MS-DOS Commands

The command structure of the MS-DOS operating system is emulated by the command processors in the various Win32 operating systems. A knowledge of MS-DOS commands and structures can be quite handy in many situations, especially when it is desirable to create automated (batch) procedures.

Accessing the Command Processor

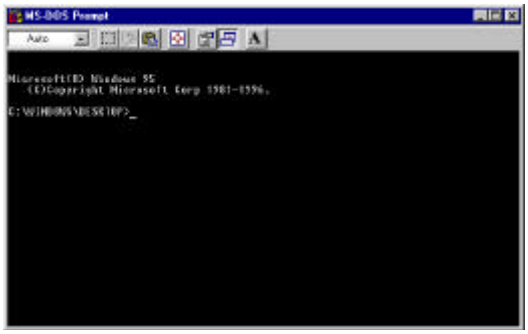
To get a command line under the various Windows 32 bit operating systems, do the following:

1. Type  - *R* to get the following dialog box:

TIP: This dialog box runs executable programs. You can also get it from the "Run..." option of the Start Menu.

2. Type *command* (or *cmd*) into the text window of the box, and press *RETURN*.

3. A command shell window will open that looks very much like this:



TIP: You can open as many command line boxes at the same time as needed. Remember that there is no "undo" for DOS commands - issue them with care!

4. When you are finished with the command processor, type *exit* to close it.

Fundamental Commands

MS-DOS Command	Function
`dir`	Shows a list of files in the current directory. Possible switches include: `/w` (wide) and `/p` (pause every screenful).
`cd`	Change directory. Entered alone on a command line, this command reports the current or "working" directory. (Windows users are accustomed to using the word "folder" instead of directory; they mean the same thing.) Example: `cd \docs\pwrpts\` Changes to the `\docs\pwrpts\` folder. All commands will now look for files in this folder first.

`copy`	Copies one file to another. Examples:
	
	`copy jim.txt mary.txt`
	
	Copies "jim.txt" to "mary.txt" (two identical files now exist.)
	
	`copy *.txt e:\`
	
	Copies all files with an extension of ".txt" to the root (top-level) folder on drive E:.
	
	The *copy* command does not copy subfolders. It can only copy groups of files. To force *binary* file copy, use the `/b` switch.

`xcopy`	Another form of the copy command, *xcopy* more efficiently copies groups of files between two places than the original *copy* command. *xcopy* can also replicate subfolder and directory structures with the `/s` switch. Example: Move everything in the `D:\COMP370` folder (including subdirectories) to the root on network drive `E:` `xcopy d:\comp370\*.* /s e:\`

`diskcopy`	Makes an exact, sector-for-sector copy of a floppy diskette. Example: `diskcopy a: b:` Copies everything from the diskette in drive A to the destination diskette in drive B.

`del`	Deletes a file or group of files. Use with caution -- once you delete a file, it is really gone! (You may be able to recover the file if no other activity has taken place since the delete.) Example:
	
	To delete *jim.txt* from the current folder:
	
	`del jim.txt`
	
	To delete everything ending with ".txt" *(Use wildcards with caution, you may get exactly what you wished for!)*
	
	`del *.txt`

`tree`	Shows the directory or folder structure of a disk. This command may not be available in all 32 bit Windows installations.

`md or mkdir`	Makes a new subdirectory (folder). The new folder will be created in the current directory unless a pathname is specified.
	
	Example: Make a new subdirectory called *myfiles* in the current folder:
	
	`md myfiles`
	
	Example: Make a new subdirectory called *billy* in the subfolder *msoft* on network drive E:
	
	`md e:\msoft\billy`

`rd or rmdir`	Removes a folder or subdirectory. The subdirectory must be empty or the command fails. Example: Remove the subfolder *myfiles* in the current folder: `rd myfiles`

`deltree`	Removes a folder and all subfolders. This command will follow all the branches of the subfolder "tree" until everything has been deleted. *Use with caution!* Example: The folder *myfiles* isn't empty and we wish to remove it and all its contents: `deltree myfiles`

`type`	Shows the contents of a text file on the screen. Useful for short files, annoying for anything else. Example:
	
	`type jim.txt`
	
	Shows the contents of "jim.txt" on screen.
	
	A better way of looking at text files (at least files less than approximately 45K in size) is the *notepad* tool:
	
	`notepad jim.txt`
	
	Brings up *notepad* with *jim.txt*.

Input and Output Redirection

MS-DOS supports "redirection" of the `stdin` (input) and `stdout` (output) streams. Unlike *UNIX*, MS-DOS does not allow redirection of `stderr` (`>2`) and makes no distinction between `stderr` and `stdout`. Redirection is useful for capturing the text output of a program or process, and for automating the input procedures for certain command-line-driven software. DOS supports a crude form of *piping* with (|) which directs the text output of one program to the input stream of another.

Symbol	Function
`>`	Output redirection - directs the console text output of a program to a file. Example: Take the directory listing of the root for network drive E: and put it into a text file called "netdir.txt": dir e:\ > netdir.txt The file "netdir.txt" can be opened using any of the standard tools to verify the contents. It can also be easily placed into an Office document.

<	Input redirection - The target program gets characters from the specified input file when it asks for keyboard data. *This feature works only for software that uses the DOS I/O services. Programs that go directly to hardware or BIOS for keyboard I/O won't redirect!* Example: Run the program *makewav.exe* with commands from the file *wavecmd.txt* instead of the keyboard: `makewave < wavecmd.txt` Caution: Unless you're very sure of what you're doing, don't use input redirection. You can cause a program to hang by failing to supply it with valid keyboard data.
|	Pipe - directs the text (stdout) output of one program to the input (stdin) of another. The most common example is the use of the utility *more.exe* to provide controlled screen output from a process: `examwav | more` Runs the program *examwav.exe* (which presumably makes lots of screen output). The ouput of *examwav* goes through the *more* filter, which automatically pauses the process after each screenful of information is produced.

Batch File Processing

The command processor supports *batch* files, which are merely text files that contain sequences of MS-DOS commands. Batch files always have an extension of ".bat" and can be placed on the desktop, where they can be double-clicked to execute. To execute a batch file from the command line, simply type its name.

Command	Function
`call`	Call a second batch file as a "subroutine." The entire contents of the second batch file are executed, and control returns to the statement after the *call*. Example: Call a batch procedure called *test* from within a batch file: echo Executing first batch file call test echo Back to the first file

`echo`	Echoes the text on the command line to the screen. Also can be used to force the command processor's diagnostic echo OFF for cleaner output.
	
	Example: Echo "Hello" to the console
	
	`echo Hello`
	
	Example: Turn diagnostic echo off:
	
	`echo off`
	
	Example: Echo a blank line to the display.
	
	`echo.`

`goto`	Branch (give control) to a different line in the batch file. The line that will receive control must have an alphanumeric label beginning with a colon.
	
	Example: Show the current folder contents repeatedly, thereby severly annoying the user:
	
	`rem this will be very annoying,`
	`rem press ^c to stop it.`
	`:loop`
	`dir`
	`goto loop`

rem	Insert a remark into a batch file. It's a very good idea to place the following information into the top of every batch file:
	
	`rem ------------------------------`
	`rem  myproc.bat`
	`rem`
	`rem Author: Joe Programmer`
	`rem Date: July 30, 1997`
	`rem Purpose: Performs a complete`
	`rem  backup of the data on`
	`rem  drive D to net drive`
	`rem  H`
	`rem`
	`rem Revision History: None`
	`rem ------------------------------`

if	Performs a conditional test, which is usually based upon the success or failure of a program's execution (ERRORLEVEL). You can also test whether or not certain files exist.
	
	Example: If the file *myfile.txt* exists, copy it to *newfile.txt*. If the file *doesn't* exist, open a *notepad* window for creation of the file.
	
	`if exist myfile.txt goto WeGotIt`
	`rem if we're here, the file does`
	`rem not exist -- create in notepad`
	`notepad myfile.txt`
	`goto End`
	
	`:WeGotIt`
	`copy myfile.txt newfile.txt`
	
	`:End`
	`rem that's all, folks`
	
	Example: Execute the C compiler, and if an error greater than or equal to 2 is reported, tell the user:
	
	`PCC life.c`
	`if errorlevel 2 GOTO Doh`
	`goto OK`
	
	`:Doh`
	`echo Compiler did not run`
	`echo successfully.`
	
	`:OK`
	`rem end of batch procedure`

<table>
<tr>
<td>%1 , %2, etc</td>
<td>Command line arguments. %1 represents the first, %2 represents the second, and so on. Allows more flexible operation by enabling the user to specify the target objects of the batch procedure.

Example: The following batch procedure, called "EditMe.bat", performs the following steps:

1. Checks for the existence of the target file.
2. If the file exists, it creates a backup called backup.txt and opens the original in notepad.
3. If the file doesn't exist, the contents of the file default.txt are copied into it, then the notepad tool opens with the resulting file.

rem EditMe.bat
if exist %1 goto FoundIt
rem doesn't exist, create default version
copy default.txt %1
:FoundIt
copy %1 backup.txt
notepad %1

To use this batch file, the following command line would be issued:

editme filename

Where filename is the name of the text file to be edited.</td>
</tr>
<tr>
<td>@</td>
<td>Precede commands in a batchfile with the @ character to prevent them from being echoed to the display. Example:

@rem
@rem these won't show up at all
@echo This text will still be displayed</td>
</tr>
</table>

Internet Commands

The following commands are useful for Internet troubleshooting, file transfer, and other purposes. They are not really part of the MS-DOS repertoire, but they are generally present in most Windows 95/98/NT installations.

`ping`	Checks for connectivity with a remote computer. You can use the 32-bit IP address or the domain name for the remote system (domain names work only if a DNS server is available in your installation; see *Networking* under *control panel* to check.)
	
	Ping sends a small message to the remote computer and measures the time it takes for the message to make a round-trip.
	
	Example: See if we can talk to host "arrl.org"
	
	`ping arrl.org`
	
	Example: See if we can talk to a computer with IP address 208.128.98.1
	
	`ping 208.128.98.1`

`ftp`	Establishes a file-transfer (FTP) session with a remote computer. Use '?' at the FTP program command line for a list of possible FTP program commands. Example: Establish an FTP session with host "ftp.birch.net": `ftp ftp.birch.net`
`tracert`	Trace route utility. Provides a detailed list of the computers that were accessed under IP to establish a connection with a remote computer. Example: What is the route to *yahoo.com?* `tracert yahoo.com`