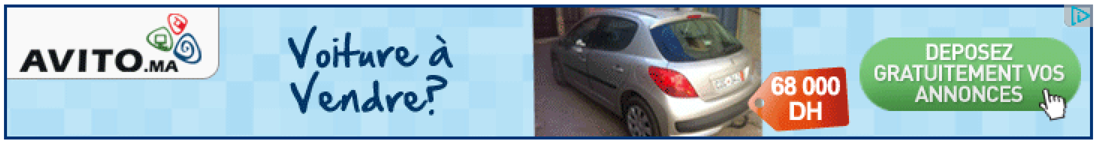
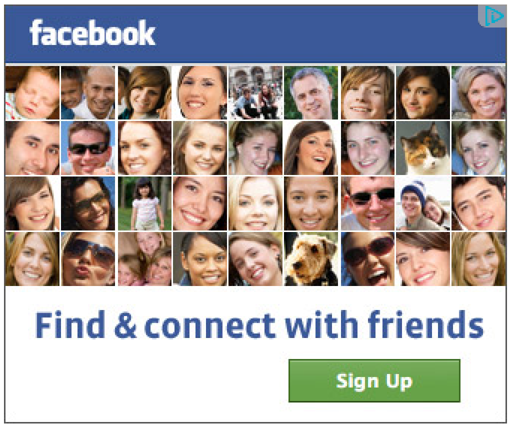
CommandWindows.com

Like 5 Tweet 0 5



More Powerful Batch Files- Branching with "If" statements



Pages

Related links

More Powerful Batch Files Part I - Branching and Looping

By Vic Laurie

The commands "If...else" and "goto" are discussed.

Although many useful tasks can be carried out with simple batch files containing just a few lines, the full power that is available can only be realized with the more advanced methods of branching, iterating, and looping. These methods are among the tools used by programmers to create very sophisticated scripts. However, the concepts are actually quite easy to grasp.and are accessible to those with no background in programming. Just a few extra lines in a batch file using these tools can add a very significant increase in versatility and power. On this page, I will discuss branching. In part two of more advanced methods

Conditional branching with "If" statements

Batch files can make decisions and choose actions that depend on conditions. This type of action makes use of a statement beginning with "If". The basic meaning of an "If" statement is `If something is true then do an action (otherwise do a different action)` The second part of the statement (in parentheses) is optional. Otherwise, the system just goes to the next line in the batch file if the first condition isn't met. The actual syntax is `If (condition) (command1) Else (command2)` The "Else" part is optional. The form "If not" can also be used to test if a condition is false. Note that "If" tests for true or false in the Boolean sense.

"If exist" statement

There is a special "If exist" statement that can be used to test for the existence of a file, followed by a command. An example would be:`If exist somefile.ext del somefile.ext` You can also use a negative existence test: `if not exist somefile.ext echo no file`

Popular Now

More Powerful Batch Files Part II - Iterating with "For"

bikhir.ma
Vendez votre portable sur Bikhir.ma
C'est facile, rapide et bikhir
Déposez vos annonces gratuitement

You May Like

Home

More Powerful Batch Files Part II - Iterating with "For"

The Command Line in Windows: Batch file basics

The Set command and how to use variables in the Windows command line

Windows Command Line |Shell List and Reference

Windows Command Line Interpreter|Shell|DOS Prompt|Batch Files|Scripting

The Command Prompt|Shell in Windows-Introduction

Windows command tips that everybody can use

The Windows command Xcopy and its uses

Windows 7 Command Line List and Reference

Contact & About

Privacy Policy

"If defined" statement

Another special case is "if defined ", which is used to test for the existence of a variable. For example: `if defined somevariable somecommand` This can also be used in the negative form, "if not defined".

"If errorlevel" statement

Yet another special case is "if errorlevel", which is used to test the exit codes of the last command that was run. Various commands issue integer exit codes to denote the status of the command. Generally, commands pass 0 if the command was completed successfully and 1 if the command failed. Some commands can pass additional code values. For example, there are five exit code values for Xcopy. These exit code values are stored in the special variable **errorlevel**. An example command would be: `if errorlevel` *`n`* `somecommand` where "n" is one of the integer exit codes. Note that the comparison is done by checking if **errorlevel** is greater than or equal to n. If used with "not" the comparison checks if **errorlevel** is less than n.

Operator	Meaning
EQU	equal to
NEQ	not equal to
LSS	less than
LEQ	less than or equal to
GTR	greater than
GEQ	greater than or equal to

Comparison operators

In some cases the condition to be met is obtained by comparing strings. For example `if string1 == string2`Note that the "equals" sign is written twice. This condition is met if the two strings are exactly identical, including case. To render comparisons insensitive to case, use the switch "/i". For more general comparisons, use the operators in Table I. (The operators are given in upper case in the table but they are not case-dependent.) Numerical comparisons only work with all-digit strings. Otherwise, the comparison is done alphabetically. For example "a" is less than "b". For case independence, use the switch "/i". An example command might read: `if /i string1 gtr string2 somecommand`

When comparing variables that are strings, it may be best to enclose the variable name in quotes. For example, use: `if "%1" == somestring somecommand`

The "goto" command

Generally, the execution of a batch file proceeds line-by-line with the command(s) on each line being run in turn. However, it is often desirable to execute a particular section of a batch file while skipping over other parts. The capability to hop to a particular section is provided by the appropriately named "goto" command (written as one word). The target section is labeled with a line at the beginning that has a name with a leading colon. Thus the script looks like . . .

```
goto :label
...some commands
```

```
:label
...some other commands
```

Execution will skip over "some commands" and start with "some other commands". The label can be a line anywhere in the script, including before the "goto" command.

"Goto" commands often occur in "if" statements. For example you might have a command of the type:

```
if (condition)
goto :label
```

The "End of File" (:eof) label for exiting a script

Sometimes it is desirable to terminate a script if a certain condition is met (or not met). One way to exit is to use the special label **:eof** in a **goto** command. The label is not actually placed in the batch file. Windows XP and later recognize **:eof** without any label explicitly placed at the end of the batch file. Thus if you need to test for a particular condition that makes script termination desirable, you can write: `if (condition) goto :eof`Note that this terminates the script but does not necessarily close the command shell.

Looping with "If" and "Goto"

An ancient method for doing repetitive tasks uses a counter,"if" statements, and the "goto" command. The counter determines how many times the task is repeated, the "if" statement determines when the desired number of repetitions has been reached, and the "goto" command allows for an appropriate action to be executed, either the repetitive task or exiting. Generally, the more elegant methods provided by the powerful "for...in...do" command are preferable and they are discussed on the next page. However, for completeness and to illustrate some of what we have discussed, I will give an example that uses the clumsier method.

The simple script below creates the numbers 1 to 99 and sends them to a file. It uses the "set" command to create a variable that is also the counter for how many times we have iterated.

```
@echo off
set /a counter=0
:numbers
set /a counter=%counter% 1
if %counter% ==100 (goto :eof) else (echo %counter% >> E:\count.txt)
goto :numbers
```

(Best programming practice would dictate that the variable **%counter%** be localized or destroyed at the end but for simplicity I have omitted the several extra lines needed to do that. As written, this environment variable would persist until the command shell itself, not just the script, was closed.)

In anticipation, I can note that the same result as the script above can be achieved with a two-line script using the "for" statement discussed on the next page:

```
@echo off
for /l %%X in (1,1,99) do (echo %%X >> E:\count.txt)
```

Back to top Next: Iteration methods

More To Explore

The Command Line in Windows: Batch file basics

The Set command and how to use variables in the Windows command line

Windows Command Line |Shell List and Reference

Windows Command Line Interpreter|Shell|DOS Prompt|Batch Files|Scripting

The Command Prompt|Shell in Windows- Introduction