size down to the nearest multiple of the largest disk track size. In other words, the actual size of the SMARTDrive cache depends on how many disk tracks will fit in the cache. When you start SMARTDrive, it determines how many disk tracks can fit in the amount of memory you specified for the cache. Thereafter, SMARTDrive will cache no more than that number of disk tracks at a time. For example, if the track size on your hard disk is 10K and you have set up a 256K cache, then 25 tracks will fit in the cache. SMARTDrive will then cache no more than 25 tracks at a time, and will create a cache that uses 250K, not 256K, of memory. In general, SMARTDrive runs most efficiently with disks that have small tracks.

NOTE On most systems, the SMARTDrive driver uses about 15K or more of conventional memory in order to run. The amount depends on the track size of your hard disk(s) and the size of the cache you specify.

Basic Recommendations

Because the optimum cache size for SMARTDrive depends on how you use your system, there is no single "best setting." You should experiment to find the best cache size for your system.

The following are basic recommendations you can start with:

- Set the cache size to between 256K and 2048K. (Set it as large as possible within that range.)
- Setting the cache size to less than 256K is not a good idea, since SMARTDrive will probably not be able to cache enough information to be effective.
- Setting the cache size to larger than 2048K might not be the best use of your system's memory. Although, in general, a larger cache is faster, you get less return on your memory investment as the cache size increases above 2048K. For example, increasing a 256K cache to 512K might improve your system's speed by 20 percent. However, increasing a 2048K cache the same amount, from 2048K to 2304K, might improve speed by only 2 percent.

Putting the SMARTDrive Cache in Extended or Expanded Memory

SMARTDrive can use either extended or expanded memory for its disk cache. By default, it uses extended memory. To tell SMARTDrive to put its cache in expanded memory instead, add the /A switch to the end of the SMARTDrive command line.

Basic Recommendation

- You should give SMARTDrive extended memory unless your system has only expanded memory. In that case, tell SMARTDrive to put its cache in expanded memory.

Putting the Cache in Extended Memory

If your computer has extended memory, you'll probably want to put SMARTDrive's cache in extended memory, and give it as much memory as possible (up to about 2MB). (This is particularly effective if you use Microsoft Windows version 3.0 or later, since Windows can shrink SMARTDrive's cache to release memory for its own use. Windows and SMARTDrive then cooperate to provide both effective disk caching and efficient memory management.)

NOTE To run SMARTDrive in extended memory, the SMARTDrive command line must appear in your CONFIG.SYS file after the HIMEM command line.

The following command lines install HIMEM.SYS and SMARTDrive, and give SMARTDrive extended memory:

```
device = c:\dos\himem.sys
device = c:\dos\smartdrv.sys  1024
```

This command line tells MS-DOS to look for the SMARTDRV.SYS file in the directory C:\DOS. It gives SMARTDrive 1024K (1MB) of extended memory, so the cache size is 1024K.

Putting the Cache in Expanded Memory

If your computer has expanded memory and you can spare some for SMARTDrive, you'll want to put SMARTDrive's cache in expanded memory. (If you use applications that need expanded memory, be sure to leave enough expanded memory for those applications.)

NOTE To use expanded memory, SMARTDrive's command line must appear in your CONFIG.SYS file after the command line that starts your system's expanded memory manager.

The following command line puts the SMARTDrive cache in expanded memory:

```
device = c:\dos500\smartdrv.sys  2048   /a
```

This command line tells MS-DOS to look for the SMARTDRV.SYS file in the \DOS500 directory on drive C. It gives SMARTDrive 2048K (2MB) of expanded memory, so the cache size is 2048K.

NOTE It is not a good idea to put the cache in expanded memory provided by EMM386.EXE. EMM386 uses extended memory to emulate expanded memory that applications can then use. Although SMARTDrive can also use this emulated expanded memory, it might not speed up your system as much as if you gave it real physical memory.

Using RAMDrive

RAMDrive is a memory-resident utility that lets you use part of your system's memory to emulate a very fast, temporary disk drive. This memory area is called a *RAM disk* because it exists in memory (RAM). RAM disks are much faster than hard disks because your computer can read information faster from memory than from a physical disk. A RAM disk appears to be a normal disk drive; you can use it just as you would any disk drive. One important difference between a real disk drive and a RAM disk is that, since it exists only in memory, information on a RAM disk is lost when you turn off or reset your computer.

Basic Recommendations

- Install RAMDrive only if you really need a RAM disk. In many cases, using the same amount of memory for SMARTDrive will improve your system's speed more than RAMDrive would.
- If your system has plenty of memory but does not have a hard disk drive, you should install RAMDrive and give as much memory as possible.
- If you often run applications that use many small temporary files, RAMDrive might improve your system's speed more than SMARTDrive. In that case, install RAMDrive and set your TEMP environment variable to the RAM disk. For additional information, see "Using the TEMP Environment Variable with RAMDrive" later in this chapter.
- List your RAM disk first in your **path** command.

Advantages

- Provides you with a very fast disk drive.
- Provides additional disk space for temporary storage. This can be extremely useful on a system without a hard disk.
- Fairly easy to set up.

Disadvantages

- Uses additional memory, which can cut down your system's speed and capacity.
- When you turn off your computer, it does not save information that is stored on the RAM disk. Because of this, the RAM disk is best for storing temporary files or copies of application files, not for saving data files that may change.
- Whenever you restart your computer, the RAM disk is re-created. This means you must recopy information (except temporary files) to your RAM disk each time you start your computer.

WARNING Do not use the RAM disk to store data files. Whenever you turn off or reboot your computer, all information on the RAM disk is erased. This means that, in the case of a system crash or a power outage, you could lose any data files stored on the RAM disk.

Installing RAMDrive

When you install MS-DOS, Install puts a copy of the RAMDRIVE.SYS file in your MS-DOS directory.

To set up RAMDrive, add a **device** command for RAMDRIVE.SYS to your CONFIG.SYS file. The RAMDrive command line tells MS-DOS:

- Where to find the RAMDRIVE.SYS file.
- How much memory to allocate to RAMDrive.
- Whether RAMDrive should use conventional, expanded, or extended memory.

The following is a typical RAMDrive command line:

```
device = c:\dos\ramdrive.sys 512 /e
```

This command line tells MS-DOS that RAMDRIVE.SYS is in the \DOS directory on drive C. The command line gives RAMDrive 512K of extended memory; it uses the /e switch to tell RAMDrive to use extended memory.

IMPORTANT In order for RAMDrive to use your system's extended or expanded memory, your CONFIG.SYS file must contain a command line for the appropriate memory manager.

If you want RAMDrive to use extended memory, your CONFIG.SYS file must contain a command line for HIMEM.SYS. The RAMDRIVE.SYS command line must come after the HIMEM.SYS command line.

If you want RAMDrive to use expanded memory, your CONFIG.SYS file must contain a command line for the expanded memory manager that came with your memory board. The RAMDRIVE.SYS command line must come after the command line for your expanded memory manager.

The following command line tells RAMDrive to run in expanded memory, and gives RAMDrive the maximum amount (4096K) of expanded memory that RAMDrive can use:

```
device = c:\dos\ramdrive.sys 4096  /a
```

The command line tells MS-DOS to look for the RAMDRIVE.SYS file in the \DOS directory on drive C.

NOTE It is not a good idea to put the RAM disk in expanded memory provided by EMM386.EXE. EMM386 uses extended memory to emulate expanded memory that applications can then use. Although RAMDrive can also use this emulated expanded memory, it will not be as efficient as if you gave it real physical memory.

Running Applications from Your RAM Disk

If you start certain applications frequently, you might want to run those applications from your RAM disk rather than from a physical disk. Starting an application from a RAM disk can be up to 3 times faster than starting the same application from a physical disk. It can be particularly useful if you normally start applications from a floppy disk.

▶ To run an application from your RAM disk:

1. Install RAMDrive as described in the preceding section.

   The drive letter of your RAM disk should be the letter after that of your last physical drive. For example, if your last disk drive is named C:, your RAM disk would be D:. If you have three hard disk drives named C:, D:, and E:, your RAM disk would be F:.
2. Copy the application's executable file(s) to the RAM disk.
3. Start the application from the RAM disk as you would from any disk drive. (Depending on the application, you might need to first make the RAM disk your current disk drive.)

You will need to recopy the application to the RAM disk each time you reboot your computer, since the RAM disk is created anew whenever you reboot. If you use the application frequently, you might want to add a **copy** command to your AUTOEXEC.BAT file. For example, to copy Microsoft Excel to a RAM disk named F: whenever you restart your computer, you would add a command like the following to your AUTOEXEC.BAT file:

```
copy c:\excel\excel.exe f:\
```

Using the TEMP Environment Variable with RAMDrive

Many applications use temporary files to store data while they're running. Some of these applications store temporary files in the directory specified by the TEMP environment variable. For information about how your applications store temporary files, consult the applications' documentation.

You set the TEMP variable using the **set**command. (Typically, the **set** command is in your AUTOEXEC.BAT file.) For example, the following command line sets the TEMP variable to the C:\TEMPFILES directory:

```
set temp = c:\tempfiles
```

Setting the TEMP variable affects only those applications that check for the value of TEMP. The location you specify for temporary files can affect the speed of applications that use the TEMP variable. For example, if the TEMP variable specifies a relatively slow hard disk, applications that store temporary files on that drive might run at less than optimum speed.

Basic Recommendations

- Set the TEMP variable to your RAM disk.

  Because it's much faster to read information from memory than from a hard disk, an application that uses temporary files will usually run faster if it stores its temporary files on a RAM disk. Since most applications delete their temporary files when they're finished using them, you don't need to worry about saving copies of those files before turning off your computer.
- Set the TEMP variable to a subdirectory, not to the root directory.

  MS-DOS allows you to create only a limited number of files in the root directory of a disk, but lets you create as many files as you need within a subdirectory.
- Make sure that the disk to which you set the TEMP variable has enough free space for the temporary files your application creates. Consult the application's documentation for more information about how the application uses temporary files.

Using the HIMEM Extended Memory Manager

HIMEM is an extended memory manager—a utility that "administrates" the use of your system's extended memory so that no two applications can use the same memory at the same time.

Advantages

- Makes extended memory available to applications that use extended memory according to XMS (the Extended Memory Specification).
- Prevents system errors that can result from applications making conflicting memory requests.
- Lets you run DOS in extended memory to conserve conventional memory.
- Compatible with Windows version 3.0 or later.

Disadvantages

- Uses up a small amount of conventional memory.
- Might not be compatible with some software.

Basic Recommendations

- You should install HIMEM if you have an 80286, 80386, or 80486 computer with extended memory.

NOTE Windows/386 ™ version 2.x will not run in conjunction with HIMEM version 3.0.

You install HIMEM by adding the HIMEM command line to your CONFIG.SYS file. The HIMEM command line must come before any command lines for applications or device drivers that use extended memory.

The HIMEM command line tells your computer where to find the HIMEM program file, controls how HIMEM manages memory, and specifies your system type.

The following command line runs HIMEM using the default values:

```
device = c:\dos\himem.sys
```

The command line tells MS-DOS to look for the HIMEM.SYS file in the directory C:\DOS. Because this command line does not include additional switches, HIMEM uses the default values for those switches. (for information on HIMEM's additional switches, see the *MS-DOS 5.0 User's Reference.*)

Installing the EMM386 Expanded Memory Emulator

An *expanded memory emulator* is a utility that uses extended memory to simulate expanded memory. Applications can then use that simulated expanded memory just as if it were physical expanded memory. MS-DOS includes the expanded memory emulator EMM386.EXE (for use on 80386 and 80486 systems only). (When you install MS-DOS, Install copies the EMM386.EXE file to your MS-DOS directory.) You should use EMM386.EXE only if necessary.

You might want to install EMM386.EXE if you use applications that require expanded memory, but your 80386 or 80486 system includes only extended memory.

NOTE If you use Microsoft Windows 3.0 or later, you may not need to use EMM386 even if you run applications that need expanded memory. When running in 386 enhanced mode, Windows can simulate expanded memory for applications that need it. See the *Windows User's Guide* for more information.

EMM386.EXE is for use on 80386 and 80486 systems only. To use an application that requires expanded memory on an 80286 or 8086 computer, you need to configure your system so that it provides as much physical expanded memory as the application needs. For more information on expanded memory, see "Understanding Memory" earlier in this chapter.

Basic Recommendations

- Install EMM386 only if your system has only extended memory, but you want to use applications that require expanded memory.
- If you install EMM386, allocate it only as much memory as the application needs.

  For example, if you want to run an application that requires 256K of expanded memory, you would allocate 256K of memory to EMM386.

Advantages

- Simulates expanded memory on systems that have only extended memory.
- Can improve the speed of applications that run faster with expanded memory.

Disadvantages

- Works only on 80386 and 80486 systems.
- Uses up some extended memory.

To install EMM386.EXE, add a command line for EMM386.EXE to your CONFIG.SYS file.

The EMM386 command line tells MS-DOS where to find the EMM386.EXE file and how much extended memory to allocate to EMM386. EMM386 will then provide that amount of expanded memory to applications that need it.

The following is a typical EMM386 command line:

```
device = c:\dos\emm386.sys 640
```

It allocates 640K of memory for use as expanded memory, and tells MS-DOS that EMM386.EXE is in the \DOS directory on drive C.

The EMM386.EXE command line must come after the command line that starts HIMEM.SYS, but before any command lines for programs that use expanded memory.

NOTE When you install EMM386, you should disable or remove any other command lines for expanded memory management software.

For details on the EMM386 command line, see the *MS-DOS 5.0 User's Reference.*

Improving the Efficiency of Your Hard Disk

This section explains how to perform procedures that can improve your system's speed without using additional memory. As explained earlier in "Understanding Disk Space," the amount of free disk space can affect your system's performance and storage capacity. In addition, your system will run faster if the information on your disk is structured as efficiently as possible.

The following methods can improve your hard disk's efficiency without taking up additional memory:

- Delete files you don't need.

  If your disk is getting full, you should delete files you don't need or use often. Depending on how you use MS-DOS, you might want to delete certain MS-DOS files to free disk space. For more information, see the following section, "Deleting Unnecessary Files."

- Run the **chkdsk** /f command to recover lost disk space, then delete the files **chkdsk** creates.

  Running **chkdsk** /f can recover lost allocation units that are taking up space on your hard disk. For more information, see "Using the Chkdsk Command," later in this chapter.

- Make sure MS-DOS is searching for files in the most efficient order.

  When you type a command, start a program, and so on, MS-DOS must first find the file you are requesting. This search process can take some time. There are several ways to speed up the sesarch process. For more information, see "Helping MS-DOS Find Files Faster," later in this chapter.

- Reorganize the files on your hard disk.

  The order in which files are stored on your hard disk can dramatically affect the time it takes to find and read those files. Running a disk-compaction utility reorganizes the file structure on your hard disk, so that all the free disk blocks are together. This can greatly improve your system's speed. For more information, see "Reorganizing Your Hard Disk to Improve Speed," later in this chapter.

- Adjust your hard disk controller's interleave.

  A disk's *interleave* affects how many times the disk must revolve in order for an entire disk track to be read. An improperly interleaved hard disk can be slow and inefficient. For more information, see "Adjusting Your Hard Disk's Interleave," later in this chapter.

The following sections explain each method.

Deleting Unnecessary Files

As explained in "Understanding Disk Space" earlier in this chapter, disk space can be a valuable system resource. It allows you to store files and applications when you're not using them. In addition, with programs that perform virtual memory management, such as Microsoft Windows 3.0 or later, free disk space can provide not only storage for files; it can provide additional capacity for running applications and storing data in memory.

If you need more disk space, an easy solution is to delete some unnecessary files. There are three categories of files you might want to delete:

- Application files, documents, or utilities that you no longer need or use.
- Temporary files that were left on your hard disk when an application terminated unexpectedly.
- MS-DOS files that were installed automatically by Install, but which you do not need or plan to use. See the list later in this section for details.

CAUTION Do not delete any MS-DOS files other than the ones listed later in this section.

Basic Recommendations

- It is particularly important to delete unnecessary files before compacting a hard disk.
- In general, keep as much disk space free as possible.

To delete unnecessary files, use the **del** command. The following are some hints for deleting files:

- Delete any temporary files that were left behind by applications.

  Many applications create temporary files while they are running. Some applications store those files in a separate directory in the directory that applications use to store temporary files. (If you have a TEMP directory, it is specified in your AUTOEXEC.BAT file using the set command.)

  You should periodically clean out your TEMP directory. (This is not necessary if your TEMP directory is on a RAM disk.) To avoid deleting a temporary file that is currently in use, you should delete files in your TEMP directory only when you are not running any applications.

- If your system is very short on disk space, you might want to delete some MS-DOS files in order to conserve disk space.

  If you plan to delete MS-DOS files, you should first use the MS-DOS Install program to install MS-DOS on a set of floppy disks. This will make it easy for you to restore individual files later.

  The following list provides more information on the MS-DOS files you can delete.

CAUTION Never delete the files COMMAND.COM, IO.SYS or MSDOS.SYS. If you delete these files, your system will not start.

You can delete the following MS-DOS files if you do not plan to use them:

Filename(s)	Description	When to Delete
EMM386.EXE	Expanded memory emulator.	If your system is not an 80386 or 80486, or if you do not use applications that require expanded memory.
RAMDRIVE.SYS	RAMDrive memory-disk utility	If you do not need a RAM disk, or if your system has only conventional memory.
SMARTDRV.SYS	SMARTDrive disk-caching utility	If your system does not have a hard disk, or if your system has only conventional memory.
NLSFUNC,, GRAFTABL.COM, KEYB.CON, *.CPI, COUNTRY.SYS, DISPLAY.SYS, KEY-BOARD.SYS, PRINTER.SYS	Files that provide international support and code page support	If you are in the U.S. and do not need international (foreign language) support.
MOUSE.COM	Mouse driver	If you do not have a mouse installed on your system.

Filename(s)	Description	When to Delete
EXE2BIN, LINK	Programming tools	If you do not plan to do any programming.

Using the Chkdsk Command

The **chkdsk** command can recover lost file clusters that are taking up space on your hard disk. (A cluster is the smallest piece of your hard disk that can be allocated to a file.) Clusters can get "lost" when an application terminates unexpectedly, leaving temporary files on the hard disk without saving or deleting them properly. Over time, lost clusters can accumulate and take up disk space.

When you run **chkdsk** with the /f option, it finds and recovers any lost allocation units. **Chkdsk /f** converts the lost units to visible files that you can examine and delete.

CAUTION Always exit all applications and memory-resident software before running **chkdsk** with the /F option; never run **chkdsk /f** from within an application. Loss of data might result.

Basic Recommendations

- Run **chkdsk /f** occasionally to make sure there are no lost file clusters on your disk.
- Run **chkdsk /f** before compacting your disk.
- You also might want to run **chkdsk /f** after an application terminates unexpectedly.
- Make sure you exit all applications before running **chkdsk**. If you use **fastopen** or SMARTDrive, disable those commands in your CONFIG.SYS file and restart your computer to ensure that those utilities do not interfere with the disk-compaction process.

▶ **To clean up lost files using chkdsk:**

1. Exit from all applications.
2. Change to the hard disk you want to clean up. For example, to clean up files on your drive D, you would type **d:** at the DOS prompt.
3. Type the command **chkdsk /f**.
   The /f option tells **chkdsk** to fix any lost file clusters it finds.

4. If **chkdsk** finds any lost clusters, it asks if you want to convert the lost clusters to files. If you want to inspect the contents of the lost clusters before deleting them, type *y* for Yes. (If you are sure the lost clusters do not contain information you want, type *n* for No. **Chkdsk** will then simply delete the information; you can then skip the remaining steps in this procedure.)

   If you answer *y*, **chkdsk** converts any lost file clusters to visible files with filenames similar to FILE0001.CHK. It puts the files it finds in the disk's root directory. **Chkdsk** also displays information about the disk it just checked.

5. Use the **type** command to examine the .CHK files. For example, to examine the file FILE0001.CHK, you would type **type file0001.chk**. (For information on the **type** command, see Chapter XX.

   Sometimes, a .CHK file might contain information you want to keep. For example, if a word processing application terminated before you saved your edits, you might find your lost edits in a recovered .CHK file.

6. Delete any .CHK files you don't want.

Helping MS-DOS Find Files Quickly

When you type a command, start a program, and so on, MS-DOS must first find the file you want before it can execute the command or program. If you type the full pathname of the file, MS-DOS can find and execute the file almost immediately. If you type only the filename, MS-DOS searches for the file as follows:

1. First, MS-DOS looks for the file in your current directory.
2. MS-DOS then looks for the file in each of the directories specified by your **path** command.

   The **path** command determines which directories MS-DOS searches Typically, the **path** command is included in your AUTOEXEC.BAT file. Your **path** command also determines the order in which MS-DOS searches for files.

This searching can take time, particularly if your path contains many directories, or if there are many files in those directories. The fewer directories and files MS-DOS has to search through before finding the executable file, the faster the response will be.

For example, if you type **word** at the MS-DOS prompt, MS-DOS looks for a file named WORD.EXE, WORD.COM, or WORD.BAT. It first looks in your current directory. If it doesn't find the file there, MS-DOS then looks in the first directory listed in your path. MS-DOS continues searching through each directory on your path until it either finds the file or until there are no more directories to search.

Basic Recommendations

- If your disk has one or two directories that contain frequently-used executable files, you might want to list those directories first in your **path** command. For example, suppose all your MS-DOS batch (.BAT) files are in the directory C:\BELFRY, and the programs you use most frequently are in the directory C:\PROGRAMS. An efficient **path** command might look like the following:

  ```
  path = c:\belfry;c:\programs;c:\dos;c:\;c:\util
  ```

- When typing the name of a program or batch file, you can speed up MS-DOS' search for that file by including the full pathname of the file. MS-DOS then looks only in that directory.
- Keep the number of files and directories on your hard disk to 150 or less. This can reduce the time MS-DOS spends searching.

Reorganizing Your Hard Disk to Improve Speed

Over time, as applications read from and write to your hard disk, information on your disk can become *fragmented*. Fragmentation occurs when a file, instead of being stored in contiguous sectors of the disk, is broken into fragments that are stored in different locations on the disk. Although fragmentation doesn't affect the validity of the information—your files are still complete when you read them into an application—it takes much longer to read them from the disk. It also takes longer for applications to write files back to the disk.

In addition to file fragmentation, disk speed can be affected by the order in which files are stored. For example, programs and other files that will not change should be stored in the early part of the disk to enhance performance and reduce future fragmentation.

There are two ways to improve the organization of your hard disk:

- Run a disk-compaction utility.

  Disk-compaction utilities reorganize the information on your disk so that all the information in each file is stored as close together as possible. This makes reading from and writing to your hard disk much more efficient.

- Back up, reformat, and restore your hard disk.

  Reformatting your hard disk "wipes it clean." You can then restore your files in the most efficient order. The process of backing up, reformatting, and restoring your hard disk can be very time-consuming.

The following sections explain how to use disk-compaction utilities and how to back up, reformat and restore your hard disk.

3. Run the **chkdsk** command with the **/f** option.

   Never run **chkdsk /f** from within an application. The previous section explains how to run **chkdsk /f**.
4. Ensure that you are not running any programs that use the hard disk.
5. Run the **backup** utility according to the instructions in Chapter ??, "??".
6. After your files are completely backed up, run **restore** to reformat your hard disk. See Chapter ??, "??," for more information.
7. Use the **backup** utility to restore your saved files onto the newly formatted hard disk.

Adjusting Your Hard Disk's Interleave

A hard disk's *interleave* determines how many times the disk must revolve in order for an entire disk track to be read.

The optimum interleave for each hard disk depends on the type of hard disk and controller. The interleave is particularly important if you are using SMARTDrive. An improper interleave can reduce your disk's speed by as much as 200 or 300 percent.

You can purchase a utility that checks and adjusts your hard disk's interleave. (For information on using such a utility, see that utility's documentation.)

Basic Recommendation

- It is a good idea to adjust your hard disk's interleave so that it is optimum for your disk type and controller.

Advantages

- An optimum interleave can drastically improve your system's speed, particularly if you are using SMARTDrive.
- Does not take up memory or disk space.
- Needs to be done only once (although you might want to experiment to find the best interleave for your system).

Disadvantages

- MS-DOS does not include a utility for adjusting the interleave of your hard disk. You can purchase such a utility from your computer store.

1. Delete any unnecessary files from that disk using the steps outlined in "Deleting Unnecessary Files" earlier in this chapter.
2. Exit from all applications and memory-resident software.

   If you use **fastopen** or SMARTDrive, disable those commands in your CONFIG.SYS file and restart your computer to ensure that those utilities do not interfere with the disk-compaction process.
3. Run the **chkdsk** command with the **/f** option.

   Never run **chkdsk /f** from within an application. The previous section explains how to run **chkdsk /f**.
4. Run your disk-compaction utility according to the manufacturer's instructions.

Reducing File Fragmentation by Reformatting Your Disk

If you do not own a disk-compaction utility and you suspect that the files on your hard disk are badly fragmented, you might want to compact your hard disk by reformatting it. Doing so involves backing up the files on your hard disk, reformatting it, and restoring the files to the reformatted disk.

Basic Recommendation

- Reformat your hard disk to reduce file fragmentation only if you suspect the information on the disk is badly fragmented. Unlike running a disk-compaction utility, this process is too time-consuming to do frequently.

Advantages

- Makes it faster to read and write files on the hard disk. This, in turn, speeds up your system's performance.
- Can significantly speed up the time it takes for applications to start up.
- Does not require purchase of an additional utility.

Disadvantages

- Can take several hours to complete.

▶ To reorganize your hard disk by reformatting it:

1. Delete any unnecessary files from that disk using the steps outlined in "Deleting Unnecessary Files" earlier in this chapter.
2. Exit from all applications.

- Some disk-interleaving utilities require that you reformat your hard disk; however, a few utilities are capable of safely resetting the interleave without affecting the data on the hard disk.

Freeing Up Memory

If you are having trouble running applications you need because there is not enough memory available, then your main goal should be not to improve speed, but to free up memory for the applications.

Of course, in order to run an application, your system must contain as much physical memory as that application requires. For example, if an application requires 512K of memory, it is not going to run on a system that has only 256K of memory, no matter how much memory you free up.

However, if your system does contain sufficient memory, but the application still won't run, then you need to free some memory for use by that application.

In most cases, an application fails to run because of insufficient conventional memory. However, some applications require additional expanded or extended memory.

The following sections explain how to make more memory available to applications.

Freeing Up Conventional Memory

All applications require conventional memory in order to run. You can make more conventional memory available to applications by minimizing how much memory MS-DOS, installable device drivers, and other memory-resident programs use. Applications can use only the conventional memory that is available when you start them. If memory-resident software is already using memory, then the application cannot use that memory.

There are several ways to free up conventional memory for use by applications:

- If your system has extended memory, you can tell MS-DOS to run in extended memory (HMA) instead of conventional memory. Because few applications use the HMA, loading MS-DOS into the HMA gets it "out of the way" of other applications.
- You can "streamline" your CONFIG.SYS and AUTOEXEC.BAT files so that they do not start unnecessary memory-resident software.

The following sections explain how to load MS-DOS into the HMA, and how to "streamline" your CONFIG.SYS and AUTOEXEC.BAT files so that they start only programs that you really need.

Running MS-DOS in Extended Memory

Normally, when you start MS-DOS, it loads its routines into your system's conventional memory. This takes up some memory, making that memory unavailable to applications. However, if your system has extended memory, MS-DOS can load its routines into extended memory. MS-DOS uses a part of your system's extended memory called the *high memory area* (HMA), which is the first 64K of extended memory. Because few applications use the HMA, it makes sense to use that memory area for running MS-DOS.

To tell MS-DOS to load its routines into extended memory, include the following command lines in your CONFIG.SYS file:

```
device = himem.sys
dos = high
```

These command lines first load the HIMEM extended memory manager, then tell MS-DOS to load its routines into extended memory. The HIMEM memory manager is required in order to run MS-DOS in extended memory.

Streamlining Your CONFIG.SYS and AUTOEXEC.BAT Files

When you start your computer, the settings in your CONFIG.SYS and AUTOEXEC.BAT files can start device drivers, utilities, and other programs that use memory. You can make more memory available to applications by removing unnecessary commands and utilities from these files.

Basic Recommendation

- Avoid starting unnecessary memory-resident software from your CONFIG.SYS and AUTOEXEC.BAT files.

To effectively streamline your CONFIG.SYS and AUTOEXEC.BAT files, you should know the function of each of the command lines in those files. Because the information in these files can determine how your computer starts and runs, you should not change these files unless you know what effect your changes may have.

CAUTION Use care when changing your CONFIG.SYS and AUTOEXEC.BAT files. If you incorrectly change or disable some values, your system may not function properly. Therefore, it is important to make a bootable floppy disk that contains a backup copy of these files.

▶ To streamline your CONFIG.SYS and AUTOEXEC.BAT files:

1. Make a bootable floppy disk that includes a backup copy of both your CONFIG.SYS file and your AUTOEXEC.BAT file. For more information, see "???."
2. Use a text editor such as Edit to open and edit your CONFIG.SYS and AUTOEXEC.BAT files.
3. Disable any command lines for unnecessary device drivers and utilities.

   When streamlining your CONFIG.SYS and AUTOEXEC.BAT files, it is better to simply disable command lines than to delete them. That way, if you accidentally disable a command line you really need, you can restore it easily. To disable a command line, insert the command **rem** (for "remark") at the beginning of the line. For example, to disable the following CONFIG.SYS command line:

   ```
   device = c:\device\mouse.sys
   ```

   you would use the **rem** command as follows:

   ```
   rem device = c:\device\mouse.sys
   ```
4. Save the file.
5. When you have finished editing both files, restart your computer by pressing CTRL+ALT+DEL.

   If your system does not start properly, insert the bootable floppy disk you created in step 1 in drive A, and start your computer again. If you know which command line(s) are causing the problem, edit the appropriate file (CONFIG.SYS or AUTOEXEC.BAT) on your hard disk, and restart your computer. Or, to start over, copy the backup version of the file from the floppy disk to your hard disk.

Your CONFIG.SYS File

As explained in Chapter 12, "Customizing Your System," your CONFIG.SYS file is a text file that defines device drivers and specifies your MS-DOS configuration. For example, a typical CONFIG.SYS file might specify the location of the MS-DOS file COMMAND.COM, define an extended memory manager, and specify how many files an application can have open at once. MS-DOS runs the command lines in your CONFIG.SYS file before those in your AUTOEXEC.BAT file.

Basic Recommendations

- Disable command lines for any unnecessary device drivers using the method explained in the preceding section.

Your CONFIG.SYS file should define only device drivers that you really need. (Device drivers are defined using the **device** command.)

If your CONFIG.SYS file contains a **device** command for SMARTDrive, RAMDrive, or **fastopen**, you might want to disable that command line to conserve memory. (SMARTDrive, in particular, can use up a lot of conventional memory.)

- If your system has expanded memory, your CONFIG.SYS file should contain a **device** command for the expanded memory manager that came with your memory board.
- If your system has extended memory, your CONFIG.SYS file should contain a **device** command for the HIMEM.SYS expanded memory manager.
- If your system has extended memory, your CONFIG.SYS file should include the following command:

  ```
  DOS = high
  ```

  This command saves conventional memory by running MS-DOS in extended memory. (The command must come after the **device** command that installs the HIMEM expanded memory manager.)
- If your CONFIG.SYS file contains a **buffers** command, try reducing the number of buffers. (Note that some applications might not run properly if you reduce the number too far.)
- Add a **stacks** command to limit the number and size of interrupt stacks that MS-DOS uses. You can conserve memory by setting **stacks** to **0,0**, as follows:

  ```
  stacks = 0,0
  ```
- If your CONFIG.SYS file includes the **lastdrive** command, you can save some memory by setting **lastdrive** to a letter such as *j* or *k*, rather than to *z*. (If you use a network, this may limit the number of network disk drives you can use simultaneously.)
- If your CONFIG.SYS file contains an **fcbs** command, try setting fcbs to a low value. If you do not use a network, you can set **fcbs** to 1.

Your AUTOEXEC.BAT File

As explained in Chapter 12, "Customizing Your System," your AUTOEXEC.BAT file is a special MS-DOS batch file. Like any batch file, it lists MS-DOS commands; when you run the batch file, MS-DOS executes the commands in the file. Unlike other batch files, MS-DOS automatically executes the commands in your AUTOEXEC.BAT file immediately after it completes the commands in your CONFIG.SYS file. (Your AUTOEXEC.BAT file is located in the root directory of your first hard disk, which is usually drive C.)

Typically, an AUTOEXEC.BAT file starts memory-resident utilities such as a network and sets up environment variables that are used later by applications. In addition, your AUTOEXEC.BAT file might set your DOS prompt or start pop-up programs such as Borland's SideKick.

Basic Recommendations

- Disable command lines that start pop-up programs or other memory-resident software that you do not need.
- If you use a mouse only with Windows and Windows applications, you might want to disable any command lines that start and enable mouse-driver software such as MOUSE.COM. (Windows has a built-in mouse driver.)

Freeing Up Extended Memory

A few applications require additional extended memory in order to run. If you are having trouble running such an application, check the following:

- Make sure your system contains as much physical extended memory as the application needs.
- Make sure your CONFIG.SYS file contains a command line for the HIMEM.SYS extended memory manager. Some applications need HIMEM.SYS in order to use extended memory.
- If your CONFIG.SYS file contains a command line for SMARTDrive or RAMDrive, ensure that those programs are not using up all of your extended memory. You can reduce the amount of extended memory you are giving to SMARTDrive or RAMDrive; or, you can simply disable those command lines by using the **rem** command.
- Make sure your CONFIG.SYS and AUTOEXEC.BAT files do not start unnecessary programs that use extended memory. See "Streamlining Your CONFIG.SYS File and AUTOEXEC.BAT File," earlier in this chapter, for details.
- If the application will not start, and displays a message such as "High Memory Area (HMA) already in use," then you need to free up the high memory area for that application.

  Few applications require use of the high memory area. If your application requires the high memory area and your CONFIG.SYS file contains the command **dos = high**, you might want to disable that command. Doing so will cause MS-DOS to run in conventional memory rather than in the high memory area of extended memory.

Freeing Up Expanded Memory

Some applications require additional expanded memory in order to run. If you are having trouble running such an application, check the following:

- Make sure your system contains as much physical expanded memory as the application needs.

  If you have an 80386 system with extended memory, you can use EMM386.EXE to provide expanded memory for applications. For details, see "Installing the EMM386 Expanded Memory Emulator" earlier in this chapter.
- Make sure your CONFIG.SYS file contains a command line for the expanded memory manager that came with your memory board. This command line is required in order for applications to use expanded memory.
- If your CONFIG.SYS file contains a command line for SMARTDrive or RAMDrive, ensure that those programs are not using up all of your expanded memory. You can reduce the amount of expanded memory you are giving to SMARTDrive or RAMDrive; or, you can simply disable those command lines by using the **rem** command.
- Make sure your CONFIG.SYS and AUTOEXEC.BAT files do not start unnecessary programs that use expanded memory. See "Streamlining Your CONFIG.SYS File and AUTOEXEC.BAT File," earlier in this chapter, for details.
- If you are using EMM386.EXE, you can make more expanded memory available to applications by giving EMM386 more extended memory. EMM386 can then use the additional extended memory to provide more expanded memory for applications.

Chapter 14

Customizing for International Use

14

This chapter describes the procedures you use to change the national language that MS-DOS uses. When MS-DOS was installed, the proper commands may already have been added to your CONFIG.SYS and AUTOEXEC.BAT files to convert your system to a language other than United States English.

MS-DOS 5.0 can use the language conventions, keyboards, and character set for the following countries/languages:

Belgium	Canadian-French
Denmark	France
Germany	Italy
Latin America	Netherlands
Norway	Portugal
Swiss-French	Swiss-German
Spain	Finland
Sweden	United Kingdom
United States	

Understanding Languages

MS-DOS assumes that you want to use United States English unless you tell it otherwise. There are three changes you can make to change the language that MS-DOS uses:

- You can change MS-DOS to use a country's language conventions for displaying dates, times, currency, character sort order, and valid filename characters.
- You can change the characters and arrangement of your keyboard to fit the standard keyboard of a country.

- You can change the character set that MS-DOS uses so that you can type, display and print characters from different languages.

To change the conventions of MS-DOS, you put a **country** command in your CONFIG.SYS file. When you type the **country** command, you specify a three-digit *country code* that tells MS-DOS which conventions you want to use. MS-DOS retrieves the appropriate conventions from the COUNTRY.SYS information file.

To change the kind of keyboard that MS-DOS assumes you are using, you include a **keyb** command in your CONFIG.SYS or AUTOEXEC.BAT file, or enter the command at the command prompt. When you type the **keyb** command you specify a two-letter *keyboard code* that tells MS-DOS which type of keyboard you want to use. When you run the **keyb** command, MS-DOS retrieves a table from the KEYBOARD.SYS information file that tells it which character to assign to each key. For example, if you choose the US keyboard, MS-DOS assigns the exclamation mark character(!) to the SHIFT+2 key. If you choose the Italian keyboard, MS-DOS assigns the quotation mark character (") to SHIFT+2.

At any time, MS-DOS has a set of 256 different characters it can assign to a key. The set of 256 characters is called a *code page*. Unless you tell it otherwise, MS-DOS uses the code pages that come with your computer. These code pages are called *hardware code pages*. Your computer has a hardware code page for the console (the keyboard and display) and your printer may also have a hardware code page.

If all of the characters you need are in your hardware code pages, you can change languages by using the **country** and **keyb** commands. If you want to use a language with characters that are not included in your hardware code page, you must use a *prepared code page*. Prepared code pages are alternative sets of 256 characters that are stored in code page information (.CPI) files.

Not all devices can use prepared code pages. For example, monochrome and CGA screens, as well as many printers, can use only their own built-in hardware code page.

MS-DOS 5.0 comes with five prepared code pages. All code pages, including hardware code pages, contain the same standard ASCII, scientific, and graphics characters. Code pages differ in the special language characters they contain. The most common code page is named "347". This is the standard code page for US English. It also contains most of the characters used by most European languages.

If your computer's hardware code page is 347, you will be able to use most European languages, although you may not be able to type certain characters unless you change to a prepared code page.

If no prepared code pages have yet been added to your system, you can find out which hardware code page you have by entering the following **chcp** command:

```
chcp
```

To use a prepared code page instead of your hardware code page you must perform three tasks:

- You must include a **device** command in your CONFIG.SYS file to install a new device driver for the device with which you want to use the code page. For example, to use a prepared code page with your display and keyboard, you must install the DISPLAY.SYS device driver.
- You must load the code page you want to use and an additional support program into your computer's Random Access Memory (RAM). You load the code page with a special form of the **mode** command. You load the support program with an **nlsfunc** command.
- You must make the prepared code page active by selecting it with the **chcp** command or the **mode** command.

With the proper commands, you can use more than one prepared code page. While you are working, you can switch between your hardware code page and any prepared code pages you have loaded.

When you use commands to set up your system for a national language, MS-DOS checks to make sure the screen, printer, and keyboard codes you request work together. If the codes you use are incompatible, MS-DOS displays an error message.

The following table shows the country codes, keyboard codes and code pages that you can use together:

Country or Language	Country Code	Keyboard Code	Valid Code
Belgium	032	be	437 and 850
Canadian-French	002	cf	863 and 850
Denmark	045	df	865 and 850
France	033	fr	437 and 850
Germany	049	gr	437 and 850
Italy	039	it	437 and 850
Latin America	003	la	437 and 850
Netherlands	031	nl	437 and 850
Norway	047	no	865 and 850

Country or Language	Country Code	Keyboard Code	Valid Code
Portugal	351	po	860 and 850
Swiss-French	041	sf	437 and 850
Swiss-German	041	sg	437 and 850
Spain	034	sp	437 and 850
Finland	358	su	437 and 850
Sweden	046	sv	437 and 850
United Kingdom	044	uk	437 and 850
United States	001	us	437 and 850

Changing Conventions and Keyboards

To change the language MS-DOS uses you must always change the language conventions and the arrangement of the keyboard. To change MS-DOS's language conventions, you include a **country** command in your CONFIG.SYS file. To change the arrangement of the keyboard, you enter a **keyb** command. You can use the **country** and **keyb** commands whether or not you load any code pages.

Changing Date and Time Formats With the Country Command

In Brief

To set MS-DOS's language conventions, include a **country** command in your CONFIG.SYS file. For example, if your COUNTRY.SYS file is in the C:\DOS directory, the following command sets MS-DOS to the language conventions of Finland (358):

```
country=358,,c:\dos\country.sys
```

Notice that there are two commas between the country and the location of COUNTRY.SYS.

Unless you tell it otherwise, MS-DOS assumes that you want to use United States language conventions for the following items:

- The way the date and time are displayed.
- The symbol that is used for currency (for example, The German mark, the Italian lira, the Danish krone)

- The sort order MS-DOS should use when alphabetizing files or listings.
- The list of characters that can be used in file and directory names.

To change the language conventions that MS-DOS uses, you include a **country** command in your CONFIG.SYS file. When you type the **country** command you specify which country's conventions you want MS-DOS to use. The conventions for each country are stored in the COUNTRY.SYS file. When you change conventions, MS-DOS uses the information in the COUNTRY.SYS file rather than the US conventions.

When you type a **country** command, you must include a three digit *country code* to specify which conventions you want to use. For example, the country code for Spain is 034. For a complete list of country codes, see the table in the previous section.

MS-DOS assumes that your COUNTRY.SYS file is located in the root directory of your startup disk. If COUNTRY.SYS is not in the root directory of your startup disk, you need to type its pathname in the **country** command.

For example, suppose you want to use the language conventions of Italy (country code 039). If your COUNTRY.SYS file is in the root directory of your startup disk, you need only type the following simple command:

```
country=039
```

If your COUNTRY.SYS file is in the C:\DOS directory, you type the following command:

```
country=039,,c:\dos\country.sys
```

In this form of the command there must always be two commas (,) between the country code and the location of COUNTRY.SYS. In general, this is the most complicated command you have to type to change language conventions.

If you want to tell the **country** command that you will be using a different code page, you can include the code page number in the command. For example, to tell the **country** command that you will be using code page 850, you type the following **country** command:

```
country=039,850,c:\dos\country.sys
```

If you specify a code page in the **country** command, you can still switch to a different code page later.

Changing Keyboards with the Keyb Command

In Brief

To change the characters on your keyboard and their arrangement, use the **keyb** command. For example, if the KEYBOARD.SYS file is in the C:\DOS directory, the following command changes to the characters and arrangement of a Swedish keyboard (sv):

```
keyb sv,,c:\dos\keyboard.sys
```

Notice that there are two commas (,) between the country and the location of KEYBOARD.SYS.

When you change from US English to another language, you add characters to your keyboard and rearrange its keys. For example, to change MS-DOS from US English to Latin American Spanish you add four characters to your keyboard (one letter and three symbols). To make room for the extra characters, many of the characters on the keyboard must be rearranged.

To rearrange the keyboard and add new characters, you use the **keyb** command. The **keyb** command works with PC1, PC XT, PC AT, and IBM PC-convertible keyboards.

There are three ways to run the **keyb** command:

- You can include a **keyb** command in your AUTOEXEC.BAT file so that it runs automatically when you start your computer.
- You can enter the **keyb** command at the command prompt.
- You can use an **install** command in your CONFIG.SYS file to run the command when MS-DOS reads the CONFIG.SYS file.

Regardless of how you run the initial **keyb** command, you can always change keyboards by entering a **keyb** command at the system prompt.

If you change keyboards, you can switch back to a US keyboard by pressing CTRL+ALT+F1. To return to the keyboard you were using, press CTRL+ALT+F2.

Entering Keyb Commands

When you enter the **keyb** command at the system prompt or type it in a batch file, you include a two-letter keyboard code to specify which keyboard you want to use. MS-DOS looks in the KEYBOARD.SYS file to find out how to rearrange your keyboard for the country you chose.

MS-DOS assumes that your KEYBOARD.SYS file is located in the root directory of your startup disk. If KEYBOARD.SYS is not in the root directory of your startup disk, you need to type its pathname in the **keyb** command.

For example, suppose you want to use an Italian keyboard (keyboard code IT). If your KEYBOARD.SYS file is in the root directory of your startup disk, you need only type the following simple command

```
keyb it
```

If your KEYBOARD.SYS file is in the C:\DOS directory, you type the following command:

```
keyb it,,c:\dos\country.sys
```

In this form of the command there must always be two commas (,) between the country code and the location of COUNTRY.SYS. In general, this is the most complicated command you will have to type to change the keyboard.

If you want to change the default code page for the keyboard, you can include the code page number in the command. For example, to tell the **keyb** command to assume that you will use code page 850, you would type the following **country** command:

```
keyb it,850,c:\dos\country.sys
```

If you specify a code page in the **keyb** command, you can still switch to a different code page later.

You can achieve exactly the same result by entering the **keyb** command without a code page and then changing the current code page to 850. For information about changing code pages, see xx later in this chapter.

Running Keyb from Your CONFIG.SYS File

Rather than having a **keyb** command in your AUTOEXEC.BAT file or entering a keyb at the command prompt, you can include an **install** command in your CONFIG.SYS file that runs the **keyb** command. To run the **keyb** command from the CONFIG.SYS file you use the same parameters as from the command line. For example, to change to an Italian keyboard from the CONFIG.SYS file, you include the following command:

```
install=keyb.com it,,c:\dos\country.sys
```

This command has the same result as if you had entered the **keyb** command at the system prompt.

Using Different Keyboards

The following diagrams show the characters and keyboard arrangements for most of the Enhanced PC keyboards MS-DOS can use.

This artwork is not available
for this release.

ciu_1

Most keys have at least two characters assigned to them. Some keys have three or four characters assigned to them. If there are three or four characters assigned to a key, the additional characters are shown on the key to the right of the usual characters. o type the different characters, you press the key in combination with one or more other keys:

- To produce the character in the lower-left of the key, you press the key.

- To produce the character in the upper-left of the key, you press the key plus the SHIFT key.
- To produce the character in the lower-right of the key, you press the key plus the ALT-GR key (just to the right of the SPACE key)
- To produce the character in the upper-right of the key, you press the key plus the ALT key, plus the SHIFT key.

NOTE For IBM PC and IBM AT-compatible keyboards, you can produce the character in the lower right portion of the key by pressing the key plus CTRL+ALT. (On Danish, Finnish, Norwegian, and Swedish keyboards, your press the key plus ALT. On French-Canadian keyboards, you press the key plus the ALT and SHIFT keys.)

To produce accented (and umlauted) characters, you press *dead keys*. Dead keys are keys that do not display a character when used alone, but when followed by a letter, display that letter with an accent. For example, you can produce umlauted characters on the Swedish keyboard by pressing the key to the left of the BACKSPACE key and then pressing the key for the character you want to umlaut.

Using Code Pages

Unless you tell it otherwise, MS-DOS uses the character set contained in the hardware code page that is built in to your computer. If you need only the characters that are in the hardware code page of your console (keyboard and display) and your printer, you do not need to learn about or use code pages. However, if you use a language with characters that are not included in your hardware code pages, you need to follow the directions in this section to understand how to install and use prepared code pages.

NOTE Monochrome and CGA displays and many printers cannot use prepared code pages.

MS-DOS has five *prepared code pages* that you can use in addition to or instead of the hardware code pages that are built into your devices. Each prepared code page has the same set of standard ASCII characters, graphics characters, and scientific characters. However, each code page has a different set of national language characters. For example, the Portuguese code page has the same ASCII, graphics, and scientific characters as the other code pages but also includes the characters particular to Portuguese.

These are the five MS-DOS prepared code pages:

- 850 — Multilingual code page. This code page has all the characters for all the languages that MS-DOS can use.
- 437 — United States code page. This code page contains the characters for English and most other European languages.
- 860 — Portuguese code page. This code page contains the characters for English and Portuguese.
- 863 — Canadian-French code page. This code page contains the characters for English and French-Canadian.
- 865 — Nordic code page. This code page includes all characters for English, Norwegian, and Danish.

Tables of the characters contained in each prepared code page are given in xx.

The prepared code pages are stored in code-page information (.CPI) files. Before you use a code page, you load it into RAM. There may be more than one code page in RAM, but only one code page is *active*. If you do not install a prepared code page, your hardware code page is active. If you install one or more prepared code pages, you can switch between the hardware code pages and any of the prepared code pages you have installed.

In general, MS-DOS uses the same code page for the screen and keyboard. However it is possible to assign different code pages to your keyboard and your screen.

Preparing Your Console for Code Pages

In Brief

To prepare your keyboard and screen to use one or more prepared code pages, include a **device=display.sys** command in your CONFIG.SYS file. For example, the following command installs DISPLAY.SYS for an EGA or VGA screen with a 437 hardware code page:

```
device=display.sys con:(ega,437,1)
```

The command reserves space for one prepared code page which will be loaded with a separate command.

Every console has a hardware code page that determines which characters can be typed and displayed. If you only need the code page that your console already uses, then there is no need to install any additional code pages for it. If you want to use characters that are not contained in your hardware code page, you must include a **device** command in your CONFIG.SYS file to reserve space for one or more extra code pages.

MS-DOS has an installable device driver called DISPLAY.SYS that allows you to use prepared code pages with an EGA, VGA or LCD display. Monochrome and CGA screens can use only their own hardware code page (usually 437). If you have an EGA, or VGA display, you can use up to two prepared code pages. If you have an LCD display, you can use only one prepared code page. For more information about installable device drivers, see xx.

If DISPLAY.SYS is in the C:\DOS directory, to install it you include the following **device** command in your CONFIG.SYS file:

```
c:\dos\display.sys con=
```

With this command, you specify the following parameters:

- The kind of display adapter you have. You can choose from EGA and LCD. You can use EGA for both EGA and VGA display adapters. If you omit this parameter, MS-DOS checks your hardware to find out what kind of display adapter you have.
- The hardware code page your console uses. You must specify the name of the hardware code page if you want to be able to make it active again after you have switched to a different code page. The most common value for this parameter is 437, the US code page. To find out which hardware code page your screen has, enter a **chcp** command with no parameters before you have added any code pages to your system.
- The number of prepared code pages you want to use. For VGA or EGA, this number can be 1 or 2. For LCD it can be 1. If you don't specify a number, MS-DOS assumes 0.
- The number of sub-fonts that are supported for each code page. If you leave this number out, MS-DOS assumes 0.

For example, if the DISPLAY.SYS file is in the C:\DOS directory, and your VGA display has a 437 hardware code page, to use one extra code page, you include the following command in your CONFIG.SYS file:

```
device=c:\dos\display.sys con=(ega,437,1)
```

All the parameters are grouped together inside parentheses and separated by commas. EGA is the type of display adapter you have. Notice that you type EGA even though you have a VGA adapter.

437 is the hardware code page of your display. Because you specified a hardware code page, you will later be able to return to the hardware code page after switching to a prepared code page.

The 1 is the number of prepared code pages that you want to use. You don't specify which prepared code page you want because this command does not load the code page. It simply reserves space for the code page.

If you want to use a prepared code page and you don't need to return to the hardware code page, you can omit the hardware code page number from your **device** command as in the following example:

```
device=c:\dos\display.sys con=(ega,,1)
```

Preparing Your Printer for Code Pages

In Brief

If you have a supported printer, you can prepare it to use one or more prepared code pages by including a **device=printer.sys** command in your CONFIG.SYS file. For example, the following command installs PRINTER.SYS for an IBM 5202 Quietwriter printer with a 437 hardware code page attached to the LPT1 port:

```
device=printer.sys lpt1:(5202,437,1)
```

The command reserves space for one prepared code page, which will be loaded with a separate command.

MS-DOS has an installable device driver called PRINTER.SYS that allows you to use prepared code pages with certain types of printers. If you have any of the following printers attached to LPT1 (PRN), LPT2, or LPT3, you can use an MS-DOS prepared code page to change its character set:

- The 4201 code page works with the IBM Proprinter Model 4201 (including the XL) or compatibles.
- The 4208 code page works with the IBM Proprinter Model 4207 or 4208 and the IBM Proprinter X24 or XL24 or compatibles.
- The 5202 code page works with the IBM Quietwriter III Printer Model 5202 or compatible.

Many programs have their own installable device drivers that supersede MS-DOS's driver. See your printer documentation for information about changing its code page. For more information about installable device drivers, see xx.

If DISPLAY.SYS is in the C:\DOS directory and your printer is attached to LPT1, you include the following **device** command in your CONFIG.SYS file to install the printer driver:

```
c:\dos\printer.sys lpt1=
```

When you type the command, you specify the following parameters

- The kind of printer you have. You can choose from 4201, 4208,or 5202 for the printers that MS-DOS supports.
- The hardware code page your printer uses. You must specify the name of the hardware code page if you want to be able to make it active again after you have switched to a different code page. The most common value for this parameter is 437, the US code page. To find out which hardware code page your printer has, see your printer's documentation.
- The number of prepared code pages you want to use. The maximum number of prepared code pages depends on which printer you have.

For example, if the PRINTER.SYS file is in the C:\DOS directory, your 4102 printer has the US hardware code page (437), and you want to use one extra code page, you include the following command in your CONFIG.SYS file:

```
device=c:\dos\printer.sys con=(4102,437,1)
```

All of the parameters are grouped together inside parentheses and separated by commas. 4102 is the type of printer you have (IBM Proprinter Model 4201 or compatible). 437 is the hardware code page of your printer. Because you specified a hardware code page, you will later be able to return to the hardware code page after switching to a prepared code page.

The 1 is the number of prepared code pages that you want to use. You don't specify which prepared code page you want because this command does not load the code page, it simply reserves space for the code page.

If you don't know which hardware code page your printer has, or if you want to use a prepared code page and you don't need to return to the hardware code page, you can omit the hardware code page from your **device** command, as in the following example:

```
device=c:\dos\printer.sys con=(4102,,1)
```

Preparing MS-DOS for Code Pages

In Brief

Before MS-DOS can recognize and change between prepared code pages, you must load the Nlsfunc program into RAM with the following command:

```
nlsfunc
```

The **nlsfunc** (National Language Support function) command loads a program into RAM. The Nlsfunc program helps MS-DOS load and switch between prepared code pages. Nlsfunc also allows MS-DOS to get information from your COUNTRY.SYS file.

Before you use any prepared code pages, you must load the Nlsfunc program into RAM with the following command:

```
nlsfunc
```

The **nlsfunc** command should precede any commands in your AUTOEXEC.BAT file that load or switch code pages.

You can also load the Nlsfunc program from your CONFIG.SYS file. For example, if the nlsfunc command is in the C:\DOS directory, you can load it with the following install command:

```
install=c:\dos\nlsfunc.exe
```

Loading Code Pages

In Brief

To load a code page into RAM, use the **mode cp prep** command with the word CON or the name of the port your printer is attached to. For example, the following command loads code page 850 from the file C:\DOS\EGA.CPI for your EGA or VGA display:

```
mode con cp prep=((850)c:\dos\ega.cpi)
```

This command loads code page 850 from the file C:\DOS\4201.CPI for an IBM 4201 Proprinter attached to LPT1:

```
mode lpt1 cp prep=((850)c:\dos\4201.cpi)
```

The **device** commands you include in your CONFIG.SYS file install the device drivers that let you use prepared code pages, but they do not load the code pages. To load prepared code pages, you use a **mode code page prepare** command (code

page prepare can be abbreviated **cp prep**). The **mode** command extracts the code page you want from the .CPI file where it is stored and loads it into RAM.

Once the code page is in RAM, you can make it the active code page, display information about it, or reload it using other forms of the **mode** command.

When you enter a **mode cp prep** command you specify the following parameters:

- The device for which you want to load the code page. CON loads the code page for the screen and keyboard. PRN, LPT1, LPT2, or LPT3 load the code page for a printer port.
- The prepared code page or code pages that you want to load. You can load as many code pages as you reserved space for in the **device** command you previously included in your CONFIG.SYS file.
- The file in which the code page is stored. All code page files have a .CPI extension. The EGA/VGA code pages are stored in EGA.CPI, the LCD code pages are stored in LCD.CPI and the printer code pages are stored in 4102.CPI, 4208.CPI and 5202.CPI.

For example, the following command extracts code page 850 from C:\DOS\EGA.CPI and loads it into RAM:

```
mode con cp prep=((850)c:\dos\ega.cpi)
```

The name of the device always precedes CP PREP. All of the other parameters are grouped together in parentheses. In addition, the code pages you want to load are grouped together in a second set of parentheses within the first set.

NOTE Before you can use the **mode cp prep** command, you must use the **nlsfunc** command. See the previous section for more information.

If you reserved space for more than one code page when you included a device command in your CONFIG.SYS file, you can load more than one code page. For example, the following command loads code pages 850 and 865:

```
mode con cp prep=((850 865)c:\dos\ega.cpi)
```

Notice that the two code pages are separated by a space.

To install code pages for the screen and printer, you use two separate commands. For example, the following two commands load code page 865 for the screen and 5202 printer attached to lpt1:

```
mode con cp prep=((865)c:\dos\ega.cpi)
mode lptl cp prep=((865)c:\dos\5202.cpi)
```

Changing Code Pages

In Brief

To change the code page for all of your devices, use the **chcp** command as in the following example:

```
chcp 850
```

To change the code page for a single device, use the **mode cp select** command. For example, the following command makes code page 850 active for the Screen and keyboard:

```
mode con cp select=850
```

After you have installed the proper device driver, loaded the Nlsfunc program, and loaded the code page you want to use for a device into RAM, you can use the code page by making it active. To make a code page active for all devices you use the **chcp** command. To make it active for a single device, you the **mode cp select** command.

You cannot change to a prepared code page if a device uses only a hardware code page or if the code page has not been loaded for the device. In addition, you cannot change the code page used with the keyboard if it does not match the keyboard's country. For example, the Danish keyboard (DF) can only be used with code pages 850 and 865. Thus, you cannot change the code page of a Danish keyboard to 437.

Using the Chcp Command to Change Code Pages

Using the **chcp** command, you can make a code page active for every device that can use it. For example, the following command makes code page 850 active for every device:

```
chcp 850
```

If the code page has not been loaded for one or more of your devices MS-DOS will display the following message:

```
Code page 850 not prepared for all devices
```

MS-DOS makes code page active for the devices for which you have loaded the code page. If a device cannot use the code page, it will retain its original code page. For example, if you try to make a code page active that has not been loaded for your printer, MS-DOS will make the code page active for you console but not your printer.

Using the Mode Command to Change Code Pages

To change the code page for a single device, you use a **mode cp select** command. Along with the command you specify the name of the device you want to change and the code page you want to change to.

For example, to change to code page 850 for the printer attached to LPT1, you use the following command:

```
mode lpt1 cp select=850
```

Before this command can be successful, you must have loaded code page 850 for the printer. If the code page is not loaded, MS-DOS will display a "Code page not prepared" message.

Displaying Code Page Information

In Brief

To display code page information for your console and printer, enter the following **mode** command:

```
mode
```

To display code page information for your console only, enter the following **keyb** command:

```
keyb
```

To display the active code page, enter the following **chcp** command:

```
chcp
```

When you use the **mode**, **keyb**, and **chcp** commands with no parameters, thev provide information about the code pages you are using.

You get the most information from the **mode** command. The mode command lists the active code page, hardware code page and any prepared code pages for both the console and LPT1. For example, suppose you have code pages loaded for LPT1 and the console. If you enter the **mode** command with no parameters, you see a display like the following:

```
Status for device LPT1:
-----------------------
LPT1: not rerouted
RETRY=NONE
No code page has been selected
Hardware code pages:
 code page 437
```

```
Prepared code pages:
 code page 850


Status for device CON:
----------------------
COLUMNS=80
LINES=25

Active code page for device CON is 850
Hardware code pages:
 code page 437
Prepared code pages:
 code page 850
```

In this case, both LPT1 and the console have a hardware code page of 437. Both devices also have a code page 850 loaded for them, but only the console has 850 as the active code page.

To display information for the console or any LPT port separately,, use a **mode cp** command. For example, the following command displays code page information for the lpt2 port:

```
mode lpt2 cp
```

If you enter the **keyb** command without parameters, MS-DOS displays a message that tells you which code pages are in use with your keyboard and screen. For example, if you are using a German keyboard and code page 850 for your keyboard and screen, and you enter a **keyb** command with no parameters, MS-DOS displays this message:

```
Current keyboard code: GR code page:850
Current CON code page: 850
```

Sample Language Changes

For many language changes, you don't need to change code pages. If you don't have to change code pages, you need only two commands to change languages. If you need to change code pages, the number of commands you must run depends on how many code pages you want to use and whether you want to use them with the console only or with the console and your printer.

If you commonly change languages, it is most convenient to put all of the commands you need in your AUTOEXEC.BAT file. Then, when you start or reset your computer, MS-DOS sets your keyboard, screen, and printer for the language you want. If you set up your languages in your CONFIG.SYS and AUTO-

EXEC.BAT files, you can quickly switch between keyboards and code pages when you need them.

Changing Languages Without Changing Code Pages

For many language changes, you need only run a **country** command and a **keyb** command. For example, if your MS-DOS files are in the C:\DOS directory, you run the following two commands to change MS-DOS to Italian language conventions and an Italian keyboard:

```
country=039,,c:\dos\country.sys
keyb it,,c:\dos\keyboard.sys
```

The country command must be located in your CONFIG.SYS file. The **keyb** command can be included in your AUTOEXEC.BAT or CONFIG.SYS file, or you can enter it at the command prompt. If you include the previous **keyb** command in your CONFIG.SYS file, it has the following form:

```
install=keyb.com it,,c:\dos\keyboard.sys
```

Using One Prepared Code Page

If the language you want to use requires a prepared code page, you have to add at least two additional commands to your CONFIG.SYS file, and two or more to your AUTOEXEC.BAT file.

For example, suppose your hardware code page is 437, but you want to be able to use code page 850 with an Belgian keyboard and an EGA display. If the MS-DOS files are in C:\DOS, you can use the following commands in your CONFIG.SYS file to change to Belgian language conventions (032) and install the display driver that allows code page switching:

```
country=032,,c:\dos\country.sys
device=c:\dos\display.sys con=(ega,437,1)
```

The **country** command sets MS-DOS to use the Belgian conventions for date, time, currency, character sort order and valid filename characters. The **device** command installs DISPLAY.SYS, tells it that you have an EGA screen with a 437 hardware code page, and reserves space for a prepared code page.

In your AUTOEXEC.BAT file, you include the following commands to load and select code page 850:

```
nlsfunc
mode con cp prep=((850)c:\dos\ega.cpi)
keyb be,,c:\dos\keyboard.sys
chcp 850
```

The **nlsfunc** command loads the Nlsfunc program. The **mode** command loads code page 850 into RAM from the EGA.CPI file. The **keyb** command changes the arrangement of the keyboard to match a Belgian keyboard. The **chcp** command makes code page 850 the active code page.

The following **nlsfunc** and **keyb** commands could have been included in the CONFIG.SYS file rather than in the AUTOEXEC.BAT file:

```
install=c:\dos\nlsfunc.exe
install=c:\dos\keyb.com be,,c:\dos\keyboard.sys
```

While you are working, you can use a US keyboard arrangement temporarily by pressing CTRL+ALT+F1. To return to the Belgian keyboard arrangement press CTRL+ALT+F2.

To return to code page 437, enter this **chcp** command:

```
chcp 437
```

MS-DOS sets the code page for the keyboard and display to 437.

Using Two Prepared Code Pages with Your Console

If you want to use two code pages with your console, you need to load and select more than one code page.

For example, suppose you want to use code pages 865 and 850 with your VGA screen and Danish keyboard. If the hardware code page is 437, and the MS-DOS files are in C:\DOS, you add the following two commands to your CONFIG.SYS file to change to Danish language conventions and install the display driver:

```
country=045,,c:\dos\country.sys device=c:\dos\display.sys
con=(ega,437,2)
```

The **country** command changes to Danish language conventions and the **device** command tells MS-DOS to reserve space for two prepared code pages. Notice that the parameter EGA works also for VGA screens.

To load both code pages and make 865 the active code page, you include the following commands in your AUTOEXEC.BAT file:

```
nlsfunc
mode con code page prepare=((865,850)c:\dos\ega.cpi)
keyb dk,,c:\dos\keyboard.sys
chcp 865
```

The **mode** command loads both prepared code pages into RAM from the EGA.CPI file. The **chcp** command makes code page 865 active before the AUTO-

EXEC.BAT file ends. While you are working, you can use code page 850 by entering the following **chcp** command:

```
chcp 850
```

To change to code page 437, you first have to change the keyboard to a type that can function with code page 437 (the Danish keyboard functions only with pages 865 or 850).

Using Prepared Code Pages with Your Printer

To add the Danish code pages described in the previous section to an IBM 4208 printer, you add another **device** command to your CONFIG.SYS file and another **mode** command to your AUTOEXEC.BAT file.

With the extra **device** command, your CONFIG.SYS file includes these commands:

```
country=045,,c:\dos\country.sys
 device=c:\dos\display.sys con=(ega,437,2)
device=c:\dos\printer.sys lpt1=(4208,437,2)
```

The second **device** command installs PRINTER.SYS and tells it that you have a 4208 printer with a 437 hardware code page. Like the first **device** command, the second command reserves space for 2 prepared code pages.

With the extra mode command, your AUTOEXEC.BAT file includes these commands:

```
nlsfunc
mode con cp prep=((865,850)c:\dos\ega.cpi)
mode con cp prep=((865,850)c:\dos\4208.cpi)
keyb dk,,c:\dos\keyboard.sys
chcp 865
```

The second **mode** command loads code pages 865 and 850 for the 4208 printer from the 4208.CPI file. The **chcp** command makes code page 865 active for both devices.