

UNDERSTANDING DOS

Michael E. Valdez, Ph. D.

1993

A THOUGHT

Most people can't stop their VCR's from flashing 12:00 after a power failure, yet PC users are expected to master DOS.

Contents

- [Chapter 1.- Introduction](#)
 - 1.1 THE COMPUTER
 - 1.2 SAFETY
 - 1.3 A WORD ABOUT INSTALLATION
 - 1.4 WHAT A COMPUTER CAN DO
 - 1.5 TALKING TO THE COMPUTER
 - 1.6 LANGUAGE
 - 1.7 THE OPERATING SYSTEM
 - 1.8 THE LANGUAGE OF THE OPERATING SYSTEM
 - 1.9 TWO WAYS OF OPERATION
- [Chapter 2.- The Command Line](#)
 - 2.1 THE COMMAND
 - 2.2 THE COMMAND NAME
 - 2.3 THE PARAMETERS
 - 2.4 THE SWITCHES
 - 2.5 TYPING A COMMAND
 - 2.6 EDITING KEYS
 - 2.7 THE COMMAND.COM FILE
 - 2.8 REDIRECTION
 - 2.9 FILTERS
- [Chapter 3.- Working with Directories](#)
 - 3.1 DISK ORGANIZATION

- 3.2 DIRECTORIES
- 3.3 ACTIVE DIRECTORY
- 3.4 CREATING DIRECTORIES
- 3.5 DELETING A DIRECTORY
- 3.6 APPENDING DIRECTORIES
- 3.7 APPENDING A DISK TO A DIRECTORY
- 3.8 THE FULL PATH
- 3.9 VIEWING DIRECTORIES
- 3.10 RUNNING A PROGRAM
- 3.11 THE SEARCH PATH
- 3.12 LOOKING AT ALL DIRECTORIES
- [Chapter 4.- Working with Files](#)
 - 4.1 DOS FILE NAMES
 - 4.2 WILD CARDS
 - 4.3 TYPES OF FILES
 - 4.4 FILE SIZE, CREATION DATE AND TIME, ATTRIBUTES
 - 4.5 FINDING A FILE
 - 4.6 VIEWING TEXT FILES
 - 4.7 PRINTING TEXT FILES
 - 4.8 COPYING FILES
 - 4.9 REPLACING A FILE
 - 4.10 A SPECIAL TYPE OF COPY
 - 4.11 VERIFYING THE COPIES
 - 4.12 DELETING FILES
 - 4.13 UNDELETING FILES
 - 4.14 MOVING A FILE
 - 4.15 RENAMING FILES
 - 4.16 COMPARING FILES
 - 4.17 FINDING A STRING IN A TEXT FILE
- [Chapter 5.- Working with Disks](#)
 - 5.1 TYPES OF DISKS
 - 5.2 DISK STORAGE
 - 5.3 FORMATTING A DISK
 - 5.4 LABELING A DISK
 - 5.5 RECOVERING A DISK
 - 5.6 COPYING A DISK
 - 5.7 COMPARING DISKS
 - 5.8 ASSIGNING A DRIVE LETTER TO A DIRECTORY
 - 5.9 CHECKING A DISK
 - 5.10 FRAGMENTATION
 - 5.11 PREPARING A HARD DISK
 - 5.12 BACK UP
 - 5.13 RESTORE

- [Chapter 6.- Customizing Your System](#)
 - 6.1 CUSTOMIZING
 - 6.2 THE CONFIG.SYS FILE
 - 6.3 INSTALLING DEVICE DRIVES
 - 6.4 USING A DRIVE IN TWO MODES
 - 6.5 THE AUTOEXEC.BAT FILE
 - 6.6 INTRODUCTION TO BATCH FILES
 - 6.7 THE ANSI.SYS SYSTEM
 - 6.8 THE PROMPT
 - 6.9 ORGANIZING YOUR SCREEN
 - 6.10 CONFIGURING THE PORTS
 - 6.11 THE ENVIRONMENT
 - 6.12 RUNNING PROGRAMS
 - 6.13 MESSAGES AND DRAWINGS
 - 6.14 KEEPING RECORD OF LOGGING
- [Chapter 7.- Optimizing Your System](#)
 - 7.1 OPTIMIZATION
 - 7.2 THE COMPUTER MEMORY
 - 7.3 OPTIMIZING FOR MEMORY
 - 7.4 THE COMPUTER SPEED
 - 7.5 OPTIMIZING FOR SPEED
- [Chapter 8.- Final Considerations](#)
 - 8.1 OTHER COMMANDS
 - 8.2 CONTINUED LEARNING

Chapter 1.- Introduction

1.1 THE COMPUTER

Although there are many types of computers, this book refers to the IBM-type computers. That is, to the computers manufactured by IBM and all their clones, or compatible. You must understand that this is not a definition of a particular computer. We are only following the custom of considering IBM computers and their clones, as a type of computers. It is very difficult to imagine there are two IBM-compatible computers exactly alike. The reason is that IBM-type computers are formed by pieces. The pieces are chosen from a large selection by the system integrator.

The description that follows considers the general characteristics of the IBM-type computers. We leave for the reader to understand the peculiarities of his or her system. We will mention the circumstances when certain command or operation depends on the hardware used by the reader. You will find instances where an explanation does not apply to your computer. You should consider the same is true with the manual that came with your computer!

1.2 SAFETY

Before starting this study on how to use a computer, we must say a few words about safety. We talk about personal safety and safety of the computer itself. Do not open your computer unless you know what you are doing. It is a very good assumption that if you are reading this course, you should never open your computer. If, it becomes necessary for you to open your computer, follow these simple rules:

- 1) Turn the computer OFF;
- 2) Remove the power connection from the wall;
- 3) Wait a few minutes;
- 4) Open the computer;
- 5) Never open the video display.

About the safety of the computer, you should consider the following points:

You can damage your display by entering a wrong command;
You cannot damage the computer from the keyboard;
You can damage the computer by turning it OFF and ON immediately;
You can damage the computer by hitting or shaking it while ON;
You can damage the computer by dropping it and turning it ON.

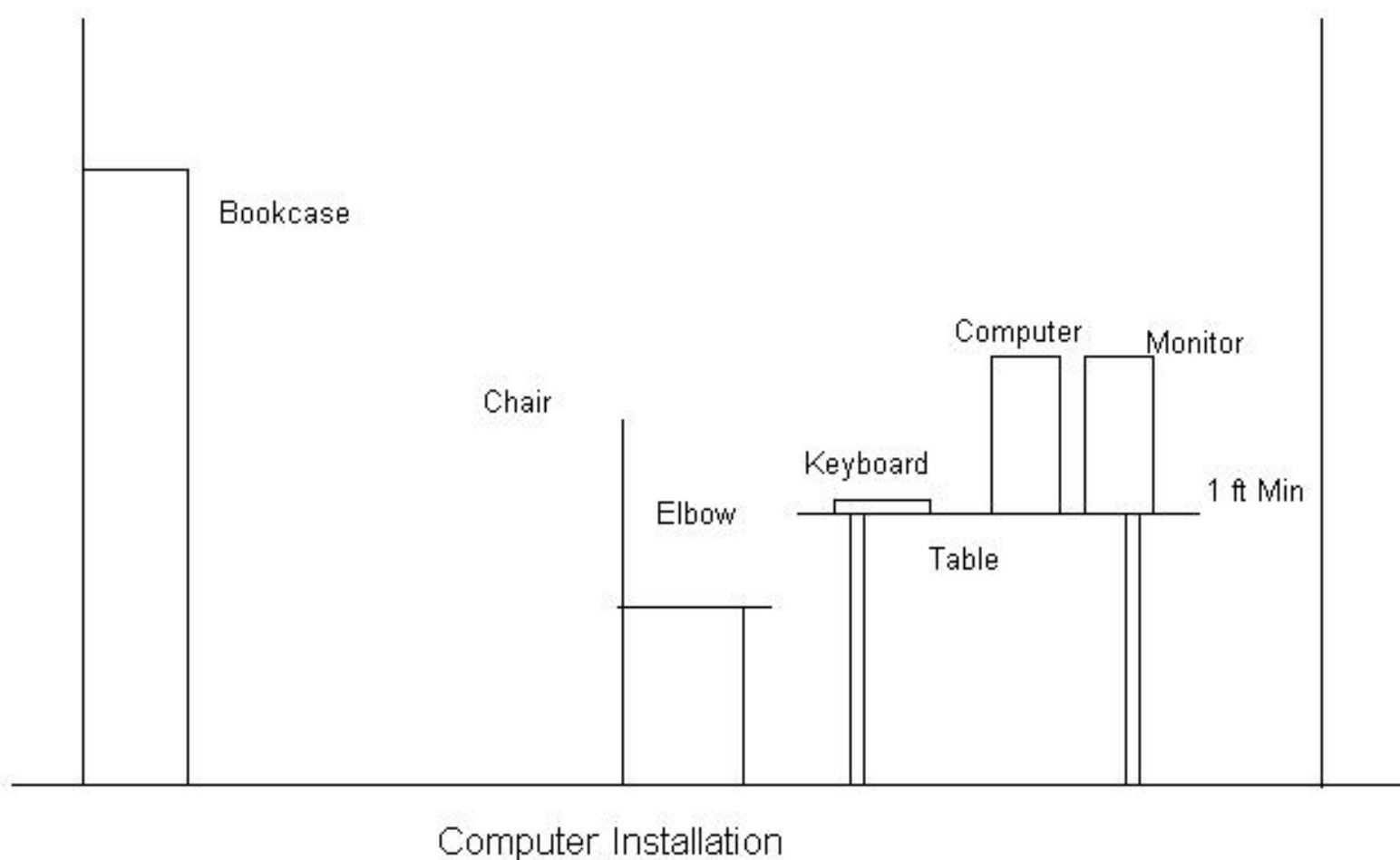
You get the idea.

If you hit or drop the computer, some of the parts might move or disengage. Turning the computer ON under these circumstances is asking for trouble. If you drop your computer, ask a friend to look at it. Let him turn it ON. The same is true with moving the computer while it is ON. You can cause permanent damage to your

drives. If you treat your computer carefully, it will last you for so long that it will become obsolete before it fails.

1.3 A WORD ABOUT INSTALLATION

Start with the chair. Use a chair where you can sit comfortably. Use a table such that the face of the keyboard is at the level of your elbows. Locate the video display a little lower than your eyes, and not too close. Locate the box of the computer in a way that it has ample ventilation. All computers have a fan in the back. Do not block the flow of air. They take air through several openings. Do not block them. Be sure that you will not tangle your feet with the many cables the computer ends up having.



A bad installation can cause you to become tired. Your arms or your back might ache after working a few minutes with the computer. You might need to stand after a few minutes of work. Your eyes or your neck might ache during a computer session. Check the installation of your computer. Measure the height of the chair and the table. Consider the height of the monitor. Changing them might end your troubles.

Connect all elements of your computer to a single wall outlet. Better yet, use one of those multiple outlets with transient suppression. If the computer has an outlet for the monitor, use it. All this prevents damage from

lightning or power transients. A very good idea is to get an uninterruptible power supply. They are not too expensive. They prevent the computer from turning OFF at every transition of the power.

1.4 WHAT A COMPUTER CAN DO

I used to tell my students that a computer can do anything. You need to tell it how to do it. I used to tell them if you knew how to tell the computer how to talk, it will talk. Today anybody can make a computer talk.

A good way to think about a computer is as an extension of your brain. Anything you normally do with your brain, you can do it with the computer in a more efficient way. Consider doing arithmetic. You can do it with pencil and paper. You can do it better and faster with a computer. Consider writing letters, articles, or books. You can do it by hand or with a typewriter. You can do it better and faster with a computer. Notice I said "You can do it". The computer cannot do it without you.

1.5 TALKING TO THE COMPUTER

Think about the computer this way. You will understand that the computer can do only what you, or an agent of you, tell it how to do. The way you tell the computer how to do something is with a program. A program explains the computer the generalities of a particular job. You will need to give commands for the details of the job. Let us consider writing letters. You get a program called word processor. This program tells the computer, in general terms, how to write whatever you want to write. It tells it how to accept your information about margins, format, etc. It tells it how to print the document the way you want. You need to give commands to the program, so it can do what you want the way you want.

A similar example is what will take most of our time. We will study the Operating System. The Operating System is a program that tells the computer how to do the routine operations you might want to perform. When you are working under the Operating System, you need to give the computer commands. These commands will tell the computer the operations you want to perform.

1.6 LANGUAGE

For the computer to do something useful, you have to tell it what you want it to do. For this purpose you need to use a language the computer understands. Most of the mystique about computers comes from the languages used by the first computers. They were very complicated to understand and use. Today, there are very simple languages you can use to tell the computer what to do. Going back to the examples above, the Operating System simplifies very much operating the computer. A command like COPY takes care of all the internal operations to copy a file. The user does not need to worry about how that is done. You might use a graphic shell or interface, like Windows. You copy a file by dragging it with your mouse to where you want it to be.

1.7 THE OPERATING SYSTEM

We call Operating System several programs to perform all the internal operations we wish to perform with the computer. There are several operating systems for IBM-type computers. The most common ones are the MS DOS, or Microsoft Disk Operating System. You also have DR DOS, or Digital Research Disk Operating System. The IBM DOS, or IBM Disk Operating System is also used. OS/2 is another operating system developed by IBM. At the command level, they are very similar. You need one of these programs between you and your computer.

Computer manufacturers do not provide for any way to work with a computer without an operating system. The Operating System controls the computer. You tell the operating system what you want the computer to do. The Operating System tells the computer how to do it.

1.8 THE LANGUAGE OF THE OPERATING SYSTEM

You need to tell the Operating System what you want the computer to do. You have to use the language the Operating System understands. The language of the Operating System is a simplified version of English. It is a group of single words and parameters. The purpose of the parameters is to change or qualify the command. Consider the COPY command. You need to say which file to copy and where to copy it. There are other cases that are more complex.

When you wish to see the contents of a disk, you use the command DIR. You need to say which disk you wish to see. You also can specify /p to pause at the end of each page. You use /w to use the wide format. Use /s to display the files in the directory and all sub directories. Use /a to display only files with certain attributes. As you see, there is large flexibility on the language of the Operating System.

The rest of this book studies the different commands. We start with the most important and useful commands, so you can start working immediately.

1.9 TWO WAYS OF OPERATION

There are two ways of using an IBM-type computer: The command line or with a shell or interface. The command line is the primitive way the operating system works. You type a command on the keyboard. The command is executed when you press the Enter key. A shell or interface is a program designed to simplify entering commands. They avoid, wherever possible, the need to type the command.

There are many types of shells and many programs of each type. In general, there is the graphics interface and the text interface. A typical graphics interface is WINDOWS. A typical text interface is DOS-SHELL. It is a good idea to learn how to use the command line before starting with a shell. In this way you can use all the capabilities of the shell.

Chapter 2.- The Command Line

2.1 THE COMMAND

The basic way of working with DOS is the command line. When using the command line, you type commands through the keyboard. Commands are entered into the command line one at a time. The command is executed when you press the Enter key. Before pressing the Enter key, you have limited capabilities for editing the command.

A command has several parts. They are the command **name**, the **parameters**, and the **switches**. The command name tells the computer what you want it to do. The parameters tell the computer how you want it to be done. One very important point to mention is that the command must be typed exactly as it should be. Any difference between the spelling of a command and how you type it brings the message: Bad Command or File Name This is one important reasons for using batch files, shells, or interfaces. Using a batch file gives you a chance to review all the commands before they are sent to DOS. Using a shell or interface, you have alternative ways of performing the functions. This is true only for a good shell or interface. There are many popular shells or graphic interfaces where you need to type the commands like in the command line. A limited shell simplifies some operations but not all. We will study this later.

2.2 THE COMMAND NAME

The command name is a single word that say what the computer is to do. Typical commands are COPY, DIRECTORY, DELETE. There are some cases where the command consists of two words joined. This is the case of CHANGE DIRECTORY, CHECK DISK, CLEAR SCREEN, DISK COPY, and many others. Usually, the command word is abbreviated. For example, instead of saying DIRECTORY, you say DIR. Instead of saying DELETE, you say DEL. The same with other single word commands. There are some cases where you can use the whole word or an abbreviation. This is the case of RENAME. You also can use REN. The case of double word command names is different. In this case you have to use the abbreviated word. CHANGE DIRECTORY is abbreviated into CD. CHECK DISK is abbreviated into CHKDSK. CLEAR SCREEN becomes CLS. The same with many other commands. If you type CHECK DISK the computer tells you that CHECK is not a command. The same is true with all the other complex words. There are some cases where you have a choice. You can type MKDIR or MD to MAKE DIRECTORY.

2.3 THE PARAMETERS

The second part of a command is formed by the parameters. They tell how you want the command be performed. A good way to see this is with an example. The command DIR asks DOS to read the content of a directory of a disk, and show it on the screen. DIR is the command name. You need to tell DOS

which directory and in which disk. This is the parameter. You also must tell DOS the details of the display. These are the switches. The command DIR, for example, can be DIR C:*.* /p /w. This means you wish to see the files in the root of drive C:. It also tells you wish the computer to pause at the end of the screen before scrolling. You also wish the computer to use the wide format. You will study all this in detail in a later chapter.

The important point now is to mention that every command can have up to three parts: Name, Parameters, and Switches. Some commands do not accept switches. Most of them require parameters. It is interesting to mention here that certain parameters and switches are defaults. This is to say that if you do not need to give a parameter or a switch. The computer uses the default values. In most cases the defaults are set in DOS and cannot be changed. There are a few cases where you can set the default values.

2.4 THE SWITCHES

You need to be careful with the use of the switches. The problem comes from the way DOS has been written. There is no relation between the letters used for the switches in one command and the letters used in another. Further more, within the same command the same letter is used for different meanings. A good example is the command ATTRIB. It has three switches with the letter s. You use +s to set the system attribute of the file. You use -s to clear the system attribute. You use /s to include all the files in the current directory and all its sub directories. The switch /s means most of the time, "include all the files in the current directory and its sub directories". This is not always the case. When you use the FORMAT command, the same switch /s is used to get a system disk. The same is true with the use of the defaults. It not easy to see which switch is the default for each command. The case of COPY is a good example. One use of the COPY command is to concatenate files. Copying files one by one defaults to binary files. Concatenating files defaults to ASCII or text files. There is a moral of all this. You should be careful how you use the switches. Unless you use a command every day, check your book for the use of the switches.

2.5 TYPING A COMMAND

Typing a DOS command is easy. At the prompt, you simply type the command, the parameters, and the switches. You can use upper, lower, or mixed case. It does not make any difference. Leading blanks have no meaning. More than one blank between parts of the command has no meaning. In most cases the order of the switches is immaterial. The important point is that the command always goes first. You cannot insert blanks in any of the parts. A blank separates parts. Certain characters, like the comma, have special meaning.

For example, if the command is COPY, you cannot type COP Y. If the name of a file is FILES.TXT, you cannot say FILES . TXT, or FILES TXT. Note that some times, the DIR command displays the files this

way. You cannot use this form in a command. The same is true with the switches. If you enter / a it does not mean /a. You get the idea.

2.6 EDITING KEYS

While you are typing a command you have very few chances of correcting what you type. You can only use the Backspace key to erase the previous letter. If you made a mistake close to the beginning, you can use the Backspace key until you get to the mistake. The Esc key erases the whole line and start over. After you execute a command, whether it works or not, you have some more ways of editing the command line. Note that all this refers to the previous command line you just executed. The idea is that the old command line is a template. You copy one characters or several from the template to the new command line. You can use the following keys:

F1, copies one character from the template to your command line;

F2 n, copies up to the first occurrence of the letter n;

F3, copies the rest of the line;

F4 n, skips up to the first occurrence of the letter n;

F5, shows the rest of the line and starts over;

DEL, skips one characters of the template;

INS, inserts whatever you type onto the command line;

BCKSP, deletes the last letter;

ESC, deletes the whole line;

ENTER, executes the command.

The symbol n used above represents any letter or symbol you enter after F2 or F4. The insert ends when you press Enter or any function key. If you type any letter, it replaces the next character of the template. Let us see two examples. Say, the template is

```
COPY B:\FILE.TXT B:\*.*.
```

What you wish to say is FILE not FIILE.

You press F2 i. This brings COPY B:\F to the screen.

You press DEL and nothing shows on the screen.

You then press F3.

The screen shows the command line with a single I.

Another use of this procedure is when you need repeated commands. Say you have several files, called FILE01.TXT, FILE02.TXT, through FILE54.TXT. You wish to copy FILE12.TXT, FILE23.TXT and FILE33.TXT. You type the first line as normal

`COPY FILE12.TXT B:*.*`.

Then, instead of retyping the whole line you simply type F2 1. This copies

`COPY B:\FILE` to the screen.

Then you type 23. The screen now shows

`COPY B:\FILE23`.

Finally, you press F3 to complete the line.

The same procedure produces the next command replacing the 2 with a 3. Instead of using F2 you can use F1 several times. The same can be done with DEL and F4. It is a matter of taste.

2.7 THE COMMAND.COM FILE

When you start the computer, after the initial exploration and checking of the system, the computer reads and executes a program called `COMMAND.COM`. This program contains the most commonly used functions, like loading a program, handling errors, etc. This program also executes the most common commands. They are called internal commands. At the beginning, you do not need to be concerned too much with this fact.

There are some commands that are internal. They exist in the part of DOS that is always in the memory of the computer. Other commands are external. They exist in a disk and not in the memory of the computer. External commands are not used than often, so they are not loaded onto memory. When you need them, `COMMAND.COM` goes to the disk to find and execute them. This is not too much of a problem if you have DOS in a hard disk. This is a problem when you run DOS from a floppy disk. In this last case, you need to have the disk in drive a: to execute external commands.

2.8 REDIRECTION

Every command that accepts an input or produces an output can have this input or output redirected. The normal input device is the keyboard. The normal output device is the screen. If you wish a command to take input from a file `COMMAND.TXT`, you need to use the input redirection symbol (<). Say, the command `FORMAT` requires you to give some input. If you enter the command as `FORMAT < COMMAND.TXT`, the input will come from the file and not from the keyboard. It is also possible to redirect the output of a command. The normal output is to the screen. You can use the symbol (>) which redirects the output. Say you wish to get the directory of drive C. You wish the output to go to the file

DRIVE-C.DIR. Entering the following command `DIR C:*.* > DRIVE-C.DIR` produces that result. To get the directory printed you say `DIR C:*.* > PRN`. Recall PRN is the file name of the printer.

2.9 FILTERS

Filters are small programs that perform specific functions. DOS has two filters, **SORT** and **MORE**. SORT takes input from a file, or the keyboard and sorts the lines the file alphabetically. SORT takes two file names, the input and the output. The normal format is

```
SORT < INFILE.TXT > OUTFILE.TXT.
```

The redirection symbols are required. SORT accepts two switches: /r to ask for sorting in reverse order. Switch /+n to start the sort with column number n. The first column is number zero. The defaults are sorting alphabetically from a to z, starting with the first column. SORT requires that each line ends with carriage return. You can use SORT with a command. Say you wish to sort the output of DIR, you say `DIR C:*.* | SORT`. Note that now you use the pipe (|), instead of the redirection. The reason for this is that DIR is a command, not a file.

The other useful filter is MORE. The work of MORE is to paginate the display of any text on the screen. Imagine you have a large directory that fills more than one screen. You say

```
DIR C:\*.* | MORE.
```

Note that in this case you use the pipe (|) to say to go through the filter. The reason is that DIR is a command. Naturally, you can use both filter one after the other. For example to sort the directory of C: and pause after each screen, you can issue the command

```
DIR C:\*.* | SORT | MORE.
```

You could add `> DRIVE-C.DIR`, to redirect the output to a file.

Chapter 3.- Working with Directories

3.1 DISK ORGANIZATION

You can consider a disk as a drawer of your desk. You can put anything you want in a drawer. Imagine you put letters, writing materials, your lunch, a coffee jar, cups and spoons. There is nothing wrong with this. The letters will have coffee stains. The coffee might have paper clips. You might not mind either. It is much better to organize your desk with partitions. Each partition contains similar elements. It has two advantages. One, nothing gets mixed up. It is easier to find anything.

The same happens with a disk. You could put your programs and files mixed up in the same disk. First, it is harder to find anything. Second, files get confused. Consider a simple case. Most programs come with a file called READ.ME, or something similar. Say you put your programs together. You do not know which file corresponds to which program.

It is possible to arrange partitions in a disk. In the jargon of IBM- type computers, these partitions receive the name of directories. You can have directories for each kind of program. Say, you have several directories. One for utilities, others for editors, financial programs, pictures, and whatever else you need. Talking about floppy diskettes, another arrangement becomes more convenient. Floppy diskettes are inexpensive. You can have one diskette for each kind of program and directories for each program.

The problem of fixed or hard disks is similar. The problem in this case gets complicated because hard disks have a much larger capacity. You must create directories to divide the hard disk into manageable sections.

3.2 DIRECTORIES

A good arrangement is to create a few directories, one for each kind of work you do. You create several directories inside each directory. Each of these sub directories contains programs of a particular topic. You create more sub directories inside each topic. They hold all the files for a specific program. In this way, you can easily find any program or file you need. The same arrangement works with data files. One possible arrangement is to put the data files in the directory of the program. Some times is better to put the data files in their own directory. You might have many different types of data files. You can organize them by types in sub directories. They all go into their own DATA directory.

This hierarchical organization has many advantages. Your work gets simplified very much if you give good names to the directories. The names should suggest what is inside. You can call a directory with any name. Your directory for letters can be L. Or it can be P. You gain nothing by using a single letter. Use full words. MS DOS always uses 12 characters for the name. You make your work easier if you use descriptive names. The name of a directory can have up to 8 characters, a period, and an extension. [WINDOWS 95 allows you to use longer names] The names of the directories follow the same rules of

the names of the files explained below. Most people do not use extensions with the name of directories. I suggest you use an extension with the name of the files. I suggest you do not use extension with the name of directories. This makes it easy to recognize directories and files.

3.3 ACTIVE DIRECTORY

DOS maintains one active directory for each drive in your computer. You change from one drive to another by typing the letter of the drive followed by a colon. In this way, D: will take you to the active directory of drive D. You can change the active directory with the command **CHDIR**, that can be abbreviated **CD**. This command requires you tell the name of the directory you wish to make active. There are some tricks. Imagine you wish to make active a sub directory of the active directory. In this case you only need to say CD and the name of the directory.

Let us illustrate with an example. Imagine your drive C has three directories: DOS, PROGRAMS and UTIL. Directory PROGRAMS has two sub directories: ARCHIVE and WP51. Directory UTIL has three sub directories: PROGRAMS, PICTURES, and TEXT. Directory TEXT has three sub directories: LETTERS, ARTICLES, and DATA. Let us draw a picture, which is called a tree:

```
C:
  DOS
  PROGRAMS
        ARCHIVE
        WP51
  UTIL
        PROGRAMS
        PICTURES
        TEXT
        LETTERS
        ARTICLES
        DATA
```

Imagine you are in the root of C. To change the active directory to PROGRAMS you only need to say CD PROGRAMS. To change directly to WP51 you need to say CD \PROGRAMS\WP51. The same works for all others. Imagine now that UTIL is the active directory. You wish to change to DATA. Again, you say CD TEXT\DATA or CD \UTIL\TEXT\DATA. If you wish to go to another path, like to go to WP51 from DATA, you say CD \PROGRAMS\WP51. Notice the beginning back slash. It shows the new path starts at the root of the drive.

There are two short cuts. Say you are in DATA. You say CD .. and you end up with TEXT as the active directory. That is, the two dots bring you to the directory that contains the active directory. Thus, it takes you one step closer to the root.

There is another short cut. You say, wherever you are, `CD \`. This command takes you directly to the root of the drive.

If you type only `CD`, DOS responds by showing the current drive and path to the active directory.

3.4 CREATING DIRECTORIES

You create a directory with the command **MKDIR** that can be abbreviated to **MD**. The use of this command requires that you give the name of the directory you wish to create. The procedure is very similar to the one explained above for changing to a directory. If you are in the directory where you wish to create the new one, you need only to give the name. If you are in another directory, you need to give the full path as explained above.

3.5 DELETING A DIRECTORY

Deleting a directory has some tricks. The basic command is **RMDIR** that is abbreviated to **RD**. You follow the command with the name of the directory you wish to delete. This name follows the same rules given above. The trick in deleting a directory is that you can only delete a directory that is empty. The directory cannot have files or sub directories. So, you need to remove first all the files and all the sub directories before you can delete a directory.

3.6 APPENDING DIRECTORIES

Some times you want to make two directories behave like if they were one. This is the case when different files needed by a program are in two directories. Say, one directory has the program and the other has the data. For this purpose one directory is appended to the current or active directory. The command **APPEND** permits you to do just that. The format of the command is `APPEND path`. For example, you wish to have the directories `C:\DATA` and `C:\UTIL` behave like one. You make `C:\DATA` the active directory. Then, you say `APPEND C:\UTIL`. You cancel this operation with the command alone. `APPEND ;` cancels any appending that has been done to the current directory. Note the use of a semicolon after the space.

3.7 APPENDING A DISK TO A DIRECTORY

Some times you might want to append a disk to directory. The command **JOIN** permits you to do this. The format is

JOIN DRIVE PATH.

All the files and directories in the **DRIVE** are considered part of the **PATH**. The path, of course, can be only a drive. This actually join both drives into one. This command accepts only one switch. The switch **/d** cancels all assignments made before. The format is

JOIN DRIVE **/d**.

3.8 THE FULL PATH

This is a good place to tell the way you specify the location of a file without ambiguity. You start from the drive letter. You add a colon. Then a back slash. Then, the name of each directory and a back slash. This continues until you get to the directory where the file is. Then, you put the name of the file. Using the example above say that you have a file **PLANTS.DAT** in the directory **DATA**. The full specification for this file is

C:\UTIL\TEXT\DATA\PLANTS.DAT.

It is very important to give the full path of a file wherever there is a chance for mistakes.

3.9 VIEWING DIRECTORIES

You need to see what files are inside a directory. DOS gives you a command to do it. This is the command **DIR**. This is a very important command. You will use it very much. It is very important you get familiar with it. The command **DIR** takes the name of the directory you wish to see. The default is the current directory. In general you say **DIR C:*.***. This means you want to see all the files in the root directory of drive **C:**. It does not mind which is the active directory of **C**. If you wish to see only the executable programs, you say **DIR C:*.EXE**. Any group of files can be specified using wild cards.

The command **DIR** takes five switches:

/p to pause display after each page.

Switch **/w** asks for the wide format.

Switch **/s** asks for the main directory and all its sub directories.

Switch **/a:attr** asks to display files with or without a given attribute.

Switch **/o:xxx** asks for the files in a particular order.

Let us start with the format. MS DOS provides two formats, vertical and wide. The vertical format is the default. The vertical display produces one line for each file. It includes the name of the file, its size, date and time of creation. The wide format shows only the name of the files five on each line. The vertical

format displays the directories with <dir> in the place of the size. It gives the date and time of creation. The wide format displays only the name of the directories.

The attributes of a file are r for read only, h for hidden, s for system, d for directory, and a for archived. You use the letter to ask for those files with that attribute. You add a dash (-) to ask for those files without that attribute. Say you wish to display only the directories. You use /d. Say you wish to display whatever is not a directory. You use /-d. The same with all the others.

The files in a directory are displayed in the natural order. That is, in the order they are in the directory. You can specify to use a different orders by using the switch /o. You use n to get them ordered by name. You use e to get them ordered by extension. Use d to get then ordered by date. Use s to get them ordered by size. Finally, g puts the directories above the files. The format is /o:e, or whatever letter you wish. The ordering is normally in ascending order. You add a dash (-) to get reverse order.

Finally, we should repeat the examples we included before. If you wish to print the directory, you redirect the output to the printer with > prn. If you wish to have it in a file, you redirect to the file.

3.10 RUNNING A PROGRAM

Running a program only requires that you type the name of the program. This requires that the active directory is where the program is stored. This is required when the program has other auxiliary files it must find. Some programs do not require any auxiliary files. You call them from any place in the system with the full path, as explained above.

This is a good time to insert an analysis of how DOS finds the programs you wish to run. Consider first the case you give only the name and extension of the program. DOS searches first the active directory. If the program is not there, then it searches the root directory of the current drive. The program might not be there. It gets to the environment variable PATH. It searches each of the directories in the path. If DOS does no find the program in any of them, it gives an error message.

Consider now that you only tell the name of the program, but not its extension. DOS searches first as explained above for a file with that name and the extension COM. If it does not find any, it repeats the search for a file with that name and the extension EXE. Finally it searches for a file with the given name and the extension BAT.

3.11 THE SEARCH PATH

The search path is in the environment variable **PATH**. Environment variables are special variables that DOS passes to each program when it runs them. DOS also uses the environment variables as explained above. An environment variable is created or changed with the command **SET**. The format is

SET VARIABLE=STRING.

VARIABLE is the name of the environment variable and **STRING** is the value given to that variable. You can see all the defined environment variables by typing only **SET**. In particular, the variable **PATH** contains all the sub directories you wish DOS to search when looking for a program. You set the value of this variable by typing

SET PATH=C:\;C:\DOS;C:\WINDOWS.

This example asks DOS to search first the root of drive C:. Then, to search the sub directory **DOS** from C. Finally, to search the sub directory **WINDOWS** from C. The name of each directory requires the full path. The names are separated by semicolons.

Many programmers tell you to add to the **PATH** the directory of their programs. If you follow this advice, very soon all the directories of your computer are in the **PATH**. This has some problems. Imagine you have a program some place called **PICTURE.COM**. You already forgot you ever had it. In some other directory you have a program **PICTURE.EXE**. Both directories are in the **PATH**. You try to run the program **PICTURE.EXE** and you only give the name **PICTURE**. You will run the program **PICTURE.COM** not **PICTURE.EXE**. You might have forgotten you have it. You will have a surprise.

There is another drawback to have a very long path. There are some users that need three or four lines to complete their **PATH**. Trying to run a program with extension **BAT** takes a long time. DOS looks all the many directories for program with extension **COM** or **EXE** and the same name. If one exists, you do not get your batch file. If none exists, it might take several minutes before the batch file runs.

A better idea is to have a **RAMDISK** and copy there all the programs you use all the time. Make your **PATH** very short. Put first the name of the **RAMDRIVE**. Then probably the directory where you have the DOS files. If you have **WINDOWS**, you put its directory next. That is all. You will find more about this when you study optimizing your computer.

3.12 LOOKING AT ALL DIRECTORIES

Some times you wish to look at the full structure of your disk. There is a command, **TREE**, that shows this structure. The command **TREE** accepts a drive and a path. It shows all directories starting from the specified directory. **TREE** accepts two switches: **/f** shows all the files in each directory; **/a** uses text characters to show the connections between directories. The default is to use graphical characters. You can use the filter **MORE**, to force displaying the tree in pages. You can redirect the output to the printer or a file. It is a good idea to get the tree of your hard drive and redirect it to a file. In this way you could reconstruct the structure if you change your disk.

Chapter 4.- Working with Files

4.1 DOS FILE NAMES

Anything you store in a disk, whatever it is, receives the name of file. Any file has to have a name. Each type of computer has its own rules for the names of the files. IBM-type computers follow the rules given by MS DOS, the operating system. A name has up to 8 characters, followed by an optional period, and an optional extension. [WINDOWS 95 allows the use of longer file names] The extension has three or less characters. If you include the extension, you need the period. It is a good practice to include an extension.

Valid characters are all the letters, all the numbers, and some special characters. The following special characters can be in a file name or its extension: **Underscore** (`_`), **caret** (`^`), **dollar sign** (`$`), **tilde** (`~`), **exclamation point** (`!`), **number sign** (`#`), **percent sign** (`%`), **ampersand** (`&`), **hyphen** (`-`), **braces** (`{}`), **parenthesis** (`()`), **at sign** (`@`), **apostrophe** (`'`) and the **grave accent** (```).

MS DOS accepts no other special character. You can use `_ ^ $ ~ ! # % & - { } () @ ' ``

MS DOS does not accept the following characters in a name. They have special meaning. **Space**, **comma**, **back slash**, and **period** cannot be in the name or extension of a file. It was mentioned before that a period separates the name from the extension. A space or a comma mark the end of the name. A back slash separates the name of a directory from the name of the file.

Extended characters are those with values above 127. You can use them in file names if you know what you are doing. It is safer not to use them because they create confusion.

There are a some reserved names: **CLOCK\$, CON, AUX, COMn, LPTn, NUL, PRN**. The letter n represents any number. These names have special meanings you will study later.

The letters can be uppercase or lowercase. It does not make any difference. **APPLE.TXT** is exactly the same file as **apple.txt**. Actually, MS DOS will consider both equal to **APPLE.TXT**.

There are some reserved extensions: **EXE** means executable file or program. **COM** means command file, a special type of program. **SYS** means program used by MS DOS. **BAT** is a special type of text file with commands MS DOS executes. The extension normally defines the type of file. Each program recognizes files with special extensions. Most people uses the extensions **TXT** and **ASC** to show the files contain only text characters. Underline, page brake, or similar characters, do not appear in these files. Every program that works with files has its preferred extension. Most personal computer users accept and recognize some extensions. This is the case of **ZIP, ARC** and **LZH** for compressed files. **GIF, PCX, RLE, WPG**, and many others for pictures and graphics.

When selecting the name for a file try to be explicit. A good name tells you what is in the file. If the file is a program, a good name tells what the program does. If the file contains information, a good name

shows what type of information. I am sure you can guess what a program called SCRNSAVE.EXE does. You also can guess what a file DISKDATA.TXT has. On the other hand, consider a program called DF.EXE, or FE.EXE. It is hard to guess that the first one is a file manager and the second one is a database program. Both are very good programs. Their names are bad.

Avoid using the same name for different programs or files. The exception is when several programs perform the same function. They might be updates of each other. They might be parts of a larger whole. In this case, use a suffix. Examples are GRABBER.EXE, GRABBER1.EXE, GRABBER3.EXE, can be updates of the same program to grab something. The same happen with the extensions. If you create files, try to use consistent extensions. Say you use Word Perfect. Use the extension WP for all the files produced with this word processor. If you also use another word processor, do not use WP as extension for its files. Use another extension that will tell you which word processor you used. Avoid extension used by commercial programs for files you produce.

It is common to have several related files with the same name. For example, you have a program called MOVIES, that is used to store the movies you have on tape. You have a file MOVIES.EXE, the program. You have a file MOVIES.DAT, the data on the movies. A file MOVIES.DIR is a directory of movies. A file MOVIES.CNF contains the configuration you use to display the movies.

4.2 WILD CARDS

Practically every command refers to files. Most of the operations you perform involve files. In many instances you wish the command to act on a group of files. DOS allows you to use two wild cards. A wild card is a symbol that matches anything in that position.

The simplest wild card is ?. The question mark matches any character in that position, including none. Consider you have several files called FILE1.TXT, FILE2.TXT, FILE3.TXT, and many more. All have a name that starts with the letters FILE, then a single digit number or letter, and then the extension TXT. You can refer the all of them by saying FILE?.TXT. The ? matches with the 1, the 2, the 3, and any other character in that position. Note that a file called FILE12.TXT does not match that call. Note also that FILE.TXT matches that call.

The other wild card is the *. The asterisk matches any number of symbols in that position, including none. In the previous example calling FILE*.TXT will catch all the files. If you call FILE*.*, you catch all the files with names that start with FILE, with any extension. A call with *.* catches all files in the directory that is shown.

There are some cases where you cannot use wild cards. This is when the command applies to only one file. This is the case of programs. You cannot run several programs. You can run only one. You need to specify exactly which one you wish to run.

4.3 TYPES OF FILES

It was mentioned, there are different types of files. The extension shows the type of file. In general, files can be classified in many different ways. There are commands that require that you tell the type of file you are working with.

One important way of classifying files considers the way the information is stored. There are **binary** files, where the information is as binary numbers. There are **text** files, where the information is characters. Note that this is a definition of convenience. Both files contain binary numbers. The idea is that text files contain binary numbers that represent characters. It is usual to talk about binary and text files. The big difference is in the way the commands handle these files. Typical binary files are programs, whether EXE, COM, or SYS. Typical text files are the documentation of the programs, the source code, letters, articles, and similar files.

Other ways to classify files are as **programs, data, documentation, letters, pictures**, and many others. These classifications are not important for the way the commands handle the files.

4.4 FILE SIZE, CREATION DATE AND TIME, ATTRIBUTES

When a file is created, or changed, the computer stores the file on disk. It puts in the directory the file name and its extension. It also puts the size of the file, the date and time the file of creation, and the attributes of the file. The name and extension were mentioned before. The date and time of creation or change tells which file is the oldest or the newest.

The attributes of a file are important. They show special characteristics of the file. Each file has four attributes:

the **r** bit, is zero for files that are for read and write; is one for read only files, cannot be written;
 the **h** bit, is zero for files that can be seen in the directory; is one for files hidden from any search;
 the **s** bit, is zero for normal files; is one for system files;
 the **a** bit, is zero for files not backed up; is one for backed up files.

You can find or change the attributes of a file with the command **ATTRIB**. The format of this command is **ATTRIB switches filename**. The letters r, h, s, a are used with a + sign to set the attribute of the file, with a - sign to reset it. Imagine you have a file called FILE.TXT. You want to make it read only. The purpose is to prevent damaging the file by mistake. You issue the command

ATTRIB +r FILE.TXT.

Say, you want to find out what attributes the file FILE.TXT has. You issue the command

`ATTRIB FILE.TXT.`

Note that you do not specify any of the switches. You say

`ATTRIB /s FILE.TXT`

to find a file is in the directory or its sub directories, . The switch /s asks to search for the file in the directory and its sub directories.

Say, you have several files called FILE, with different extensions. You wish to find the attributes of all of them. You enter the command

`ATTRIB /s FILE.*.`

Recall that the * is a wild card. It matches any extension.

Another case. Say you have several files with names starting with FILE. The case is like FILE1.TXT, FILEONE.ASC, FILE32.TXT, and others. You can get the attributes for all of them by saying

`ATTRIB /s FILE*.*.`

4.5 FINDING A FILE

Let us use this moment to explain an application that is useful and elegant. Many times you forget where you put a file. Other times you want to know if you already have a file with a given name. There are many complex program floating around to find a file. The **ATTRIB** command permits you to do it in a very simple way. Say you wish to find where is your program WHEREIS.EXE. You know you have it some place in your hard drive. The hard drive is C: and you make it the active drive. You apply the knowledge you just gained. You need to do is to issue the following command:

`ATTRIB /s WHEREIS.EXE.`

Let us analyze it. You are asking DOS to give you the attributes of a file called WHEREIS.EXE. You also ask to search the current directory (the drive C:) and all its sub directories. DOS responds by giving you the attributes. It also gives you the whole name of the file, including the directory. If you have several files whose names start with WHERE, you can find all by typing

`ATTRIB /s WHERE*.*.`

You get a list of all the files with names starting with WHERE. The list has the place where they are and

their attributes.

4.6 VIEWING TEXT FILES

Often enough you need to read a text file. Program usually come with instructions in the form of text files. You will study now another simple command that will perform this function. The command is **TYPE**. It requires you specify a file name. The file appears on the screen the way it was created. Note that if the file is not a text file, the screen interprets each byte as a character. You get funny characters on the screen with little meaning. You already studied how to prevent the file from rushing through your screen. This requires the use of Control-S to stop the display and Control-Q to restart. This not always the easiest way to do it. Some times it is better to use the pipe (|) to **MORE**. You can view a text file by using the **COPY** command. You find this when studying the **COPY** command.

4.7 PRINTING TEXT FILES

There are several ways to send a file to a printer. You can use the command **TYPE** and redirect the output to the printer. The command is then:

```
TYPE FILE.TXT > PRN.
```

Recall that **PRN** is the name of the printer. The second way to send a file to the printer is to use the command **PRINT**. One advantage is that this command permits you to send several files to the printer, up to the capacity of the printing buffer. You can continue working while the printer prints. You can see a list of files waiting for printing by using the command alone as **PRINT**. You can cancel the printing of a file by using

```
PRINT filename /C.
```

Using wild cards cancels all that match. You can cancel all the waiting files by saying

```
PRINT /T.
```

You can print a file by copying to the printer. You find this with the **COPY** command.

4.8 COPYING FILES

Probably the most common operation is to copy a file from one place to another. You use the command **COPY**. The structure is as follows:

COPY which file to where.

Specify where you want the file, permits you to give it another name. Using wild cards you can copy many files with a single command. You can copy all the files in the source directory by using the *.* specification. Although the command is simple, there are some switches you can use. The command copies considering them binary. This is the default. This means that the file does not have an end-of-file marker. If you wish to have an end-of-file marker, you need to specify text files. You do this with the /a switch. The /b switch forces to consider the files binary. These switches go after the **COPY** command. They also can go after the name of the file to copy, or at the end of the command. The usual way is just after the **COPY** command.

A variation of the copy command is to append files. That is, you give a list of files. They are all put one after the other, in a single file. To do this, you use the + sign. Consider you have three files, JAN.DAT, FEB.DAT and MAR.DAT. You wish all three files converted into one called 1992.DAT. The command is

```
COPY JAN.DAT+FEB.DAT+MAR.DAT 1992.DAT.
```

One important point is the defaults. The **COPY** command assumes that the files appended are text files. Note that in single file copy it assumes the files are binary. If the files are binary, you need to add the /b switch. To append binary files you need to issue the command

```
COPY /b JAN.DAT+FEB.DAT+MAR.DAT 1992.DAT.
```

The difference in this case is important. Consider the files are binary. They might contain the code for end-of-file as a data point. The copy truncates the file at that point.

The command **COPY** also accepts the switch /v. It asks for verifying that the copy was correct. This switch only has effect on the copy of the file or files specified in that command. See the command **VERIFY** for a permanent verification.

The input or the output of the **COPY** command can come from devices. Recall that DOS also treats the devices as files. Each device has a file name. The console is **CON**. The printer is **PRN**. The serial port is **AUX**. Then, you can send the file **READ.ME** to the screen by saying

```
COPY READ.ME CON.
```

You can print the same file by saying

```
COPY READ.ME PRN.
```

Some times you need to develop small text files. The use of a word processor or a text editor is not justified. You can use the **COPY** command. Say you wish to develop a small text file called **READ.ME**. You enter the command

```
COPY CON READ.ME.
```

This creates a file called **READ.ME**. The computer puts there everything you type on the console or keyboard. You finish the file with an end-of-file character. You produce it either with a Control-Z or with the F6 key. When you press Enter, the computer responds with 1 File Copied.

4.9 REPLACING A FILE

The command **REPLACE** permits you to replace files in one directory with files in another directory with the same name. This is useful when you want to have the same update of a file in both directories. The format is the same as the one for **COPY** but you cannot change names. This command takes several switches.

The switch **/a** add new files to the destination instead of replacing existing ones.

The switch **/p** asks for confirmation before making a replacement or addition.

The switch **/r** asks for replacing also read-only files.

The switch **/s** asks for searching also the sub directories of the destination.

The switch **/w** asks to wait for you to insert the source disk before starting looking for files.

The switch **/u** is useful. It asks to replace only those files of the destination that are older than the ones in the source.

There is no way to use **REPLACE** with hidden or system files.

4.10 A SPECIAL TYPE OF COPY

There is a related command that is very useful, especially when copying groups of files. This command is **XCOPY**. The main advantage of **XCOPY** over **COPY** is that **XCOPY** accepts several switches. On the other hand, **XCOPY** cannot concatenate files. The command **XCOPY** behaves as **COPY**. There are differences. One is that **XCOPY** can change the archive bit of the files it copies. It is useful for backing up files. The other is that **XCOPY** accepts the following switches:

/v verifies the copied files;

/s copies from directories and sub directories, creating the sub directories, but not empty ones;

/e used with **s**, creates also empty sub directories;

/d:date copies only those files created or changed on or after the specified date;

/m copies only the files that have the archive bit set. That is, created or changed since last time you used

XCOPY. The archive bit of the copied files is reset; /a copies files like m, but without changing the archive bit; /p prompts for confirmation before creating files; /w displays a message. It waits for the <Enter> key before starting the copy.

As you can see, there are very important possibilities. The chance to verify the copy is important when backing up files. Naturally, it takes more time, but increases safety. The other case is to copy sub directories. This means that the files inside a sub directory are, not only copied, but they go inside a sub directory with the same name. This means you maintain the structure of the disk you are backing up. There are some cases, especially when doing back up, that you wish to keep the structure, even copying empty sub directories.

4.11 VERIFYING THE COPIES

Normally DOS makes the copies without any verification. You have seen that the /v switch permits to ask for verification of the copies. If you wish to verify every copy, whether made with **COPY** or **XCOPY**, you can use the command **VERIFY**. **VERIFY** accepts on to turn it on, or off to turn it off. After verify is turned on, DOS verifies every copy.

4.12 DELETING FILES

Deleting or erasing a file from disk requires the use of the command **DEL**. This command requires the use of a file name. If the file name contains wild cards, more than one file could be deleted. This command only takes a switch. This is /p that this time means to ask for confirmation before deleting each file. If you answer y or Y, the file is deleted. Answer n or N and the file is not deleted. You should be careful when using wild cards with the **DELETE** command. You might end deleting files you wish to keep.

4.13 UNDELETING FILES

We have mentioned the dangers of deleting files. DOS provides a command to try a solution. This command is **UNDELETE**. The format of this command is **UNDELETE path**. The path can be the one for a directory or the one for a file. In the first case, DOS will try to undelete the files of the directory. It presents you with a list of all the deleted files from that directory, one by one. The names have the first letter missing. This is the way DOS deletes files. DOS asks you if you wish to undelete that file. If you answer Y or y, DOS asks you to supply the first letter for the file name. For a single file, you get only that file, with the same procedure.

One word of caution. The undelete operation takes effect only in the disk directory. The directory of the disk has information where the file starts. Every block has information on where the file continues. The

UNDELETE command only restores the name of the file. If you delete a file and immediately you call **UNDELETE**, it is almost sure you will recover it. This might not be true if you have performed any operation with that directory. This is like copying a file onto it, or change a file. When you copy or change a file, DOS uses the first available free space. The space used by the file is now free. The old file might have been corrupted. After you undelete a file, you should check it.

UNDELETE accepts several switches. The switch **/list** asks to list the files available but do not work on them. The switch **/all** recovers all files without prompting for confirmation. All files will has the character **#** as the first character of the name. The switch **/dos** recovers all files listed as deleted by DOS. The switch **/dt** recovers only those files listed as deleted by the deletion tracking system (See **MIRROR**).

4.14 MOVING A FILE

DOS does not have a command for moving a file. Practically any other operating system and shell permits you to move a file. The move operation is to copy a file from one place to another and delete it from the original place. Say you have the file **READ.ME** in directory **UTIL** of drive **C:**. You wish to move it to the directory **DATA** of drive **D:**. The following procedure will do it:

```
COPY C:\UTIL\READ.ME D:\DATA\*.*
DEL C:\UTIL\READ.ME
```

Naturally, you can move more than one file by using wild cards. As in any case of deleting with wild cards, you should be careful.

4.15 RENAMING FILES

Changing the name of a file uses a simple command that does not accept any switches. The command is **RENAME**, or **REN**. You should give the old name of the file you wish to rename followed by the new name. Say, you have a file **READ.ME** in the directory **UTIL** of drive **C:**. You wish to call it **README.1ST**. You say

```
REN C:\UTIL\READ.ME README.1ST.
```

Several points to note. Say, the active directory is not where the file is. You need to give the full path of the file. In any case, you never put the full path in the new name. The file with the new name will be in the same directory it is now.

4.16 COMPARING FILES

There is a command that compare files, this command is **COMP**. The format is COMP filename filename. The comparison is made only when the two files have the same size. Two files of different size cannot be equal. You can use wild cards to compare several files. You could compare whole disks. The command COMP A: B: tells you if all the files in one drive match exactly the files in the other. The drives do not have to be similar. In this way you can compare a 3 1/2 inch diskette with a copy in a 5 1/4 diskette.

COMP accepts several switches.

The switch /d asks to display the differences in decimal, instead of the hexadecimal default.

The switch /a asks to display the differences as characters.

The switch /l asks to display the differences with line numbers instead of the default byte offset.

The switch /n=number asks to compare only the first number of lines, even if the files are different.

The switch /c asks to ignore the case in the comparison; that is, a = A. The default is to compare case sensitive so a is not equal to A.

DOS also has the command **FC** for file comparison. The operation is similar but FC accepts more switches. The meaning of the switches is different from those for COMP.

The switch /a asks for an abbreviated display. Instead of displaying all the lines that are different, it only displays the first and last line of a block.

The switch /c ignores the case of the letters.

The switch /l compares the files in ASCII mode.

The switch /lbuffer sets the number of lines of the internal line buffer; the default is 100. If the files are larger than what fits the buffer, FC does not compare them.

The switch /n add line numbers.

The switch /t does not expand tabs. The default is to expand tabs to 8 spaces.

The switch /w compresses white spaces (tab and space). If a file contains several consecutive spaces, FC considers them as a single space. FC always ignores spaces at the beginning and at the end of the line.

The switch /number tells the number of consecutive lines that must match for FC to consider the files synchronized. The default is 2.

The switch /b asks to compare the files in binary mode.

4.17 FINDING A STRING IN A TEXT FILE

Finding a string in a text file can be useful when you want to find a file with some information. It is useful also when you wish to find some information in a file. Say you have a file with names, addresses and telephone numbers. You wish to find the name and address of somebody. You only recall the city where he lives. DOS has the command **FIND** that can be used for this purpose. The structure is FIND "string" filename. The command FIND will search the file and display all the lines containing the specified string. The command FIND accepts several switches.

The switch /v displays all lines not containing the string.

The switch /c displays only the count of the lines containing the string.

The switch /n precedes the line with the line number.

The switch /i tells the search is not to be case sensitive.

Something important. FIND expects the string to be in the same line. Say, you search for a string formed by two words, like "black dog". FIND does not find occurrences where black is in one line and dog in the next.

Chapter 5.- Working with Disks

5.1 TYPES OF DISKS

You need to understand the organization of the external space of the computer. This is the storage devices you use in your computer. You interact with the computer through its external space. You ask the computer to run a program. You need to tell it where is the program. You need to know where you put it. The basic part to study are the disks. They are the area where you store programs for the computer to run. You also put there the files the computer manipulates.

Consider first the removable disks. You call them **floppy disks** because they are not rigid. There are several types of disks. Many types existed in the past. Today there are two types.

One is 5 1/4 inch in diameter. This type of disk rotates in a flexible paper enclosure. There are two qualities of magnetic material. One receives the name of double density. A diskette can hold 360 kilobytes. The other receives the name of high density. A diskette can hold 1.2 megabytes. This one is the most inexpensive way to store files.

The other type of diskette is 3 1/2 inch in diameter. It rotates in a rigid plastic enclosure. The two qualities of magnetic material mentioned above produce diskettes with 720 kilobytes and 1.4 megabytes.

The fixed disks receive the name of **hard disks**. They cannot be taken from the computer. Thus, the name of fixed. The magnetic material covers a solid metallic disk. Thus, the name of hard disk. Hard disks have much larger capacity than floppy disks. Hard disks work much faster than floppy disks. Working with floppy disks and hard disks is very similar.

There is another type of disks that are not physical. They exist only in the memory of the computer. They are virtual disks. They are also **ram disks**. They are temporary storage. Their advantage is their speed.

5.2 DISK STORAGE

A disk has some concentric **tracks**. Each track has many **sectors**. A group of sectors forms one **allocation unit**. A file uses several allocation units. The last allocation unit might have some free space. Here you have an example. Say a disk has 1000 allocation units. Each allocation unit has 1024 bytes. You put 1000 files with 1 byte each. The disk is full. You also can put 10 files with 102400 bytes each. The disk is equally full.

Original diskettes had a certain capacity for storage. As the technology advanced, the heads are smaller and the magnetic materials are better. The capacity of the diskette increased. This new capacity received the name of **double density**. Disk capacity is the number of bytes the disk can store. This is not the total size of the files. A 5 1/4 inch, double density diskette can store for 360 Kilobyte. It has 354 allocation

units. Each allocation unit can store 1024 bytes. A 5 1/4 inch, **high density** diskette has a nominal capacity of 1.2 Meg. It has 2371 allocation units. Each unit has capacity for 512 bytes. A 3 1/2 inch, double density diskette has a nominal capacity of 720 Kilobyte. It has 713 allocation units. Each allocation unit stores 1024 bytes. A 3 1/2 inch, high density diskette has 2847 allocation units. Each allocation unit stores 512 bytes.

5.3 FORMATTING A DISK

Before you can use a physical disk, you need to format it. This is not so for a virtual disk. Formatting is the process of storing some information on the magnetic surface of the disk. The process of formatting divides the disk in tracks. They are concentric lines where the computer stores data. There are several sectors on each track. The computer writes information on the sectors. The number of sectors and tracks depends on the size of the disk. Formatting a disk destroys all the data in the disk. Formatting a disk is a complex process.

The command **FORMAT** permits you to format a disk. The basic command is **FORMAT B:**. This asks DOS to format the diskette in drive B. DOS takes the type of disk from the type of drive. It first asks you to put a new disk in drive B and press Enter when ready. At the end of the formatting procedure, DOS gives you a report on the process. It tells you how many bytes the disk has. It also tells how many are bad, how many are useful. It tells you how many allocation units and their size. It also gives you the volume ID number. Then it asks you for a volume label. The volume label can have up to 11 characters. Finally, DOS asks you if you wish to format another diskette.

The **FORMAT** command accepts many switches. You can avoid having to enter the label at the end by specifying the switch **/v:label**. You can specify the size of the diskette with the switch **/f:size**. The problem with this procedure is that the disks you format this way might not be readable in other computers. You can use the switch **/q** to perform a quick format. That is, only erase the FAT and the root directory without scanning the surface for errors. You can execute the command with the switch **/u** for unconditional format; that is, all the data is destroyed. Finally, you can format a disk with the switch **/s**. In this case you get a disk that is bootable. It has the three system files **IO.SYS**, **MSDOS.SYS** and **COMMAND.COM**. The first two are hidden files. There are other switches that specify the number of tracks and sectors and other characteristics of the disk.

One related command is **SYS**. This command copies the three system files to a formatted diskette.

5.4 LABELING A DISK

You might not have put a label on a disk when formatting it. You might want to change the existing label. You have the command **LABEL** for this purpose. The format is **LABEL drv: label**. You can see the label of a disk when seeing the directory. You can use the command **VOL** with the letter of the drive.

5.5 RECOVERING A DISK

After formatting a diskette, it is possible to recover the information with the command **UNFORMAT**. This command will restore the files to the disk. This is only possible if the /u switch was not used. This command uses several switches.

The switch /j permits to verify if the disk can be unformatted. That is, if the **MIRROR** file exists. The /u switch unformats the disk without the file created by mirror. The switch /l displays the name of every file that **UNFORMAT** finds. The switch /test displays information on how **UNFORMAT** would recreate disk information. The switch /partn uses the file **PARTNSAV.FIL**, created by **MIRROR**, to restore the partition table of a hard disk.

Another related command is **MIRROR** that creates a copy of the file allocation table and to install deletion-tracking. **MIRROR** can use several switches and the drive to protect.

The switch /drv saves only the latest information for the drive. The switch /partn saves information of the partition table of a hard disk. The switch /tdrv-# loads deletion tracking for drive drv with # entries. The switch /u unloads deletion tracking.

Deletion tracking is a procedure for making a list of the files deleted from the disk. These files can be recovered as explained above. The recovery procedures are dubious, to say the least. It is better to be extremely careful when deleting files. Backing up your work is a safer procedure.

5.6 COPYING A DISK

A diskette can be copied in two forms, file by file or the whole disk. To copy file by file you use the command

```
COPY A:\*.* B:\*.*,
```

for example. To copy the whole disk you use the command **DISKCOPY**. The basic requirement of this command is that the two diskettes are equal. They must have the same size and capacity. The format of the command is **DISKCOPY A: B:**. This copies the disk in drive A: to the disk in drive B:. The target diskette does not have to be empty, or even formatted.

If you do not have two equal drives, you can copy with a single drive. For example, **DISKCOPY A: A:** will copy a disk in drive A to another disk in drive A. DOS will prompt you to change disks.

The command **DISKCOPY** accepts two switches.
The switch **/1** copies only the first side of the disk.
The switch **/v** verifies the copy.

5.7 COMPARING DISKS

The command **DISKCOMP** compares two floppy disks. It does not work with hard disks. This command requires that the two disks are of the same type. That is, the two disks must be of the same size and capacity. The comparison is made track by track. The format of this command is **DISKCOMP drv: drv:**. There are two switches.

The switch **/1** compares only the first side of the disk.
The switch **/8** compares only the first 8 sectors of each track, even if the disks have more.

This switch permits to compare two disks of different number of sectors per track. They still need to have the same number of tracks.

5.8 ASSIGNING A DRIVE LETTER TO A DIRECTORY

Under some circumstances it might be desirable to have a drive letter representing a directory. The command **SUBST**, substitute, permits you to do just this. The format of this command is **SUBST drv: path**. This assigns the letter **drv** to the directory specified by the path. From there on you treat the directory as it were a drive. Naturally, you can use this command to assign another letter to a drive. The command **SUBST Z: B:** assigns the letter **Z** to the drive normally called **B**. Note that you still can call **B:** to drive **B:**. The use of this command is preferred over the use of **ASSIGN**. To cancel an association use **SUBST /d**. When using the **ASSIGN** command you type **ASSIGN** to cancel the associations. The **ASSIGN** command permits you to get a list of associations by typing **ASSIGN/STATUS**.

5.9 CHECKING A DISK

DOS provides a command for checking the disks. The command that you use for this purpose is **CHKDSK**. This command is complex and should be used with caution. The format of the command is **CHKDSK drv:files**. In this form, the command checks the disk and the files, if they are specified. If no files are specified, it checks all the files. Then, it gives you a report on how much space is used. It also reports on the structure of the disk and the use of memory. The command **CHKDSK** accepts several switches.

The switch **/v** simply shows the name of the files as they are checked.

The switch /f tells DOS to fix any problem found with the files. CHKDSK asks you for permission to repair the files. You have the option of saying Y or y, to permit the repair. You can say N or n to prevent it.

Some times CHKDSK reports **lost clusters**. These are blocks of disk space that for some error, are assigned to a file that does not exist. You can recover them, to be used by other files. If you allow CHKDSK to recover the lost clusters, it creates files. It uses the name FILE0000.CHK, for the first cluster, FILE0001.CHK for the second, and so on. You can examine them and see if they are of any use. You can delete them to recover the space they use. In any case, when you find lost clusters it means one of two cases. You turned the computer OFF while running a program; or one of the programs that you use has an error.

You can redirect the report produced by CHKDSK to a file. This permits you to analyze it later, or save it as reference. You cannot use the /f switch when you redirect the output.

5.10 FRAGMENTATION

Fragmentation relates to how DOS stores files. We mention the allocation units. DOS maintains a record of the units used and those free. When you store a file, DOS finds the first available allocation unit and starts storing the file. If the file does not fit, it looks for the next free allocation unit and continues the process. At the beginning, when the disk is empty, the files use allocation units that are contiguous. As you move files around, allocation units become free in the middle of the disk. They are used. Files are changed and do not fit the previous space. The tail end of the file goes to the first free unit. In this way, very soon the files get scrambled. It is said that the files are fragmented. When the situation is pretty bad, you notice the disk becomes slower.

The solution to fragmentation is to empty the disk and restore the files. This is not too difficult if you have a large hard disk with several partitions. You can take one partition at a time and move all the directories to another that has enough space. If that is not possible, you can move some directories to one partition and others to another. In extreme cases, you can move some of the directories to floppy disks. After this, you move the directories back to their original partition. This assures all the files are in consecutive clusters. This represents some work. You will notice your hard disk speeds up. This procedure can be simplified by using batch files.

5.11 PREPARING A HARD DISK

The hard disk is one important additions to your computer. If you do not have a hard disk, you should consider it the first improvement you can make. The cost of a hard disk is a good percentage of the cost of the computer. The increase in efficiency with the hard disk justifies the investment. You need to prepare the floppy disks before they can be used. You also need to prepare the hard disk. Floppy disks

are simply formatted. The hard disk is much larger than a floppy disk and requires more preparation. DOS provides a command to prepare a hard disk. This is **FDISK**. This command is a complex program, as all external commands. In this case, you are presented with several menus. You perform the functions by making selections from the menus. One word of caution. When you use **FDISK** you will wipe out all the programs and files in your disk. You can work in one partition without damaging the others. You can see the partitions and logical drives without damaging anything. It will clutter this book very much to explain the different menus and how to work with them. It is much better to explain what functions you must perform and how you select between the different cases.

The first characteristic of the hard disk is its size. You can make the hard disk behave like one disk or like several. The advantage of having a single disk is that is simple to work with it. You do not need to be concerned with changing disks. On the other hand, the directories become larger. You need to have more directories. The operation becomes slower because DOS has to search large directories. There is a limit of 252 entries in the root directory of any disk. When you have 100 or so entries, the operation is so slow that you want to subdivide it.

Another point is that you normally use the computer for several purposes. The computer might be used by several persons. When you have a large hard disk, say more than 20 Megabyte, you should partition it in several logical disks. One partition for every type of operation or person using the computer. You select one of them, normally C, as the boot disk, and the others as secondary disks. If you do not have any preference, make the partitions equal size. Be sure that the partitions are not so small as to be useless. They must be large enough to contain the programs you plan to use. A good guide is not to have partitions smaller than 10 megabyte. The main partition, drive C, should have room for all DOS programs, room for **WINDOWS** if you use it, and room for general purpose programs. The other partitions, for special purposes or persons, should be of adequate size.

FDISK permits you to partition the hard disk any way you want. Once you have partitioned your hard disk with **FDISK**, you need to **FORMAT** each of the partitions. The main partition, drive C, should be formatted as a system disk, so you can boot from it. The others are formatted as normal disks.

One point of procedure. If you are preparing a new hard disk or you are changing an existing disk, you need to boot the computer from a floppy disk in drive A. Recall that you do not have the hard disk once you call **FDISK**. Further more, recall that **FDISK** destroys everything you have in the hard disk. Before starting a change of your hard disk, you need to backup all the files of interest. We must say a few words about the backup procedure.

5.12 BACK UP

The procedure called backup is to copy some files. This is done to recover them if they get destroyed. This is also useful when you make changes you later consider bad. This is also useful when you will wipe out your disk for reformatting, partitioning, or defragmentation. You do not need to copy

everything. You have the originals for most of the programs. The important files to backup are those you have created. Consider you use WINDOWS. The same applies to any other commercial program you use. Get the directory of WINDOWS and look at the date of the files. You could use the /o:d to order the files by date. Copy any file with a date after you installed WINDOWS in your computer. You will need those files. They represent what you have done and the way you have customized WINDOWS. If you wish to change what you have done, you can destroy those files and start over. Do the same with all the other directories. It is a good practice to do this from time to time to protect your computer in of a failure.

There is a very simple way to do this automatically. Use the XCOPY command with the /d:date switch to copy all the files changed after that date. Use the switch /s to examine also the sub directories. Direct the copy to the drive you wish to use. Repeat for all the directories where you have files of interest. Consider reinstalling the programs that do not have to be configured. You can generate a batch file to perform these operations with a single command. Save the batch file for later use.

DOS provides a command to backup your hard disk. This is **BACKUP**. This command is used as **BACKUP SOURCE DESTINATION**. The source is what you wish to backup. The destination is the drive where you wish to backup. This command only copies to the root of the drive. If the diskette has some files stored in the root, they are destroyed. If there are directories, they are not touched. The command uses only the space available in the root of the diskette. This command produces two files on the root of the destination diskette. One is BACKUP.000 with the files all packed together. The files are not compressed. The other file is CONTROL.000. If a second disk is needed, the extensions are 001. The same with the next files. Note that the files are not compressed. The CONTROL file is very small.

You can compute the number of disks you need by adding the size of the programs and dividing by the capacity of the disk. You recall what was said about allocation units and space left empty and the end of the files. That does not apply here because DOS puts the files one after another without gaps. The disks are full.

This command accepts several switches:

/s tells to backup the source and all its sub directories.

The switch /m tells to backup only those files with the archive bit on and turn it off.

The switch /a tells to add the backup files to the existing files, instead of destroying them.

The switch /f:size asks to format the disk to the given size. Recall the comment on the format command.

The switch /f without size tells to format the files to the standard size.

The switch /d:date backs up only files created after that date.

The switch /t:time backs up the files created today after that time.

The switch /l asks to create a log of the files as they are backed up. You can add a colon and a path and file name.

5.13 RESTORE

A related command is **RESTORE**. It is used to restore files that were backed up with the **BACKUP** command. The format is **RESTORE DRIVE DESTINATION**. The drive tells where the backup disk is placed. The destination tells where you want the files. If you say a path with the destination, it has to be the same as when the files were backed up. You can tell the name of a file. This restores a single file. **RESTORE** accepts several switches.

The switch **/s** tells to restore all sub directories.

The switch **/p** ask for permission to restore read-only files.

The switch **/b:date** restores only those files changed on or before that date.

The switch **/a:date** restores all files changed on or after that date.

The switch **/e:time** restores those files created earlier than that time.

The switch **/l:time** restores all files changed on or later than that time. Note the switch uses the letter **l**.

The switch **/m** restores those files changed since last backup.

The switch **/n** restores those files that no longer exist in the destination.

The switch **/d** displays a list of the files that match the destination. It does not restore any.

Chapter 6.- Customizing Your System

6.1 CUSTOMIZING

It was mentioned in the Introduction that no two IBM-type computers are equal. Each has its peculiarities and its differences. The size of the memory is different. The number and type of diskette are different. The location of the diskettes might be different. You might have a mouse, tape, CDROM, or other board. The programs you run are different. This is why, you must configure your computer. When you got your computer, the system integrator probably did some configuration, to make it run. For your computer to work the way you want, you need to change the configuration. You need to customize your system.

The idea of customizing the computer is to tell DOS what you have and how you want to use your resources. Part of the configuration is done by the bootup procedure, when the system is examined and tested. You might have noticed that the lights on all the devices turn on for a moment. At the end of this procedure, you have a table on the screen telling what you have.

There are many values that cannot be configured automatically. For example, there is no way for the computer to know which type of programs you run. It cannot know how you want the screen, which colors, which prompt. You have to tell the computer what you want. All this is done during the customizing of the computer.

When the computer boots, it reads two files. The configuration and customizing are done with these two files. They should be in the root directory of the disk you use to boot the computer. These files are the **CONFIG.SYS** and the **AUTOEXEC.BAT**.

6.2 THE CONFIG.SYS FILE

The CONFIG.SYS file is the first one read during the start up of the computer. This file is used to tell the computer how you want to use your resources. There are a few commands you can use in the CONFIG.SYS file. Most of them can only be used there. They are:

BREAK, specifies when DOS checks for Control-C and Break keys. DOS normally checks these keys only when it reads the keyboard or writes to the screen. You can make DOS check them more often by turning **BREAK ON**. You should be aware that many programs change the setting of **BREAK**. Only a few programmers have the courtesy of putting it back the way they found it. This command is simple. You say **BREAK ON**, to turn it on. You say **BREAK OFF**, to turn it off. You say **BREAK** to find out if it is on or off.

BUFFERS, sets the amount of memory that DOS reserves for transfer of data to and from the disks. If the number of buffers is too low, the operation is slow. If the number of buffers is large, less memory is available for programs. In any case, each buffer is only 512 bytes. You use it by saying **BUFFERS=20**.

COUNTRY, sets the language and other parameters used by DOS.

DEVICE, loads the device drivers. These are small programs that handle the different devices you have in your system, that are not loaded automatically. You will find its use when we talk about drivers.

DEVICEHIGH, similar to **DEVICE**, but loads the drivers in high memory, that memory above 1 megabyte, if you have it.

DOS, sets the area of memory where DOS will be located. You can set it in conventional or upper memory. This command is used by typing **DOS=HIGH**, or **DOS=LOW**. This command also accepts **UMB** and **NOUMB** to ask DOS to maintain or not a link between upper and conventional memory. You need **DOS=UMB** to load programs or drivers in upper memory.

DRIVPARM, sets the characteristics of disk drives that are not recognized by the system.

FCBC, sets the number of file control blocks that DOS can have open simultaneously. Very few programs use now the control blocks. You can set it to 1 by typing **FCBC=1**.

FILES, sets the maximum number of files that DOS can have open concurrently. This is another parameter that depends on the type of programs you run. You will need a large number, about 35, if you run **WINDOWS**, database, spreadsheet, or similar programs. For simple programs, the default is usually good. You use it by typing **FILES=35**.

INSTALL, is another way to load device drivers or programs that stay in memory. This is used for those drivers that are programs with extension **EXE** or **COM**. You use **DEVICE** to load those drivers with extension **SYS**. The difference between **INSTALL** and running the program is in the environment. **INSTALL** does not pass an environment to the program. The program cannot use environment variables.

LASTDRIVE, tells DOS how many drive letters you need. The default is one more than the physical drives you have. You need more if you wish to assign letters to drives. You also need more if you wish to have a **RAMDISK**. Another reason is when you wish to operate a drive in different ways. You use it as **LASTDRIVE=H**.

REM, is used to include comments in the file.

SHELL, shows DOS which command interpreter is being used. It is also used to tell DOS that **COMMAND.COM** is not in the root of the boot disk. The typical use is

SHELL=C:\COMMAND.COM.

SHELL does not accept switches. You can pass switches to the command interpreter. This command can be used to tell COMMAND.COM to increase the size of the environment space. The use is

```
SHELL=C:\COMMAND.COM /e:512 /p.
```

This command sets the space at 512 bytes and it is permanent.

STACKS, sets the amount of memory DOS reserves to process hardware interrupts. Its use is STACKS=10,512. It asks for 10 stacks of 512 bytes each.

SWITCHES, specifies the use of conventional keyboard functions when you have an enhanced keyboard.

A typical CONFIG.SYS is as follows:

```
LASTDRIVE=J
DEVICE=C:\WINDOWS\HIMEM.SYS
DOS=HIGH,UMB
DEVICE=C:\WINDOWS\EMM386.EXE 512 RAM
FILES=35
BUFFERS=30,8
STACKS=9,256
FCBS=1
DEVICEHIGH=C:\SYSTEM\ANSI.SYS
rem A Ramdisk for programs with 4 Meg in extended memory
DEVICEHIGH=C:\SYSTEM\RAMDRIVE.SYS 4096 /E
rem drive B: as 360 K
DEVICEHIGH=C:\SYSTEM\DRIVER.SYS /D:01 /F:00
rem Drive A: as 720 K
DEVICEHIGH=C:\SYSTEM\DRIVER.SYS /D:00 /F:02
BREAK=ON
Rem CDROM Driver
DEVICEHIGH=C:\SYSTEM\MTMCDD.SYS /D:MSCD001
SHELL=C:\COMMAND.COM C:\ /E:1024 /p
```

The devices are loaded in high memory. Most of DOS is also loaded high. This CONFIG.SYS is for a 486 computer with 16 Mb of RAM, two 120 Mb hard disks, 1.2 Mb and 1.44 Mb floppies, tape drive and CDROM drive. Note the use of SHELL to set 1024 bytes for environment. Note also the 4 Mb RAMDRIVE. You will study the RAMDRIVE with the study optimization of the operation of your computer.

6.3 INSTALLING DEVICE DRIVES

One of the important functions of the CONFIG.SYS is to install the drivers you need. Typical devices that need to be loaded is the mouse driver and the ANSI.SYS. The mouse driver is loaded this way only if it is a system program. Note in the example above how the driver for the CDROM is loaded this way. Consider ANSI.SYS is in the directory SYSTEM of drive C. You load ANSI.SYS with the line

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

The same with all other drivers you need to install. ANSI.SYS is a very versatile system to increase the flexibility of using the display. You will study in detail later in the Chapter.

6.4 USING A DRIVE IN TWO MODES

It was mentioned before that the format command permits you to format a double density disk in a high density drive with the /f switch. It also was mentioned that the disk so formatted might not be read by any other computer but yours. The way to get disks that can be read in any computer is by assigning another letter to the same drive. You also assign the characteristics of a double density drive. You use **DRIVER.SYS** for this purpose. See example above. DRIVER.SYS is loaded with the DEVICE command. When you use DRIVER.SYS you need to specify several switches.

The switch /d:number gives the number of the disk you wish to use. Disk A is 0, disk B is 1, through 127 for the floppy disks.

The switch /c tells the disk can detect when the door is open.

The switch /f:factor shows the type of drive you wish to emulate. A 0 tells 360 K, a 1 tells 1.2 Mb, a 2 tells 720 Kb, a 7 tells 1.44 Mb, and a 9 tells 2.88 Mb. If you use the /f switch you do not need to specify the /h:heads, /s:sectors and /t:tracks switches.

Say you have a 5 1/4 inch, high density drive in A and you wish to format double density diskettes. You add to your CONFIG.SYS file a line reading,

```
DEVICE=C:\SYSTEM\DRIVER.SYS /d:0 /c /f:0.
```

Note that this is only needed to format double density diskettes. You can read them with your drive without any problem.

6.5 THE AUTOEXEC.BAT FILE

The second file that is read and executed during the start up procedure is the **AUTOEXEC.BAT** file. This is a normal batch file and only its name makes it special. This file is used to set up the search path

for the computer, the environment variables, configuring ports. This file is also used to load memory resident programs and whatever other operation you wish to perform every time at start up. Note that all the operations performed by the AUTOEXEC.BAT file can be performed later. This is the big difference between the CONFIG.SYS and the AUTOEXEC.BAT files. The CONFIG.SYS file can only be run during the start up procedure of the computer. None of the commands you use in this file can be used from the keyboard. The reason for using the AUTOEXEC.BAT file is only convenience.

As an example, consider the case of a RAM disk. This is an auxiliary piece of memory that is made to behave like a disk drive. The advantage is its speed. If you have enough memory, you can create a RAM disk and copy there all the programs you use all the time. This is a typical operation for the AUTOEXEC.BAT.

Many programs, including DOS, use what is called environment variables. These variables can be set later, but not after you run a program. A good use of the AUTOEXEC.BAT file is to set all the environment variables for the programs you use all the time. These variables take so little space that is good to set them for all the programs you use.

The AUTOEXEC.BAT is a normal batch file. You must spend some time studying batch files. A typical AUTOEXEC.BAT file, for the same computer described above is

```
@echo off
SET COMSPEC=C:\COMMAND.COM
PATH G:\;C:\DOS;C:\WINDOWS
SET APPEND=C:\DOS
SET INCLUDE=C:\INIT
SET LIB=C:\INIT
SET TEMP=G:\
SET TMP=G:\
SET MASM=/B63 /E /N /P /T /W2 /Z
SET LINK=/CP:1 /DO
SET INIT=C:\INIT
SET DIRCMD=/o
SET CWK=D:\LOGS
SET PCOPTION=/iC:\INIT/ml/e
SET LIBRARY=D:\DO\POWERC
COPY D:\DO\RAM\*.* G:\*.* > NUL
rem Mouse Driver
LOADHIGH C:\SYSTEM\IMOUSE.COM /B3 /U
rem CDROM Driver
LOADHIGH C:\SYSTEM\MSCDEX.EXE /D:MSCD001 /E /L:H
rem ALARM System
LOADHIGH C:\SYSTEM\WHENRES.EXE
C:\SYSTEM\WHEN INIT
```

```

rem Check CMOS RAM
CD \SYSTEM CHEKCMOS
CD \
echo ADJUSTING THE CLOCK
echo Please wait
C:\SYSTEM\CWK 2A > NUL
PROMPT $e[37;40m$e$P$g
G:\LOGON.EXE
cls
echo WIN = WINDOWS
echo MEN = MENOLLY

```

The PATH and the requirements for loading drivers have already been mentioned. Only drivers that are programs with extension COM or EXE can be loaded from the AUTOEXEC.BAT. Those with extension SYS must be loaded from the CONFIG.SYS file. Other parts of this example will be studied later in this Chapter.

6.6 INTRODUCTION TO BATCH FILES

Any command you use from the keyboard you can use in a batch file. There also exist commands that are only used inside batch files. A typical command of this type is **ECHO**. This command permits to turn the echo of the commands on or off. Turning the echo off avoids cluttering your screen with the commands from a batch file. The ECHO command has five forms: ECHO ON, turns the echo on. ECHO OFF, turns the echo off. ECHO, displays if the echo is on or off. You use @ECHO, either on or off, to avoid echoing the command where you turn the echo on or off. Finally, you use ECHO message to display the message on the screen.

Another command you use only in batch files is **CALL**. This command is used to execute a batch file from another. Say you are executing the batch file AUTOEXEC.BAT. You need to perform other operations you have in the batch file DOIT.BAT. You have two chooses. You can simple add a line that says DOIT.BAT. You can put a line CALL DOIT.BAT. In both cases the batch file DOIT.BAT is executed. In the first case, when the batch file DOIT.BAT ends execution, control returns to DOS. In the later case, when the file DOIT.BAT ends execution, controls return to the calling file, which continues execution.

You can use the command **FOR** both, in batch files and from the command line. The format of this command is

```
FOR %%A IN (.....) DO COMMAND %%A.
```

The idea is there is a list between the parenthesis, for example a list of files. This command takes the

elements of this list one at a time and replaces each %%A for this value. Then executes the command after the DO. Say you want to type three files. The command that types all three, one after the other, is

```
FOR %%S IN (A.DOC B.DOC C.DOC) DO TYPE %%S.
```

Note that the DO and the spaces are required. The only element that is case sensitive is the name of the dummy variable. %%S is not the same as %%s. If you use this command from the keyboard, you replace the double percent sign by a single one.

The command **GOTO** permits you to change the flow of control of a batch file. The GOTO requires a label to mark the place where you want to go.

You define a **LABEL** with any word that starts with a colon. The colon is not used in the GOTO.

The command **IF** permits you to include conditions in a batch file. A command is executed only if the condition is true. There are only three conditions: **errorlevel**, **equality**, and **exist**. Any of them can be reversed with **NOT**. The error level is a value that every program returns to DOS when it finishes execution. A value of zero means normal end. Any other number means something abnormal. The format used to test abnormal end is

```
IF ERRORLEVEL 1 COMMAND.
```

To execute the command when the end is normal you use

```
IF NOT ERRORLEVEL 1 COMMAND.
```

Note that this means error level 1 or above.

Equality is tested with the symbol ==. For example, to execute a command when two strings are equal you say,

```
IF string1==string2 COMMAND.
```

To execute it is they are not equal,

```
IF NOT string1==string2 COMMAND.
```

You use the exist condition to test if a file exists or not. To execute a command if a file exists you say

```
IF EXIST filename COMMAND.
```

To execute it is the file does not exist,

IF NOT EXIST filename COMMAND.

You use the command **PAUSE** to halt the execution of the batch file until a key is pressed. This is useful when you need the user to change disks, for example. You put a message "Do this and that and", then you put PAUSE. The computer adds the "Press any key to continue". Another use of the pause is when you wish to give a chance of interrupting operation. You put a message "Press Control-C to stop, otherwise", and pause. Pressing Control-C the user ends the batch file. The Break can be used for this purpose.

The command **REM** has been mentioned before. It permits to add comments to the batch file, explaining something.

The command **SHIFT** has to do with a very useful capability of batch files. You can develop a general batch file that is used for doing something. Say, you develop a batch file to compress any file. You tell the batch file which program you want to compress by passing parameters. Parameters are values you write after the name of the batch file, or the program.

We must elaborate on this topic. The batch file for compressing any file, called ZIP.BAT, can be as follows:

```
@Echo off
PKZIP -a e:\%%1 %2\*.*
```

This command will use PKZIP, produced by PKWARE. It compresses whatever is in the directory specified by the second parameter. The data goes into a zip file with a name given by the first parameter. The compressed file always goes to drive e. You call this back file by saying

```
ZIP MYFILES D:\DATA\FILES.
```

All the files in the sub directory FILES, will be zipped into a file called MYFILES in drive e:. Now the SHIFT command. A batch file can use only nine parameters %1 to %9. %0 is the name of the batch file. If you need more parameters you SHIFT them to the left and %1 disappears, %2 becomes %1, and so on. The next parameter in the list becomes %9.

6.7 THE ANSI.SYS SYSTEM

Installing the **ANSI.SYS** in your computer gives a large flexibility in the operation of your screen. There are many programs that require you have ANSI.SYS installed before they can run. In any case, it does not do any damage to have ANSI.SYS installed. In this way you can use its facilities if you wish. You

install ANSI.SYS as any device, from your CONFIG.SYS file. Note that this file is only run when you boot the computer. You need to boot your computer for the changes to take effect. The line you add to the CONFIG.SYS file is

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

It is assumed that the ANSI.SYS file is in the directory SYSTEM of drive C. You can use two switches.

The switch /x remaps the extended keys independently in enhanced keyboards. For example, the enhanced keyboard has two Home keys. They can be remapped independently when you specify the /x switch.

The switch /k ignores the extended keys in enhanced keyboards.

You access the ANSI.SYS with the Escape sequences. These sequences start with the Esc key and the square bracket. You can use the following Escape sequences:

ESC[line;columnH, moves the cursor to the line and column;

ESC[line;columnf, works the same as the previous sequence;

ESC[numberA, moves the cursor up number of lines;

ESC[numberB, moves the cursor down number of lines;

ESC[numberC, moves the cursor right number of places;

ESC[numberD, moves the cursor left number of places;

ESC[s, saves the cursor position;

ESC[u, restores the cursor position;

ESC[2J, erases the display and puts the cursor home;

ESC[K, erases the line;

ESC[number;number;numberm, sets the text attributes, the foreground and background, as explained below;

ESC[numberh, changes the video mode of the screen;

ESC[numberl, resets the video mode, the last character is lower case letter l;

ESC[key;stringp, changes the assignment of key to string.

The escape sequence **m** has three values. The first one is the attribute. The second is the foreground color. The third is the background color. You use the following values for text attributes:

0 all attributes **off**;

1 **bold** on;

4 **underscore** (monochrome only);

5 **blink** on;

7 **reverse** video on;

8 **concealed** on.

When selecting colors, the foreground color are numbers from 30 to 37; the background colors are from 40 to 47, as follows:

30 or 40 **black**;
 31 or 41 **red**;
 32 or 42 **green**;
 33 or 43 **yellow**;
 34 or 44 **blue**;
 35 or 45 **magenta**;
 36 or 46 **cyan**;
 37 or 47 **white**.

The escape sequence `h` can cause trouble. The number you use specifies the video mode you wish to use. Safe ones are:

0 40 x 25 monochrome text;
 1 40 x 25 color text;
 2 80 x 25 monochrome text;
 3 80 x 25 color text;
 4 320 x 200 2-color graphics;
 5 320 x 200 monochrome graphics;
 6 640 x 200 monochrome graphics;
 7 Enables line wrapping.

There are other modes a video adapter can work. Sending a command the video adapter does not support can damage it or the monitor. You better read the manual of the video adapter and of your monitor, before using this escape sequence.

6.8 THE PROMPT

One interesting use of the AUTOEXEC.BAT is to set the prompt you see when DOS waits for a command. You use the command **PROMPT** followed by whatever text you wish to give. There are some symbols that have special meaning. They all start with the \$ sign. These symbols are

\$q puts =
 \$\$ puts \$
 \$t puts the current time
 \$d puts the current date
 \$p puts the current drive and path
 \$v puts the DOS version number
 \$n puts only the current drive

\$g puts >
\$l puts <
\$b puts |
\$_ puts ENTER_LINEFEED
\$e puts Esc
\$h puts backspace.

Imagine you wish to have a prompt like C:\DOS=>. This when you are in the directory DOS of drive C. The same for any active directory. The command is

```
PROMPT $p$q$g.
```

Let us analyze it. The \$p calls for the drive and path. The \$q asks for the equal sign. The \$g asks for the greater than sign.

6.9 ORGANIZING YOUR SCREEN

Imagine you wish to set the colors of your screen. You wish to have bold cyan letters in a red background. You also wish to specify the prompt as above. The command is

```
PROMPT $e[1;36;41m$p$q$g.
```

The \$e puts an Esc, then you have the square bracket. The 1 calls for bold, note the semicolon. The 36 calls for cyan foreground, the semicolon. The 41 calls for red background, the m is the code. The rest is the prompt.

Now, let us talk about characters. Anything you enter after the prompt, including blanks, are copied into the prompt. Imagine you wish the prompt to be like C:\DOS HELLO!! =>. The command to produce this prompt is

```
PROMPT$p HELLO!! $q$g.
```

Note that the prompt can have several lines by using the \$_. Once the ANSI.SYS is installed by the CONFIG.SYS file, you can use it from the command line, from the AUTOEXEC.BAT file, or from any batch file you wish to write. The ANSI.SYS is also very useful when you write programs, to format the screen the way you want.

6.10 CONFIGURING THE PORTS.

Very seldom you need to configure the ports. Normally, the program you use for the ports takes care of

configuring them. This is the normal case of communication programs, printing programs, word processors, and many others. If you run into the problem, usually configuring the display or the keyboard, you have the command **MODE** to perform this function. You can set the display with this command. The format is

```
MODE CON: COLS=C LINES=N.
```

The colon and the values are optional. The same command can be used to set the repeating rate of the keyboard. The format is

```
MODE CON: RATE=r DELAY=d.
```

6.11 THE ENVIRONMENT

The environment is an area of memory that DOS passes to the programs it runs. This area has several variables with a string attached to each. They are used to tell to the program values of interest. For example, programs can find where are other files. It can find where to create temporary files, by using environment variables. In all cases, these are values the program cannot determine by itself. They depend on the wishes of the user. The environment as a whole is not too large. The computer described in the examples above uses only 1 Kb for environment. You can load all the environment variables for the programs you normally use with the AUTOEXEC.BAT. It is logical to load the variables for programs you use only once in a while. You load environment variables with the command **SET**. This command can be used in the AUTOEXEC.BAT file or from the keyboard. You cannot use it after you have run a program. For example, you cannot use it in a batch file to create a variable. You change an environment variable with the same command. You can change a variable after you have run a program, or from inside a program, if the variable exists. The format you use in either case is **SET VARIABLE=STRING**. Note that there is one or more spaces between SET and the name of the variable. Spaces between the name and the = or between the = and the string are taken literally. SET A=MINE is not equal to SET A =MINE, to SET A= MINE, or to SET A = MINE. The first one sets the variable A to the string MINE. The second one sets the variable A space to the same string. The third example changes the value of A space to the string space MINE. The final example changes the value of variable A space to the string space MINE. You erase a variable but typing SET A=. You get a list of all defined variables by typing SET.

6.12 RUNNING PROGRAMS

The example above shows several instances of program run by the AUTOEXEC.BAT file. You might notice a program called CKW that is run to keep the clock accurate. Another program checks the content of the CMOSRAM to be sure it has not changed. In the same way you can run any program you wish. Note also the use of the AUTOEXEC.BAT file to transfer programs from the hard disk to a RAM disk.

More about this later.

6.13 MESSAGES AND DRAWINGS

You can put any messages from the AUTOEXEC.BAT file. One good example is in the file shown above. A message is displayed on the screen before running the program that checks and adjusts the clock. The reason is that this program takes several seconds for the test. Without the message, you do not know what happens. Another example is the message put just before ending. Although the file returns to DOS, it suggests two possible alternates. One is WINDOWS and the other is my shell, MENOLLY. Further more, it tells to call WINDOWS by using WIND instead of the usual WIN. The reason is that WIND logs when the WINDOWS program is run and when it ends.

You could put frames around the messages. These frames are easy to produce using the **extended ASCII** characters, those with code larger than 127. Most text editors and word processors let you enter these codes by pressing the ALT key and typing the three digits of each character with the numeric keypad.

6.14 KEEPING RECORD OF LOGGING

Another use of the AUTOEXEC.BAT file is to keep record of when you turn the computer on. This is a good application of the command **ECHO**, the command **NOW**, and the redirection. You include in your AUTOEXEC.BAT the following lines:

```
ECHO Computer turned on at >> C:\LOG.ASC
NOW >> C:\LOG.ASC
```

The first line above tells you what operation you are logging. The double greater than symbol shows to append this phrase to the file LOG.ASC in drive C. The second line asks for the date and time to be redirected, appended, to the same file.

If you always use a shell or graphics interface, like WINDOWS, you can call it from the batch file. After the call you can log off and park the hard drives (a very good habit). The whole system can be as follows:

```
Echo Computer turned on at >> C:\LOG.ASC
NOW >> C:\LOG.ASC
WIN
Echo Computer turned off at >> C:\LOG.ASC
NOW >> C:\LOG.ASC
PARK
:End
Echo Turn computer off
```

GOTO End

You can put a better message if you wish to make it pretty. The message above runs over and over until you turn the computer off. You could add a PAUSE before the GOTO, if you like.

Chapter 7.- Optimizing Your System

7.1 OPTIMIZATION

You can run your computer as you got it from the dealer. You can run your computer with the customizing as studied in the previous chapter. You can get more performance from your computer if you optimize its operation. Similarly to customizing, optimizing depends very much on what you do with your computer. It also depends on the resources you have. Optimization also depends on what you want. You can optimize the computer so you can run very large programs, at the expense of speed. You can optimize for speed, at the expense of memory. In the same way, you can optimize for many other criteria. So, it is not possible to give procedures to optimize the computer that are good for everybody. You must make the choice by yourself. The next sections give you the analyzes of the different procedures, their advantages, and their side effects.

7.2 THE COMPUTER MEMORY

The unit of memory is the bit. Eight bits taken together form a **byte**. You might have 640 kilobytes of memory. This means 640 times 1024 bytes. A **kilobyte** is 1024 bytes. You might have several megabytes. A **megabyte** is 1024 kilobytes. It is also 1,048,576 bytes. You use **Kb** to represent a kilobyte; **Mb** to represent a megabyte. The memory is the basic space of the computer. The more megabytes you have, the larger the program can be. The memory space belongs to the computer. Only the computer can access the memory. To put something in memory you should ask the computer to do it. You cannot do it.

There are several other points of view. The memory is where the computer works. The computer stores information in the disks. The memory is temporary. It exists only while the computer has power. The disk is permanent. The disks also connect with other computers. You can take a diskette from your computer and put it in another. The memory in IBM-type computers is divided in four parts:

Conventional memory is the memory originally used in the computer. It is formed by the first 640 Kilobytes of memory.

The first expansion of the computer memory was to install one megabyte. A computer working under MS DOS, cannot access the memory above 640 kilobytes. This memory receives the name of **upper or high memory**. You need a **memory manager** to use the upper memory. The second expansion was to put more memory in a card that plugs into one of the expansion slots. This memory is called **expanded memory**. The cards with expanded memory come with a program to manage this memory. The programs cannot access it directly. As memory became cheaper, more was installed in the computers. This received the name of **extended memory**. The computer cannot access this memory directly. This comes from faults in the basic design of the computer and the operating system. Accessing this memory requires a special program. This program lets the computer look at this memory through a window. The computer sees a block at a time. This is the extended memory. The program is called **extended memory manager**.

The conventional memory is the only memory that MS DOS can use for running a program. If you

normally run large programs, you might want to conserve your conventional memory. This is what is called optimizing for memory.

7.3 OPTIMIZING FOR MEMORY

If you wish to optimize for memory you need to know which programs are stored in your memory. DOS provides you with the command **MEM** for this purpose. This command accepts three switches.

The switch /p shows the programs that are loaded in memory.

The switch /d shows the status of the programs loaded in memory, the internal drivers, and other programming information.

The switch /c lists the programs loaded in conventional and upper memory, with its size. It also provides a summary of memory usage.

These switches are used one at a time. Normally, the output of this command is a long list of programs and empty areas. The list is too long to fit in one screen. Use the pipe (|) to send the output to MORE and read it by pages. Another chance is to redirect the output to the printer. Study the output from MEM and see if you really need all the programs that occupy space in memory. These programs are the DOS programs, the drivers, and the memory resident programs. You can do little with the DOS programs. You should remove any driver you really do not need. For example, consider you have a CDROM reader. You do not use it all the time. When you need to read a CD ROM you install the driver and reboot the computer. The way to do this is to put a REM at the beginning of the line that installs those drivers. You remove the REM and reboot to install the driver.

There are many memory resident programs or TSR (terminate and stay resident). Most of them perform useful functions. If you need the memory, you should analyze if you really want them. These programs are normally loaded from the AUTOEXEC.BAT file. Some are installed, or added to your AUTOEXEC.BAT file when you install an application. Remove all those you do not need all the time. Some of them can be loaded only when you need them. You can recognize a memory resident program because it uses what is called a hot key. If you have memory above 640 Kb, you can load many of the drivers in high memory. You use the command DEVICEHIGH or LOADHIGH, depending on the type of driver. Some of the memory resident programs can be loaded in high or upper memory. All this saves space.

If you have upper memory or extended memory, you can load most of DOS in upper memory. The command DOS permits you to do this. The format is DOS=HIGH. This alone will save you 160 Kb of conventional memory! If you have memory above 640 Kb, you need to load a program called HIMEM.SYS. You load it as any device driver. This is the memory manager mentioned above. The line that loads HIMEM.SYS should be above any line where you load a driver with DEVICEHIGH. That is, you need to load HIMEM first.

Recall that extended memory is called the one above 1 Mb. If you have extended memory, you need to load the EMM386. This is a memory manager for the extended memory. You load the EMM386 driver after HIMEM.SYS but before any driver that goes into high memory. At every change, test your memory with MEM to see if you have achieved what you intended. A driver might not load in upper memory when there is memory available. Recall that upper memory is normally broken in pieces. The first piece available might not be large enough for the driver. Note from the output of MEM, the size of the driver. You need to load the driver with the command DEVICEHIGH and the SIZE attribute. The format is

```
DEVICEHIGH SIZE=20E8 C:\SYSTEM\MOUSE.SYS,
```

for example. Note that the size goes in hexadecimal. This forces DOS to load the driver in the first block that is at least of that size. Another technique is to change the order in which you load the drivers. Some times a small driver takes the first place, which happens to be a large block. This prevents other drivers from loading there. Changing the order puts the large driver in that block and the small driver in the next block.

Optimizing for memory, as any optimization, takes time and patience. You need to go step by step. The first time you might not achieve all you want. Try again later. From time to time, check you memory with MEM and see if there is not something you can do. If you have memory above 640 Kb, you can organize your computer to have more that 600 Kb free to run programs. This takes some doing, but it can be done.

7.4 THE COMPUTER SPEED

Most people consider the speed of a computer depends only on the frequency of the clock. Those 25 Mhz, or 33 Mhz, the salesman emphasized so much! This is a factor. This is not the only factor and not the most important. The speed of the computer depends on how the computer has been organized. As important, depends on how it is maintained. There are some very important points to study. Consider the time it takes the computer to load a program, or in general, to find a file. This time depends on the organization of the hard disk. It has little relation with the clock rate of your computer. It has little to do even with the type of computer you have. A 486 with a 33 Mhz clock rate might take longer to load a program than a 286 with 12 Mhz. The access to the disk is controlled by the speed of the disk. The access to a file is controlled by the way the disk is organized and the state of the files.

We mention before the problem of fragmentation. As you use files, they get fragmented. The more fragmented a file is the more time it takes to load it. The reason is simple. What takes time in loading a file is not the loading itself. What takes time is finding the piece you wish to load. This is clear when you use floppy disks. You ask the computer to run a program from floppy, or to read the directory. It takes a long time to start the disk, find the directory and read it. The hard disk is always running but it takes time for the head to move from one track to the next. So, file fragmentation in a disk can be a very important cause for reducing the speed of the computer.

It was mentioned before the influence of the PATH variable on the speed of the computer. If your PATH is very long and includes many directories, the computer must search all of them and this takes time. Another related problem is when you call a program with only its name. DOS permits you to call a program MYPROG.EXE by typing only MYPROG. This requires that DOS searches the whole PATH, first with MYPROG.COM, and then with MYPROG.EXE. If the path is long and complicated, this can take a long time.

Finally, a factor that influences the speed of your computer is the number of files you have in each directory. This is especially true for the directories in the PATH. A very interesting experiment is to create a little batch file. It simply creates a directory and then copies the same file, under consecutive names, into that directory. Make it do it for 200 or 300 times. At the beginning, when the directory is empty, the files are transferred very fast, one after the other. As the directory becomes cluttered with files, it takes longer and longer to create each file. This shows you the point.

7.5 OPTIMIZING FOR SPEED

Optimizing for speed is very similar to the process of optimizing for memory. You need to look at your system and find the bottle necks. That is, the places that are holding the computer. There is a big difference between memory and speed. Once you achieve a certain level of optimization for memory, it remains optimized until you make a change. The speed of the computer degrades continuously as you use it. You need to check the speed of you computer from time to time. You might want to have a special batch file for this purpose. This batch file stores any file in each directory of the computer. Then, it deletes all the files. This batch file should read the time when it starts, and the time at the end of the sequence. That can tell you when your computer needs optimization for speed. Do it soon after you have optimized the computer. Put the value in an echo command of the batch file. You can compare to see if your computer has slowed down. A simple example is as follows:

```
@Echo Off
Now
Copy c:\dummy.fil c:\dos > nul
Copy c:\dummy.fil c:\system > nul
Copy c:\dummy.fil c:\windows > nul
Copy c:\dummy.fil c:\windows\system > nul
..... to other directories in your system
Del d:\dos\dummy.fil
Del c:\system\dummy.fil
Del c:\windows\dummy.fil
Del c:\windows\system\dummy.fil
..... same with the other files you copied
Now
Echo The first time it took 43 seconds.
```

Pause

The file to be copied can be any one in your system. It does not have to be in the root. Do not delete it by mistake. Note that all the copies are redirected to NUL. The idea is to avoid the message "1 File Copied" on the screen. Note also the last echo that shows how long it took the first time you run the batch file. When you notice that this time increases too much, it is time to do something about your speed.

The solution for the problem of fragmentation has already been studied. You need to take all the files from the disk and reload them. Since you have the disk (or partition) empty, it is a good idea to format it. The advantage is that formatting checks and marks any defective sectors you might have. It will recover any lost clusters. You should do this from time to time, depending on how much work you do. A good figure is to do this process every 500 hours of use of the computer. The actual number depends on the kind of work you do. If you do not format the disk, run CHKDSK with the /f switch, to recover any lost clusters you might have.

The solution to the problem of the PATH has already been mentioned, but we need to elaborate more. The first point is to call the programs with the whole path and extension. This avoids any search. DOS goes directly to the directory and loads the program. The only solution to the problem of the PATH is to make it very short. This can be achieved with a RAM disk. Look for the programs you really use. Tally how much disk space they require. One easy way to do this is to copy the files to a single directory. Copy only the files that are needed to run a program. Do not copy documentation or read me files. When you have all the programs you normally use, look at the size of the directory. Do not put this directory in the PATH. You need to create a RAM disk with some more capacity than the size of the directory. If you have extended memory, be generous and leave ample space for the temporary files of DOS and other programs. You create a RAM disk with RAMDRIVE.SYS and the DEVICE command in the CONFIG.SYS file. You load it as follows:

```
DEVICE=C:\SYSTEM\RAMDRIVE.SYS SIZE SECTOR ENTRIES.
```

The sector and entries are optional. The defaults are 512 for the sector and 64 for the entries. The size is in kilobyte. If you have memory above 640 Kb, you might want to use DEVICEHIGH. Note that this does not load the RAM disk in high memory. It loads **RAMDRIVE.SYS** in high memory. You can use two switches with RAMDRIVE.SYS.

If no switch is used, the RAM disk goes into conventional memory.

The switch /a tells to put it in expanded memory.

The switch /e asks to put it in extended memory.

DOS gives the RAM drive the next letter. The message that appears on the screen while the CONFIG.SYS runs, tells you the letter assigned to the RAM disk. You add to your AUTOEXEC.BAT file a line. It copies all the files from the directory you formed with programs, onto the RAM disk. See the example of CONFIG.SYS and AUTOEXEC.BAT given before. The interesting point of using a

RAM disk is the speed. You will notice a marked difference between loading a program from the hard disk or from the RAM disk. Further more, put the RAM disk as the first directory of your path. You can call the programs from any place and your path is not cluttered.

The problem comes when you do not have memory above 640 Kb. You might still want to have a RAM disk, even if you cannot put every program there. Put only the programs you use more often. You need to make a balance between the speed of the programs in the RAM disk and the size of the programs you can run. If you have extended memory, you can put a large RAM disk. Add a line to set the temporary area for DOS into the RAM disk. Say your RAM disk is F. You add to your AUTOEXEC.BAT file a line that says SET TEMP=F:\. The same with any other program that requires a temporary storage area. Typical programs are word processors, text editors, database, spreadsheets, and many others. Some times the instructions ask you to add an environment variable for TEMP, TMP or some name like that. Other times there is a configuration procedure where you enter this information. This speeds up the program very much.

The problem of the directories is important. Throughout this book it has been mentioned that certain programs we need for our computer are either in a directory DOS or SYSTEM. It is a good idea to have at least two directories for DOS programs. When you install DOS it puts all the files in the same directory, even the examples for BASIC and the packing list. You should create another directory where you put all the programs DOS uses while starting. It is called SYSTEM in my system. Note that these programs are never used after starting. The drivers, all of them, should be passed to this directory. You can put here other programs you like to use. Typical are the programs to check the CMOS RAM and maintain the accuracy of the clock, mentioned in the examples.

You need another directory where you put all the other files from DOS that you do not use that often. You can even have it in a floppy disk, if you do not have enough space in your hard disk. You should follow the same procedure with all the other directories. Erase or transfer to other directory all the auxiliary files. In this way the main directory get uncluttered. Most good programs give you the chance of separating the files. For example, Word Perfect gives you the chance to have the documents, the macros, the graphics, the spell checker, each group in a separate directory. It is a very good idea of using this capability. It speeds up the operation of the program. The same idea works with other programs. In general, classify the programs in groups. Make a directory for each program that requires several files. After you have done all this, your computer has gained in efficiency.

There are circumstances where you can still do more. It depends on the type of programs you use. Some programs are continuously going to the hard drive for files and overlays. When you run the programs you normally use more, look at the activity of the hard drive. If the light comes on very often, you might gain speed by adding what is called a cache. **Cache** is a French word that found its way into English. It is used to define a block of memory used as a temporary, buffer, or intermediate storage. The cache needs a program to handle the transfer of information. A cache system is used between the disk and the microprocessor or between the memory and the microprocessor. DOS comes with two programs for this purpose. **FASTOPEN.EXE** and **SMARTDRV.SYS**. FASTOPEN.EXE maintains a list of the files you use more often and where they are on the disk. SMARTDRV.SYS creates and maintains a cache system.

A good way to understand the operation of the cache is with an example. Consider a disk. It is quite slow compared with the computer. The disk has information stored on it. This information is required by the program. The program asks for the information as it needs it. Under normal circumstances, a command goes to the drive. The drive searches for the data, passes it to the computer. Many times, the information is organized close together. The disk works only in blocks called sectors. If the program needs a single value, it takes it and discards the rest. The process is repeated for the next value. Imagine now that you have a cache system. This system is formed by a program and some memory assigned to it. When the computer asks for the first piece of information, the cache program gets and stores the whole sector. The next request might be in the same sector. The cache program passes it to the computer without calling the disk. The larger the memory available to the cache program, the more information it can have ready. The faster the computer works. The cache system works independently of the program you are running. The cache works blindly. This is a drawback. It does not have any idea what the computer will request next. There are many cases where a cache system will actually slow down the system. An intelligent program maintains a cache for itself. It retrieves the information in sectors or files. The advantage you can get from installing a cache system is not clear. It depends on the type of programs you run. It might actually slow down the computer running certain programs. You need to check the results after installing the cache.

Chapter 8.- Final Considerations

8.1 OTHER COMMANDS

There are many more commands that have not been explained. This does not mean they are not important. They did not fit in the analyzes. The commands that have not been mentioned fall into two groups: commands that are useful, and commands that are not too useful. There are some that are actually useless. This is the case of the commands or programs to customize the computer for other languages, if you only use English. The following commands are useful and simple: **CLS**, clear screen; **Date**, displays the current date and permits updating it; **Time**, displays the current time and permits updating it. The following commands are not used that often. You will need to read your manual when you run into an opportunity for using one of them. **Command**, starts another copy of COMMAND.COM; **Ctty**, changes the type of keyboard you use; **Exe2bin**, converts EXE programs into binary (COM) files; **Exit**, returns to a program when you shell to DOS; **Expand**, expands the files in the original DOS disks; **Help**, provides some help nobody understands; **Recover**, recovers bad or unreadable information from a bad disk; **Setver**, tells a program whichever version you wish; **Share**, permits sharing files in a network; **Ver**, displays the real version number of DOS. The following are really programs, auxiliary to DOS. **Debug**, is used for debugging a program you wrote; **Doskey**, is used to edit previous commands; **Dosshell**, provides a text interface to DOS; **Edit**, is a limited text editor; **Edlin**, is a very limited line editor. Use Edit instead; **Graphics**, loads a program to print in graphics mode; **Qbasic**, is a BASIC interpreter. The following commands are used to customize the computer for international use: **Chcp**, change code page; **Country**, sets the language and other characteristics; **Graftabl**, enables DOS to use extended character in the code page; **Keyb**, configure the keyboard for a specific language; **Nlsfunc**, loads country specific information.

8.2 CONTINUED LEARNING

If you got to this place, you have a very good idea of how to use DOS commands from your keyboard. You probably realize that this is a very complicated way to control the computer. It was mentioned before there are other ways. DOS provides a text interface that makes it easier to give commands to DOS. This interface is good but not the best. Microsoft also offers WINDOWS, a graphical interface that makes more natural to control the computer. On the other hand, using either DOSSHELL or WINDOWS is not trivial. You will need to make some effort and learn how to use the interface. You need to understand that the same people who wrote DOS, wrote DOSSHELL and WINDOWS. At the end, the time and aggravation you save by using the interface pays for the time spent learning to use it. Of the two, WINDOWS is by far superior. WINDOWS permits you to perform all operations without ever typing any command line. You will need to fill blanks in some windows, but it is much simpler than using the command line. Now you can use the command line. This does not mean you do not have anything else to learn. It was mentioned before that the computer should be considered as an extension of your brain. The more you learn how to use the computer, the more useful that extension of your brain becomes. I hope that reading this book has expanded that extension.