*echo*

**Overview:**

Displays messages, or turns command echoing on/off.

**help echo**

Displays messages, or turns command-echoing on or off.

ECHO [ON | OFF]

ECHO [message]

Type ECHO without parameters to display the current echo setting

# *edit (MS-DOS Editor)*

## Overview:

**Note:** If Windows is running then use notepad instead since it is easier to use than edit. See: notepad.

Type edit at the command prompt and the MS-DOS text editor program will be run. The editor is a text editor that uses a simplistic window and menu system inside an MS-DOS window.

Even though the edit program was designed for MS-DOS (a text based environment), it does recognize the mouse. You can use the mouse to select commands and to move the cursor (which appears as an underline). Although the program is archaic compared to today's interface standards, anyone familiar with Windows should be able to use the program.

Since the edit program was originally designed for use in the text based user interface of MS-DOS rather than in the graphic interface of modern computers, the window automatically resizes to 80-characters wide. The window is locked at this size and cannot be resized. If you view a file that has lines wider than 78 characters (2 characters are used for the "border"), you can scroll horizontally by using the left/right/Home/End arrow keys. Unfortunately, there is no horizontal scroll bar. There is a vertical scroll bar and you can use the up/down/PgUp/PgDown arrow keys.

## Menus:

The edit program has menus similar to what appear in modern Windows programs:

* **File:** New, Open..., Save, Save As..., Close, Print..., Exit
* **Edit:** Cut, Copy, Paste, Clear
* **Search:** Find..., Repeat Last Find, Replace...
* **View:** Split Window, Size Window, Close Window
* **Options:** Settings..., Colors...
* **Help:** Commands..., About...

Commands can be selected by the mouse or by keyboard. As with a modern Windows application, menus can be accessed by keyboard by pressing the Alt key and the first letter of the menu title (i.e.: Alt-F, Alt-E, Alt-S, Alt-V, Alt-O, Alt-H). Once a menu is visible, use the arrow keys to move the highlight, and pres Enter key to select.

## Command Variations:

edit

Runs the editor and starts with a blank document. When you select Save or Save As..., the file will be saved to the current directory (unless you select otherwise from the popup dialog). If you want to select a different directory before starting edit, use the cd command before using edit.

edit  *filename*

This command will cause the file named *filename* to be opened by the edit editor. For example, edit mylist.txt would open the mylist.txt file. If the file *filename* does not exist, the editor will start with a blank document with that name (no file is created until you select Save or Save As...).

If you want to edit a file that is in a directory other than the current directory, then use the "cd  *directory*" command first or specify the directory name as part of the filename, such as: "edit \mystuff\ebooks\list.txt". If the *filename* contains spaces, enclose the entire *filename* in quotation marks, such as: edit "\mystuff\my downloads\list.txt"

## help edit

There is no built-in help for the edit command. The program uses a simplistic window and menu system that is fairly easy to understand for any person who has experience using Windows.

# *exit*

## Overview:

Closes the MS-DOS command window.

## help exit

Quits the CMD.EXE program (command interpreter) or the current batch

script.

EXIT [/B] [exitCode]

/B specifies to exit the current batch script instead of

CMD.EXE. If executed from outside a batch script, it

will quit CMD.EXE

exitCode specifies a numeric number. if /B is specified, sets

ERRORLEVEL that number. If quitting CMD.EXE, sets the process

exit code with that number.

# *fc*

## Overview:

Compares two files or sets of files and displays the differences.

## help fc

Compares two files or sets of files and displays the differences between

them

FC [/A] [/C] [/L] [/LBn] [/N] [/OFF[LINE]] [/T] [/U] [/W] [/nnnn]

[drive1:][path1]filename1 [drive2:][path2]filename2

FC /B [drive1:][path1]filename1 [drive2:][path2]filename2

/A Displays only first and last lines for each set of differences.

/B Performs a binary comparison.

/C Disregards the case of letters.

/L Compares files as ASCII text.

/LBn Sets the maximum consecutive mismatches to the specified

number of lines.

/N Displays the line numbers on an ASCII comparison.

/OFF[LINE] Do not skip files with offline attribute set.

/T Does not expand tabs to spaces.

/U Compare files as UNICODE text files.

/W Compresses white space (tabs and spaces) for comparison.

/nnnn Specifies the number of consecutive lines that must match

after a mismatch.

[drive1:][path1]filename1

Specifies the first file or set of files to compare.

[drive2:][path2]filename2

Specifies the second file or set of files to compare.

# *find*

## Overview:

Searches for a text string in a file or files.

## help find

Searches for a text string in a file or files.

FIND [/V] [/C] [/N] [/I] [/OFF[LINE]] "string" [[drive:][path]filename[ ...]]

/V Displays all lines NOT containing the specified string.

/C Displays only the count of lines containing the string.

/N Displays line numbers with the displayed lines.

/I Ignores the case of characters when searching for the string.

/OFF[LINE] Do not skip files with offline attribute set.

"string" Specifies the text string to find.

[drive:][path]filename

Specifies a file or files to search.

If a path is not specified, FIND searches the text typed at the prompt

or piped from another command.

# *findstr*

## Overview:

Search for a regular expression text string in a file or files.

## help findstr

Searches for strings in files.

FINDSTR [/B] [/E] [/L] [/R] [/S] [/I] [/X] [/V] [/N] [/M] [/O] [/P] [/F:file]

[/C:string] [/G:file] [/D:dir list] [/A:color attributes] [/OFF[LINE]]

strings [[drive:][path]filename[ ...]]

/B Matches pattern if at the beginning of a line.

/E Matches pattern if at the end of a line.

/L Uses search strings literally.

/R Uses search strings as regular expressions.

/S Searches for matching files in the current directory and all

subdirectories.

/I Specifies that the search is not to be case-sensitive.

/X Prints lines that match exactly.

/V Prints only lines that do not contain a match.

/N Prints the line number before each line that matches.

/M Prints only the filename if a file contains a match.

/O Prints character offset before each matching line.

/P Skip files with non-printable characters.

/OFF[LINE] Do not skip files with offline attribute set.

/A:attr Specifies color attribute with two hex digits. See "color /?"

/F:file Reads file list from the specified file(/ stands for console).

/C:string Uses specified string as a literal search string.

/G:file Gets search strings from the specified file(/ stands for console).

/D:dir Search a semicolon delimited list of directories

strings Text to be searched for.

[drive:][path]filename

Specifies a file or files to search.

Use spaces to separate multiple search strings unless the argument is prefixed

with /C. For example, 'FINDSTR "hello there" x.y' searches for "hello" or

"there" in file x.y. 'FINDSTR /C:"hello there" x.y' searches for

"hello there" in file x.y.

Regular expression quick reference:

. Wildcard: any character

\* Repeat: zero or more occurances of previous character or class

^ Line position: beginning of line

$ Line position: end of line

[class] Character class: any one character in set

[^class] Inverse class: any one character not in set

[x-y] Range: any characters within the specified range

\x Escape: literal use of metacharacter x

\ Word position: end of word

For full information on FINDSTR regular expressions refer to the online Command

Reference.

# *goto*

## Overview:

Used in a batch program file to jump to a particular line.

## help goto

Directs cmd.exe to a labeled line in a batch program.

GOTO label

label Specifies a text string used in the batch program as a label.

You type a label on a line by itself, beginning with a colon.

If Command Extensions are enabled GOTO changes as follows:

GOTO command now accepts a target label of :EOF which transfers control

to the end of the current batch script file. This is an easy way to

exit a batch script file without defining a label. Type CALL /? for a

description of extensions to the CALL command that make this feature

useful.

# *if*

## Overview:

Used in a batch program file to perform conditional testing.

## help if

Performs conditional processing in batch programs.

IF [NOT] ERRORLEVEL number command

IF [NOT] string1==string2 command

IF [NOT] EXIST filename command

NOT Specifies that Windows XP should carry out

the command only if the condition is false.

ERRORLEVEL number Specifies a true condition if the last program run

returned an exit code equal to or greater than the number

specified.

string1==string2 Specifies a true condition if the specified text strings

match.

EXIST filename Specifies a true condition if the specified filename

exists.

command Specifies the command to carry out if the condition is

met. Command can be followed by ELSE command which

will execute the command after the ELSE keyword if the

specified condition is FALSE

The ELSE clause must occur on the same line as the command after the IF. For

example:

IF EXIST filename. (

del filename.

) ELSE (

echo filename. missing.

)

The following would NOT work because the del command needs to be terminated

by a newline:

IF EXIST filename. del filename. ELSE echo filename. missing

Nor would the following work, since the ELSE command must be on the same line

as the end of the IF command:

IF EXIST filename. del filename.

ELSE echo filename. missing

The following would work if you want it all on one line:

IF EXIST filename. (del filename.) ELSE echo filename. missing

If Command Extensions are enabled IF changes as follows:

IF [/I] string1 compare-op string2 command

IF CMDEXTVERSION number command

IF DEFINED variable command

where compare-op may be one of:

EQU - equal

NEQ - not equal

LSS - less than

LEQ - less than or equal

GTR - greater than

GEQ - greater than or equal

and the /I switch, if specified, says to do case insensitive string

compares. The /I switch can also be used on the string1==string2 form

of IF. These comparisons are generic, in that if both string1 and

string2 are both comprised of all numeric digits, then the strings are

converted to numbers and a numeric comparison is performed.

The CMDEXTVERSION conditional works just like ERRORLEVEL, except it is

comparing against an internal version number associated with the Command

Extensions. The first version is 1. It will be incremented by one when

significant enhancements are added to the Command Extensions.

CMDEXTVERSION conditional is never true when Command Extensions are

disabled.

The DEFINED conditional works just like EXISTS except it takes an

environment variable name and returns true if the environment variable

is defined.

%ERRORLEVEL% will expand into a string representation of

the current value of ERRORLEVEL, provided that there is not already

an environment variable with the name ERRORLEVEL, in which case you

will get its value instead. After running a program, the following

illustrates ERRORLEVEL use:

goto answer%ERRORLEVEL%

:answer0

echo Program had return code 0

:answer1

echo Program had return code 1

You can also using the numerical comparisons above:

IF %ERRORLEVEL% LEQ 1 goto okay

%CMDCMDLINE% will expand into the original command line passed to

CMD.EXE prior to any processing by CMD.EXE, provided that there is not

already an environment variable with the name CMDCMDLINE, in which case

you will get its value instead.

%CMDEXTVERSION% will expand into a string representation of the

current value of CMDEXTVERSION, provided that there is not already

an environment variable with the name CMDEXTVERSION, in which case you

will get its value instead.

# *md (mkdir)*

## Overview:

Creates a directory.

The command mkdir is the same as the md command; they do the same thing. You can use either the mkdir command or use the md command which takes fewer characters to type.

## md /?

Creates a directory.

MKDIR [drive:]path

MD [drive:]path

If Command Extensions are enabled MKDIR changes as follows:

MKDIR creates any intermediate directories in the path, if needed.

For example, assume \a does not exist then:

mkdir \a\b\c\d

is the same as:

mkdir \a

chdir \a

mkdir b

chdir b

mkdir c

chdir c

mkdir d

which is what you would have to type if extensions were disabled.

# *more*

## Overview:

Displays the contents of a file one screen at a time.

## help more

Displays output one screen at a time.

MORE [/E [/C] [/P] [/S] [/Tn] [+n]] < [drive:][path]filename

command-name | MORE [/E [/C] [/P] [/S] [/Tn] [+n]]

MORE /E [/C] [/P] [/S] [/Tn] [+n] [files]

[drive:][path]filename Specifies a file to display one

screen at a time.

command-name Specifies a command whose output

will be displayed.

/E Enable extended features

/C Clear screen before displaying page

/P Expand FormFeed characters

/S Squeeze multiple blank lines into a single line

/Tn Expand tabs to n spaces (default 8)

Switches can be present in the MORE environment

variable.

+n Start displaying the first file at line n

files List of files to be displayed. Files in the list

are separated by blanks.

If extended features are enabled, the following commands

are accepted at the -- More -- prompt:

P n Display next n lines

S n Skip next n lines

F Display next file

Q Quit

= Show line number

? Show help line

Display next page

Display next line