Repeatable Security Hardening and Monitoring

New features in the Solaris 10 OS make it easier than ever to minimize and harden a system. The Reduced Networking Metacluster install option creates a minimized Solaris OS image, ready for administrators to add functionality and services in direct support of their system's purpose.

Mandatory Access Control and Labeling

If your system requirements include privacy, increased accountability, and reduced risk of security violations, then Solaris Trusted Extensions is for you. A standard part of Solaris, true multi-level security is available for the first time in a commercial-grade operating system that runs all your existing applications and is supported on over 1,200 x64/x86 and SPARC platforms.

- **WEAKNESS AND STRENGHT**

A security weakness in Solaris Trusted Extensions Policy configuration may allow a remote unprivileged user who has authorized or unauthorized access to the X server, to leverage an additional vulnerability which could lead to arbitrary code execution as a local privileged or unprivileged user.

Sun has acknowledged a weakness in Pidgin on Solaris, which can be exploited by malicious people to cause a DoS (Denial of Service).

CONCLUSION

The development of the Solaris OS demonstrates Sun Microsystems' ability to be on the cutting edge of the computing world without losing touch with the current computing environment. Sun regularly releases new versions of Solaris incorporating the latest development in computer technology, yet also included more cross-platform compatibility and incorporating the advances of other systems. The OpenSolaris project is the ultimate display of these twin strengths- Sun has tapped into the creative energy of developers across the world and receives instant feedback about what their audience wants and needs. If all software companies took a lesson from Sun, imagine how exciting and responsive the industry could be.

CHAPTER 4 : MS-DOS

1.0 INTRODUCTION

MS DOS is an acronym that stands for **M**icro**S**oft **D**isk **O**perating **S**ystem. It is often referred to as **DOS**. It is an old operating system for x86-based personal computers, purchased by Microsoft that manages everything on your computer: hardware, memory, files. It is an operating system that existed prior to Windows.
MS-DOS was the most commonly used member of the DOS family of operating systems, and was the main operating system for personal computers during the 1980s up to mid 1990s. It was preceded by M-DOS (also called MIDAS), designed and copyrighted by Microsoft in 1979. MSDOS was written for the Intel 8086 family of microprocessors, particularly the IBM PC and compatibles. It was gradually replaced on consumer desktop computers by operating systems offering a graphical user interface (GUI), in particular by various generations of the Microsoft Windows operating system. MS-DOS developed out of **QDOS** (**Q**uick and **D**irty **O**perating **S**ystem), also known as 86-DOS. DOS, as with any operating system, controls computer activity. It manages operations such as data flow, display, data entry amongst other various elements that make up a system.
The role of DOS is to interpret commands that the user enters via the keyboard.
These commands allow the following tasks to be executed:
- file and folder management
- disk upgrades
- hardware configuration
- memory optimization
- program execution

These commands are typed after the prompt, in the case of MS-DOS (Microsoft DOS, the most well known): the drive letter followed by a backslash, for example: A:\ or C:\. And after them, the enter key. The files that make up DOS involves:
IO.SYS : This is a program to handle input/output to your peripheral devices. It stays in memory when you run applications programs

MSDOS.SYS: This is a program for application programs to use. It contains special subprograms to make many commonly needed operations easy for programmers. COMMAND.COM : This program accepts the commands you enter and runs the right program. CONFIG.SYS: Configures the hardware environment Mouse , Printer, Keyboard , Country codes (time, date, currency),other devices and system commands AUTOEXEC.BAT: Programs/commands to be run at system

start Batch file (automatically executing set of programs/commands) IO.SYS and MSDOS are loaded into the PC memory by a special program called a boot record each time you start up DOS . The command used to initialize new disks with DOS,FORMAT/S puts this on the disk along with IO.SYS and MSDOS.SYS

2.0 HISTORY

MS-DOS (Microsoft Disk Operating System) is a single-user, single-tasking computer operating system that uses a command line interface. In spite of its very small size and relative simplicity, it is one of the most successful operating systems that have been developed to date.

A Quick and Dirty History

When IBM launched its revolutionary personal computer, the IBM PC, in August 1981, it came complete with a 16-bit operating system from Microsoft, MS-DOS 1.0. This was Microsoft's first operating system, and it also became the first widely used operating system for the IBM PC and its clones. MS-DOS 1.0 was actually a renamed version of QDOS (Quick and Dirty Operating System), which Microsoft bought from a Seattle company, appropriately named Seattle Computer Products, in July 1981. QDOS had been developed as a clone of the CP/M eight-bit operating system in order to provide compatibility with the popular business applications of the day such as WordStar and dBase. CP/M (Control Program for Microcomputers) was written by Gary Kildall of Digital Research several years earlier and had become the first operating system for microcomputers in general use.

QDOS was written by Tim Paterson, a Seattle Computer Products employee, for the new Intel 16-bit 8086 CPU (central processing unit), and the first version was shipped in August, 1980. Although it was completed in a mere six weeks, QDOS was sufficiently different from CP/M to be considered legal. Paterson was later hired by Microsoft. Microsoft initially kept the IBM deal a secret from Seattle Computer Products. And in what was to become another extremely fortuitous move, Bill Gates, the not uncontroversial cofounder of Microsoft, persuaded IBM to let his company retain marketing rights for the operating system separately from the IBM PC project. Microsoft renamed it PC-DOS (the IBM version) and MS-DOS (the Microsoft version). The two versions were initially nearly identical, but they eventually diverged.

The acronym DOS was not new even then. It had originally been used by IBM in the 1960sin the name of an operating system (i.e., DOS/360) for its System/360 computer. At that time the use of disks for storing the operating system and data was considered cutting edge technology. Until its acquisition of QDOS, Microsoft had been mainly a vendor of computer programming languages. Gates and co-founder Paul Allen had written Microsoft BASIC and were selling it on disks and tape mostly to PC hobbyists.

MS-DOS soared in popularity with the surge in the PC market. Revenue from its sales fuelled Microsoft's phenomenal growth, and MS-DOS was the key to company's rapid emergence as the dominant firm in the software industry. This product continued to be the largest single contributor to Microsoft's income well after it had become more famous for Windows. Subsequent versions of MS-DOS featured improved performance and additional functions, not a few of which were copied from other operating systems. For example, version 1.25, released in 1982, added support for double-sided disks, thereby eliminating the need to manually turn the disks over to access the reverse side.

Version 2.0, released the next year, added support for directories, for IBM's then huge 10MB hard disk drive (HDD) and for 360KB, 5.25-inch floppy disks. This was followed by version 2.11 later in the same year, which added support for foreign and extended characters. Version 3.0 launched in 1984, added support for 1.2MB floppy disks and 32MB HDDs. This was soon followed by version 3.1, which added support for networks. Additions and improvements in subsequent versions included support for multiple HDD partitions, for disk compression and for larger partitions as well as an improved diskchecking utility, enhanced memory management, a disk defragmenter and an improved text editor.

The final major version was 7.0, which was released in 1995 as part of Microsoft Windows 95. It featured close integration with that operating system, including support for long filenames and the removal of numerous utilities, some of which were on the Windows 95 CDROM. It was revised in 1997 with version 7.1, which added support for the FAT32 file system on HDDs.

3.0 OPERATING SYSTEM FUNCTIONS

3.1 SCHEDULING

Scheduling is a key concept in computer multitasking, multiprocessing operating system and real-time operating system designs. **Scheduling** refers to the way processes are assigned to run on the available CPUs, since there are typically many

more processes running than there are available CPUs. This assignment is carried out by software's known as a **scheduler** and **dispatcher**.
Objectives of a scheduler

 CPU utilization - to keep the CPU as busy as possible.
 Throughput - number of processes that complete their execution per time unit.
 Turnaround - total time between submission of a process and its completion.
 Waiting time - amount of time a process has been waiting in the ready queue.
 Response time - amount of time it takes from when a request was submitted ntil the first response is produced.
 Fairness - Equal CPU time to each thread.

But MS-DOS is non-multitasking, and as such did not feature a scheduler. MS-DOS was not designed to be a multi-user or multitasking operating system, but many attempts were made to retrofit these capabilities. Since it does not perform scheduling functions, when you run a sub process synchronously on MS-DOS, make sure the program terminates and does not try to read keyboard input. If the program does not terminate on its own, you will be unable to terminate it, because MS-DOS provides no general way to terminate a process. Pressing "ctrl C" or `C-<BREAK>' might
sometimes help in these cases.

3.2 MEMORY MANAGEMENT

MS-DOS Memory Management Functions

 Provide students with a brief overview of memory management in the MS-DOS operating system. Mention that to run a second job, the user must close or pause the first file before opening the second.

 Point out that the Memory Manager uses a first-fit memory allocation scheme in early DOS versions because it is the most efficient strategy in a single-user environment.

 Discuss briefly the two forms of main memory, ROM and RAM. MS-DOS provides three memory management functions- allocate, deallocate, and resize (modify). For most programs, these three memory allocation calls are not used.When DOS executes a program, it gives all of the available memory, from the start of that program to the end of RAM, to the executing process. Any attempt to

allocate memory without first giving unused memory back to the system will produce an “insufficient memory” error.

ALLOCATE MEMORY

Function (ah): 48h
Entry parameters: bx- Requested block size (in paragraphs)
Exit parameters: If no error (carry clear):
ax:0 points at allocated memory block
If an error (carry set):
bx- maximum possible allocation size
ax- error code (7 or 8)
This call is used to allocate a block of memory. On entry into DOS, bx contains the size of the requested block in paragraphs (groups of 16 bytes). On exit, assuming no error, the ax register contains the segment address of the start of the allocated block. If an error occurs, the block is not allocated and the ax register is returned containing the error code.
If the allocation request failed due to insufficient memory, the bx register is returned containing the maximum number of paragraphs actually available.

DEALLOCATE MEMORY

Function (ah): 49h
Entry parameters: es:0- Segment address of block to be deallocated
Exit parameters: If the carry is set, ax contains the error code (7,9)
This call is used to deallocate memory allocated via function 48h above. The es register cannot contain an arbitrary memory address. It must contain a value returned by the allocate memory function. You cannot use this call to deallocate a portion of an allocated block. The modify allocation function is used for that operation.

MODIFY MEMORY ALLOCATION

Function (ah): 4Ah
Entry parameters: es:0- address of block to modify allocation size
bx- size of new block
Exit parameters: If the carry is set, then
ax contains the error code 7, 8, or 9
bx contains the maximum size possible (if error 8)

This call is used to change the size of an allocated block. On entry, es must contain the segment address of the allocated block returned by the memory allocation function. Bx must contain the new size of this block in paragraphs. While you can almost always reduce the size of a block, you cannot normally increase the size of a block if other blocks have been allocated after the block being modified. Keep this in mind when using this function.
Group C Page 8

3.3 FILE SYSTEM

Before we go any further, it would be a good idea to look at the DOS file system. The **file system** lets us store information in named **files**. You can call a file anything you like which might help you remember what it contains as long as you follow certain basic rules:

1. File names can be up to 8 characters long. You can use letters and digits but only a few punctuation marks (! $ % # ~ @ - ( ) _ { }). You can't exceed 8 characters or use spaces or characters like * or ? or +. Names are case-insensitive, i.e. it doesn't matter whether you use capitals or lowercase letters; "A" and "a" are treated as the same thing.
2. File names can also have an **extension** of up to three characters which describes the type of file. There are some standard extensions, but you don't have to use them.
Examples include COM and EXE for executable programs, TXT for text files, BAK
for backup copies of files, or CPP for C++ program files. The extension is separated by a dot from the rest of the filename.
For example, a file called FILENAME.EXT has an 8-character name (FILENAME) followed by a three-character extension (.EXT). You could also refer to it as filename.txt since case doesn't matter, but I'm going to use names in capitals for emphasis throughout this document. Files are stored in **directories**; a directory is actually just a special type of file which holds a list of the files within it. Since a directory is a file, you can have directories within directories. Directory names also follow the same naming rules as other files, but although they can have an extension they aren't normally given one (just an 8-character name). The system keeps track of your **current directory**, and if you just refer to a file using a name like FILENAME.EXT it's assumed you mean a file of that name in the current directory. You can specify a **pathname** to identify a file which includes the directory name as well; the directory is separated from the rest of the name by a backslash ("\"). For example, a file called LETTER1.TXT in a directory called

LETTERS can be referred to as LETTERS\LETTER1.TXT (assuming that the current directory contains the LETTERS directory as a subdirectory). If LETTERS contains a subdirectory called PERSONAL, which in turn contains a file called DEARJOHN.TXT, you would refer to this file as Group C Page 9

LETTERS\PERSONAL\DEARJOHN.TXT (i.e. look in the LETTERS directory for PERSONAL\DEARJOHN.TXT, which in turn involves looking in the PERSONAL subdirectory for the file DEARJOHN.TXT).
Every disk has a **root directory** which is the main directory that everything else is part of. The root directory is called "\", so you can use **absolute pathnames** which don't depend on what your current directory is. A name like \LETTERS\LETTER1.TXT always refers to the same file regardless of which directory you happen to be working in at the time; the "\" at the beginning means "start looking in the root directory", so \LETTERS\LETTER1.TXT means "look in the root directory of the disk for a subdirectory called LETTERS, then look in this subdirectory for a file called LETTER1.TXT". Leaving out the "\" at the beginning makes this a **relative pathname** whose meaning is relative to the current directory at the time. If you want to refer to a file on another disk, you can put a letter identifying the disk at the beginning of the name separated from the rest of the name by a colon (":"). For example,

A:\LETTER1.TXT refers to a file called LETTER1.TXT in the root directory of drive A. DOS keeps track of the current directory on each disk separately, so a relative pathname like A:LETTER1.TXT refers to a file called LETTER1.TXT in the currently-selected directory on drive A. For convenience, all directories (except root directories) contain two special names: "." refers to the directory itself, and ".." refers to the parent directory (i.e. the directory that contains this one). For example, if the current directory is \LETTERS\PERSONAL, the name ".." refers to the directory \LETTERS, "..\BUSINESS" refers to \LETTERS\BUSINESS, and "..\.." refers to the root directory "\".
Group C Page 10

3.4 PROCESS MANAGEMENT

MS-DOS boot process

1. The BIOS, having completed its test and setup functions, loads the boot code found in the master boot record and then transfers control of the system to it. At that point, the master boot record code is executed. If the boot device is a floppy disk, the process skips to step 7 below.

2. The next step in the process is the master boot code examining the master partition table. It first must determine if there is an extended DOS partition, then it must determine if there is a bootable partition specified in the partition table.

3. If the master boot code locates an extended partition on the disk, it loads the extended partition table that describes the first logical volume in the extended partition. This extended partition table is examined to see if it points to another extended partition table. If it does, this second table is examined for information about the *second* logical volume in the extended partition. Logical volumes in the extended partition have their extended partition table chained together one to the next. This process continues until all of the extended partitions have been loaded and recognized by the system.

4. Once the extended partition information (if any) has been loaded, the boot code attempts to start the primary partition that is marked active, referred to as the boot partition. If no boot partitions are marked active, then the boot process will terminate with an error. The error message is often the same as that which occurs if the BIOS could not locate a boot device, generally shown on screen as "No boot device", but also can show up as "NO ROM BASIC - SYSTEM HALTED". If there is a primary partition marked active and there is an installed operating system, the boot code will boot it. The rest of the steps presume this example is of an MS- DOS primary partition.

5. At this stage, the master or volume boot sector is loaded into memory and tested, and the boot code that it contains is given control of the remainder of the boot process. 6. The boot code examines the disk structures to ensure that everything is correct. If not, the boot process will end in an error here.

7. During the next step, the boot code searches the root directory of the device being booted for the operating system files that contain the operating system. For MS-DOS, these are the files "IO.SYS", "MSDOS.SYS" and "COMMAND.COM".

8. If no operating system files are found, the boot program will display an error message similar to "Non-system disk or disk error - Replace and press any key when ready". Keep in mind that this message does not means that the system was never booted. It means that the BIOS examined the floppy disk for example and just rejected it because it couldn't boot an operating system. The volume boot code

was indeed loaded and executed, as that is what posts the message when it can't find the operating system files.

9. In the final stages of the boot process, presuming that the operating system files are found, the boot program will load those operating system files into memory and transfer control to them. In MS-DOS, the first is IO.SYS and its code is executed.

10. SYS will then execute MSDOS.SYS. Then the more complete operating system code loads and initializes the rest of the operating system structures beginning with the command interpreter COMMAND.COM and then the execution of the CONFIG.SYS and AUTOEXEC.BAT files. At this point the operating system code itself has control of the computer.

MS-DOS Process Management

This relates to the Operating System's activity of managing the processor in the system, i.e. allocating and de-allocating of the processor to the process. The Operating System decides. Group C Page 12
the same based on the priority of the process depending if there exists any and certain predefined algorithms. Although MS-DOS is a single tasking operating system, this does not mean there can only be one program at a time in memory. However we can still load several programs into memory at one time under DOS. The only catch is that DOS only provides the ability for them to run one at a time in a very specific fashion. Unless the processes are cooperating, their execution profile follows a very strict pattern. That's why Dos exhibit Serial Multi tasking.
Users often wish to perform more than one activity at a time (load a remote file while editing a program) and uni programming does not allow this. So DOS put in things like memory resident programs that invoked asynchronously, but still have separation problems. One key problem with DOS is that there is no memory protection - one program may write the memory of another program, causing weird bugs.

Child Processes in DOS

In Dos we have one process and one thread.

When a DOS application is running, it can load and executing some other programs using the DOS EXEC function. Under normal circumstances, when an application (the parent) runs a second program (the child), the child process

executes to completion and then returns to the parent. This is very much like a procedure call, except it is a little more difficult to pass parameters between the two. Group C Page 13

3.5 INPUTS/OUTPUT IN MSDOS

The core of MS-DOS is a device-independent input/output (I/O) handler, represented on a system disk by the hidden file MSDOS.SYS. It accepts requests from application programs to do high-level I/O, such as sequential or random access of named disk files, or communication with character devices such as the console. The handler processes these requests and converts them to a very low level form that can be handled by the I/O system. Because MSDOS.SYS is hardware independent, it is nearly identical in all MS-DOS versions provided by manufacturers with their equipment. The I/O system is totally device dependent and is represented on the disk by the hidden file IO.SYS. It is normally written by hardware manufacturers (who know their equipment best, anyway) with the notable exception of IBM, whose I/O system was written to IBM's specifications by Microsoft. The tasks required of the I/O system, such as outputting a single byte to a character device or reading a contiguous group of physical disk sectors into memory, are as simple as possible.

Departing From the Windows for MS-DOS I/O Model

The I/O system used in Windows for MS-DOS is based on, and limited by, capabilities of the underlying MS-DOS operating system. The device drivers at the core of this I/O system are commonly:

 Written in assembly language--they're not portable: they can only run on Intel 80x86 family processors.

 Monolithic in design, not layered--similar code for common tasks is repeated in each device driver for a given class of devices. Monolithic design also makes mixing and matching different file systems and device drivers impossible.

 Not designed for pre-emptive multitasking use--Windows has to perform many nasty tricks to allow even non-pre-emptive multitasking with non-multitasking MS-DOS and its non-multitasking device drivers.

 Incompatible with multiprocessor platforms--MS-DOS is inherently a singleprocessor OS, so MS-DOS device drivers are utterly incapable of synchronizing access to shared resources in a multiprocessor situation.
Group C Page 14

3.6 SECURITY

The security syntax set in MSDOS is masked on the command base and the password identifier which cannot be altered by someone who is not authorized. Password can be set for some programs starting from MSDOS like QBASIC.
The command – line is not prone to virus. Also it MSDOS cannot be used by someone who does not know the commands limiting the risk of harming the system. Group C Page 15

4.0 DESIGN ISSUES

MS-DOS Design Criteria

The primary design requirement of MS-DOS was CP/M-80 translation compatibility, meaning that, if an 8080 or Z80 program for CP/M were translated for the 8086 according to Intel's published rules, that program would execute properly under MS-DOS. Making CP/M-80 translation compatibility a requirement served to promote rapid development of 8086 software, which, naturally, Seattle Computer was interested in. There was partial success: those software developers who chose to translate their CP/M-80 programs found that they did indeed run under MS-DOS, often on the first try. Unfortunately, many of the software developers Seattle Computer talked to in the earlier days preferred to simply ignore MSDOS. Until the IBM Personal Computer was announced, these developers felt that CP/M-86 would be *the* operating system of 8086/8088 computers. Other concerns crucial to the design of MS-DOS were speed and efficiency. Efficiency primarily means making as much disk space as possible available for storing data by minimizing waste and overhead. The problem of speed was attacked three ways: by minimizing the number of disk transfers, making the needed disk transfers happen as quickly as possible, and reducing the DOS's "compute time," considered overhead by an application program. The entire file structure and disk interface were developed for the greatest speed and efficiency. The last design requirement was that MS-DOS be written in assembly language. While this characteristic does help meet the need for speed and efficiency, the reason for including it is much more basic. The only 8086 software-development tools available to Seattle Computer at that time were an assembler that ran on the Z80 under CP/M and a monitor/debugger that fit into a 2K-byte EPROM (erasable

programmable read-only memory). Both of these tools had been developed in house. Group C Page 16

5.0 IMPLEMENTATION

MSDOS is a single-user operating system. MS-DOS employs a command line interface and a batch scripting facility via its command interpreter, command.com (all operations to be carried out must be written through commands); without the knowledge of those commands, the user cannot execute a job. Despite its command-line interface, it is easy to learn. The commands of MSDOS are not case sensitive. MSDOS is supported and embedded in other operating system like *Microsoft windows, MacOs;* it can be launched in windows by:

(i) Start -> All programs -> accessories -> command prompt or
(ii) Start -> run -> type **cmd** -> ok
The MSDOS contains five files (IO.SYS, MSDOS.SYS, COMMAND.COM, CONFIG.SYS and AUTOEXEC.BAT). The commands in MSDOS can be categorized in two forms:
INTERNAL and **EXTERNAL** commands.
Group C Page 17

Internal commands are executed without loading a separate program file. Command.com is responsible for the execution of internal commands and the special batch commands. Internal commands are part of the command processor always available to be used. External command exists as executable files handled by separate programs from the DOS diskette. Each program is a member of the .COM or .EXE family. An external command can only be used when the disk containing the program is in drive. They can be used for peripheral devices like printer. A file's full name is called FILESPEC. The FILESPEC for a disk file has four parts: ***drive name, directory name, file name and extension.*** Characters like. " / \ [ ] : | < > + = ; , cannot be used in naming files because they mean other things in MSDOS. When you start up DOS, the current directory is automatically the root directory. The CD will change the current directory. Other folders in the root directory are called sub – directories. A file in any directory can be accessed by typing:

cd ROOT DIRECTORY\SUB-DIRECTORY (Level 1)\ SUB-DIRECTORY (level2)...\filename.ext. CLASSIFYING MS-DOS COMMANDS

Either internal or external command will be specified for each command of MSDOS in this text. MS-DOS commands fall roughly into three categories;
Group C Page 18

1. **Environment Commands:** These report on or affect the operating system environment. Examples are CLS (clear screen), TIME, DATE, VER (display MS-DOS version number), and HELP.

 BREAK (internal): Used from the DOS prompt or in a batch file or in the CONFIG.SYS file to set (or display) whether or not DOS should check for a Ctrl + Break key combination.
BREAK =on|off

 CLS (internal) – clears screen: Clears (erases) the screen. **CLS**

 DATE AND TIME (internal): Displays and/or sets the system date. **DATE mm-dd-yy** or **DATE**

 GRAPHICS (external): Provides a way to print contents of a graphics screen display.

GRAPHICS [printer type][profile] [/B][/R][/LCD][/PB:(id)] [/C][/F][/P(port)]

 MODE (external): Sets mode of operation for devices or communications.
MODE n
MODE LPT#[:][n][,][m][,][P][retry]
MODE [n],m[,T]
MODE (displaytype,linetotal)
MODE COMn[:]baud[,][parity][,][databits][,][stopbits][,][retry]
MODE LPT#[:]=COMn [retry]
MODE CON[RATE=(number)][DELAY=(number)]
MODE (device) CODEPAGE PREPARE=(codepage) [d:][path]filename
MODE (device) CODEPAGE PREPARE=(codepage list) [d:][path]filename
MODE (device) CODEPAGE SELECT=(codepage)
MODE (device) CODEPAGE [/STATUS]
MODE (device) CODEPAGE REFRESH

 VER (internal): Displays the DOS version number. **VER**

 SELECT (external): Formats a disk and installs country-specific information and keyboard codes (starting with DOS Version 6, this command is no longer available).

SELECT [d:] [d:][path] [country code][keyboard code]

2. **Directory and File Commands:** These manipulate files. Examples are COPY, DEL (delete), TYPE (display file to screen) and, DIR (directory - or list all files in current directory). It can be further divided into three (3);

(a) **Commands for disk maintenance**

 BACKUP (external): Makes a backup copy of one or more files. (In DOS Version 6, this program is stored on the DOS supplemental disk.)

BACKUP d:[path][filename] d:[/S][/M][/A][/F:(size)] [/P][/D:date] [/T:time] [/L:[path]filename]

 RESTORE (external): Restores to standard disk storage format files previously stored using the BACKUP command.

RESTORE d: [d:][path]filename [/P][/S][/B:mm-dd-yy] [/A:mm-ddyy][/ E:hh:mm:ss] [/L:hh:mm:ss] [/M][/N][/D]

 RECOVER (external): Resolves sector problems on a file or a disk. (Beginning with DOS Version 6, RECOVER is no longer available).

RECOVER [d:][path]filename or **RECOVER d:**

 VERIFY (internal): Turns on the verify mode; the program checks all copying operations to assure that files are copied correctly.
VERIFY on|off

 FORMAT (external): Formats a disk to accept DOS files. **FORMAT d:[/1][/4][/8][/F:(size)] [/N:(sectors)] [/T:(tracks)][/B|/S][/C][/V:(label)] [/Q][/U][/V]**

 SYS (external): Transfers the operating system files to another disk.
SYS [source] d:

 CHKDSK (external): Checks a disk and provides a file and memory status report.
CHKDSK [d:][path][filename] [/F][/V]

 DISKCOPY (external): Makes an exact copy of a diskette.
DISKCOPY [d:] [d:][/1][/V][/M]

 DISKCOMP (external): Compares the contents of two diskettes.
DISKCOMP [d:] [d:][/1][/8]

 LABEL (external): Creates or changes or deletes a volume label for a disk.
LABEL [d:][volume label]

 VOL (internal): Displays a disk's volume label.
VOL [d:] (b) **Commands for directory control**

 DIR (internal): Displays directory of files and directories stored on disk.
DIR [d:][path][filename] [/A:(attributes)] [/O:(order)]
[/B][/C][/CH][/L][/S][/P][/W]

 ASSIGN (external): Redirects disk drive requests to a different drive.
ASSIGN A:=B: [...] /sta

 MKDIR (internal): Creates a new subdirectory.
MKDIR (MD) [d:]path

 CHDIR (internal): Displays working (current) directory and/or changes to a different directory.
CHDIR (CD) [d:]path or **CHDIR (CD)[..]**

 RMDIR (internal): Removes a subdirectory.
RMDIR (RD) [d:]path

 TREE (external): Displays directory paths and (optionally) files in each subdirectory.
TREE **[d:][path] [/A][/F]**

 PATH (external): Sets or displays directories that will be searched for programs not in the current directory.
PATH; or **PATH [d:]path[;][d:]path[...]**

 JOIN (external): Allows access to the directory structure and files of a drive through a directory on a different drive.
JOIN d: [d:path] or **JOIN d: [/D]**

 SUBST (external): Substitutes a virtual drive letter for a path designation.
SUBST d: d:path or **SUBST d: /D**
(c) **Commands for file control:**

 COPY (internal): copies source file to target file and appends file.
COPY [d:][path]source [d:][path][target] [/V]
Or COPY [d:][path]filename+[d:][path]filename[...][d:][path][filename] [/V]

 COMP (external): Compares two groups of files to find information that does not match. **COMP [d:][path][filename] [d:][path][filename] [/A][/C][/D][/L][/N:(number)]**

 RNAME (internal): Changes the filename under which a file is stored.
RENAME (REN) [d:][path]filename [d:][path]filename

 ERASE/DELETE (internal): Deletes (erases) files from disk.
DEL (ERASE) [d:][path]filename [/P]

 TYPE (internal): Displays the contents of a file.
[d:][path]filename

 PRINT (external): Queues and prints data files.
PRINT [/B:(buffersize)] [/D:(device)] [/M:(maxtick)] [/Q:(value] [/S:(timeslice)] [/U:(busytick)] [/C][/P][/T] [d:][path][filename] [...]