# IF

Conditionally perform a command.

**File syntax**

```
IF [NOT] EXIST filename command

IF [NOT] EXIST filename (command) ELSE (command)
```

**String syntax**

```
IF [/I] [NOT] item1==item2 command

IF [/I] item1 compare-op item2 command

IF [/I] item1 compare-op item2 (command) ELSE (command)
```

**Error Check Syntax**

```
IF [NOT] DEFINED variable command

IF [NOT] ERRORLEVEL number command

IF CMDEXTVERSION number command
```

**key**

```
item         May be a text string or an environment variable
             a variable may be modified using either
             Substring syntax or Search syntax

command      The command to perform

NOT          perform the command if the condition is false.

==           perform the command if the two strings are equal.

/I           Do a case Insensitive string comparison.

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 >=

              This 3 digit syntax is necessary because the > and <
              symbols are recognised as redirection operators
```

IF ERRORLEVEL *n* statements should be read as IF *Errorlevel* >= *number*
i.e.
IF ERRORLEVEL 0 will return TRUE when the errorlevel is 64

An alternative and often better method of checking Errorlevels is to use the string syntax along with the %ERRORLEVEL% variable:

```
IF %ERRORLEVEL% GTR 0 Echo An error was found
IF %ERRORLEVEL% LSS 0 Echo An error was found

IF %ERRORLEVEL% EQU 0 Echo No error found
IF %ERRORLEVEL% EQU 0 (Echo No error found) ELSE (Echo An error was found)
IF %ERRORLEVEL% EQU 0 Echo No error found || Echo An error was found
```

Note some errors are negative numbers.
When working with errorlevels in a batch file it's a good idea to also use SETLOCAL so that the %ERRORLEVEL% variable is reset each time the batch file runs.

IF EXIST *filename* will return true if the file exists (this is not case sensitive).

**Examples:**

```
IF EXIST C:\install.log (echo complete) ELSE (echo failed)

IF DEFINED _department ECHO Got the department variable

IF DEFINED _commission SET /A _salary=%_salary% + %_commission%

IF CMDEXTVERSION 1 GOTO start_process

IF %ERRORLEVEL% EQU 2 goto sub_problem2
```

## Does %1 exist?

To test for the existence of a command line parameter - use empty brackets like this

```
IF [%1]==[] ECHO Value Missing
or
IF [%1] EQU [] ECHO Value Missing
```

In the case of a variable that may be NULL - a null variable will remove the variable definition altogether, so testing for NULLs becomes easy:

IF NOT DEFINED *_example* ECHO Value Missing

IF DEFINED will return true if the variable contains any value (even if the value is just a space)

## Test the existence of files and folders

IF EXIST *name* - will detect the existence of a file or a folder - the script empty.cmd will show if the folder is empty or not.

## Brackets

Brackets can be used to split commands across multiple lines. This enables writing more complex IF… ELSE… commands:

```
IF EXIST filename.txt (
    Echo deleting filename.txt
    Del filename.txt
 ) ELSE (
    Echo The file was not found.
 )
```

When using brackets the CMD shell will expand [read] all the variables at the beginning of the code block and use those values even if the variables value has just been changed. Turning on DelayedExpansion will force the shell to read variables at the start of every line.

## Delimiters

If the string being compared by an IF command includes delimiters such as [Space] or [Comma], then either the delimiters must be escaped with a caret ^ or the whole string must be "quoted".
This is so that the IF statement will treat the string as a single item and not as several separate strings.

## Testing Numeric values

IF only parses *numbers* when one of the `compare-op` operators (`EQU, NEQ, LSS, LEQ, GTR, GEQ`) is used.
The == comparison operator always results in a *string* comparison.

This is an important difference because if you compare numbers as strings it can lead to unexpected results: "2" will be greater than "19" and "026" will be greater than "26".

Correct numeric comparison:

```
IF 2 GEQ 15 echo "bigger"
```

Using brackets or quotes will force a string comparison:

```
IF (2) GEQ (15) echo "bigger"
IF "2" GEQ "15" echo "bigger"
```

This behaviour is exactly opposite to the SET /a command where quotes are required.

## Wildcards

Wildcards are not supported by IF, so `%COMPUTERNAME%==SS6*` will not match SS64

A workaround is to retrieve the substring and compare just those characters:

```
SET _prefix=%COMPUTERNAME:~0,3%
IF %_prefix%==SS6 GOTO they_matched
```

## Pipes

When piping commands, the expression is evaluated from left to right, so

`IF... | ...` is equivalent to `(IF ... ) | ...`

you can also use the explicit syntax `IF (... | ...)`

## ERRORLEVEL

To deliberately raise an ERRORLEVEL in a batch script use the EXIT /B command.

It is possible (though not a good idea) to create a string variable called %ERRORLEVEL% (user variable)
if present such a variable will prevent the real ERRORLEVEL (a system variable) from being used by commands such as ECHO and IF.

To test for the existence of a user variable use `SET errorlevel`, or `IF DEFINED ERRORLEVEL`

If Command Extensions are disabled IF will only support direct comparisons: IF ==, IF EXIST, IF ERRORLEVEL
also the system variable `CMDEXTVERSION` will be disabled.

IF is an internal command.

*You see things; and you say 'Why?' But I dream things that never were; and I say 'why not?' ~ George Bernard Shaw*

**Related:**

Using brackets to group and expand expressions.
Conditional execution syntax (AND / OR)
SET - Display or Edit environment variables
ECHO - Display message on screen
EXIT - Set a specific ERRORLEVEL
IFMEMBER - NT Workgroup member (Resource kit)
SC - Is a Service running (Resource kit)
Powershell: if - Conditionally perform a command
Equivalent bash command (Linux): if - Conditionally perform a command