

Forum:[Hack Forums](#)

Thread: [Occasionally Useful Batch Concepts](#)

WOLGA - August 5, 2015, 10:18 am

This tutorial will cover:FOR

Pointers

Arrays

Multi-Threading

I will try my best to answer any question and comment. If there are any mistakes in this post, please let me know.

FOR ([Click to View](#))

FOR

For this tutorial, I will only be going over the most commonly used functions.

The for command is very versatile and complex.

It generally loops.

/F flag

The /F switch is designed to be loop through files, however it will allow the for command to be as specific as possible.

Command syntax:

Code for /f "[1]" %%[2] in ([3][4][3]) do ([5])

Explanation:

[1] - Options:delims

This option will set the delimiter.

The delimiter is the character (or characters, plural) to split the character set into individual tokens.

Example usage:

Code "delims=|"The previous example will mean the character set will be ONLY split when '|' or '|' is found. '|' and '|' are not included in the tokens

Note: By default, delims are space and tab. New line is always a delimiter.

tokens

This option specifies which tokens will be used or included

Example usage:

Code "tokens=1,2-4,6,*"The previous example will mean tokens 1, 2, 3, 4, 6, and everything that follows will be used. Note that 5 is not included.

usebackq

This option changes the quotes.

[2] - Variable

This single character is the initial variable to be referenced. If used with more than one token, each token can be reference in order lexicographically, starting from the variable.

[3] - Quotes

This part specifies what the character set represents.

Using double-quotes means the character set is a string.

Using single-quotes means the character set is a command (or commands) to be executed. The command output will become the character set to be parsed.

Using no quotes means the character set specifies a file (with path). The file plaintext will become the character set to be parsed.

Example Usage:

Code 'ping google.com'

However, if usebackq is used, the following becomes relevant. Using single-quotes means that the character set is a string.

Using grave accent means the character set is a command (or commands) to be executed. The command output will become the character set to be parsed.

Using no quotes means the character set specifies a file (with path). The file plaintext will become the character set to be parsed.

[4] - Character set

This is the character set to be parsed by the for command.

[5] - Action

This is what should be executed every loop. You may use the variables as specified by the options and variable.

Now let's put it all together.

Command Example Usage:

Code for /f "tokens=1-2 delims= " %%A in ("hello world") do (echo %%B)

Output:

Code world

/L flag

The /L switch is designed to loop across numbers.

Command Syntax:

Code for /l %[%1] in ([2],[3],[4]) do ([5])

Explanation:

[1] - Variable

This is the variable to be referenced throughout the action.

[2] - Initial Quantity

This is the initial quantity to be used.

[3] - Step

This is the quantity that the loop should increment every time.

[4] - Stopping Quantity

This is the quantity that should cause the loop to stop after the initial quantity is sufficiently

incremented to this value, or above.

[5] - Action

This is the commands that should executed every loop. The variable may be referenced.

Now let's put it all together.

Command Example Usage:

```
Code set sum=0
```

```
for /l %%A in (1,1,10) do (set /a sum+=%%A)
```

```
echo %sum%
```

This code has added all the numbers from 1 to 10.

As expected, the output is:

Code 55

Important Things:

Since the action portion is commonly surrounded by parenthesis, you must use variables with delayed expansion if the variable will change throughout the loop, or something unexpected, interesting, and frustrating will surely happen.

Pointers [\(Click to View\)](#)

Pointers

Pointers are variables whose value contain the name of another variable.

Example Usage:

```
Code set "str=Hello World"
```

```
set strp=str
```

```
echo %str%
```

```
echo !%strp%!
```

```
set "!strp!=Bye World"
```

echo !str!The previous code sets a variable, str (string), which contains the value "Hello World". Another variable, strp (string pointer), which contains the value "str", the name of the former variable, is set.

Taking apart the code, we see that during the first ECHO, we get the value of str, and is ECHO'd. During the second ECHO, the value of strp is gotten, str, and then that value of the variable with the name of str is gotten, and ECHO'd.

Next, we set the value of the variable whose name is stored in strp, str, to be equal to "Bye World". We then ECHO the value of str.

There are various uses for pointers.

Arrays [\(Click to View\)](#)

Arrays

An array is a group of variables whose value can be easily accessed, usually by incrementing an

index value, which is usually a variable.

Arrays share some key concepts with pointers.

Here is a example of a array containing 10 elements, each with a value of its own index:

```
Code for /I %%A in (1,1,10) do (  
set array%%A=%%A  
)
```

We can then access each element like so:

```
Code for /I %%A in (1,1,10) do (  
echo !array%%A!  
)
```

You can also set multi-dimensional arrays like so:

```
Code for /I %%A in (1,1,10) do (  
for /I %%B in (1,1,10) do (  
set array%%A.%%B=%%A.%%B  
)  
)
```

This sets a 10x10 dimensional array.

Arrays generally have more use than pointers.

Multi-Threading ([Click to View](#))

Multi-Threading

Multi-threading allows more than one process to be run at the same time (sort of). It is especially useful for non-blocking user interfaces.

The main command for multi-threading in batch is:

```
Code start "" /b /d %cd% cmd /c ""%~0" [1]"
```

The only part about this command you need to know is [1]. The rest is not important to know.

This command will create a instance of CMD running the same batch file in the same window as the thread that started it.

[1] denotes arguments, and should be used in the same was as CALL parameters. These parameters will be available in the same way CALL functions are used.

Found this in my drafts somewhere, and so I decided to just finish it up and post it. Enjoy

?CARUs L?VE?™ - August 5, 2015, 1:56 pm

Well done, Qwerty. I'm going to experiment with that multi-threading

Ventriloquism - August 5, 2015, 6:45 pm

(Aug 5, 2015 01:56 PM)?CARUs L?VE?™ Wrote: [\[CITE\]](#) Well done, Qwerty. I'm going to experiment with that multi-threading

Yeah, same. That looks pretty interesting... thanks bro ?

Arrays aren't bad either. Very cool.

xxv2012 - August 6, 2015, 8:24 am

thank you this is very useful ^_^

WOLGA - August 9, 2015, 10:45 am

(Aug 6, 2015 08:24 AM)xxv2012 Wrote: [\[CITE\]](#) thank you this is very useful ^_^
Thank you for your top-quality response.

?CARUs L?VE?™ - August 9, 2015, 7:45 pm

(Aug 9, 2015 10:45 AM)Qwertyuiopasdfghjklzxc Wrote: [\[CITE\]](#) Thank you for your top-quality response.

Sometimes I wish I could rate comments. I'd give you +3 just for that haha. Also, can you explain your multi-threading? I've played with it but I ended up fork bombing myself lol

WOLGA - August 10, 2015, 5:42 pm

(Aug 9, 2015 07:45 PM)?CARUs L?VE?™ Wrote: [\[CITE\]](#) Sometimes I wish I could rate comments. I'd give you +3 just for that haha. Also, can you explain your multi-threading? I've played with it but I ended up fork bombing myself lol

Of course! I don't blame you at all for having trouble understanding, I explained it poorly (if at all), and will have to update it soon.

Remember that 'game' (<https://hackforums.net/showthread.php?tid=4885997>) a while ago, where the 'enemy' would move, even if you do not press a key? Multi-threading was crucial then. This is because usually, the enemy will not move unless you move.

What the multi-threading did was essentially let 2 pieces of code run, with what I like to call the 'displayer' and the 'input source'.

The input source would run, and then all it would do is wait for a keypress, and then update a file. Nothing fancy there.

The displayer had two roles. For every few sub-seconds that passed, it would check if the file was updated, and then update the display/screen with the new movements. And then, regardless of whether the file was updated or not, it would move the enemy.

As White Light demonstrated, this technique could also be applied to chat rooms.

Of course, non-blocking inputs are not the only function of multithreading. It can be used to pseudo-parallel process code, as White Light used in his quicksort, which is now unfortunately expired ?.

Did I explain it well??

'Palladium - August 11, 2015, 2:13 am

Well done on this tutorial, others will find this very useful.

?CARUs L?VE?™ - August 11, 2015, 12:16 pm

(Aug 10, 2015 05:42 PM)Qwertyuiopasdfghjklzxc Wrote: [\[CITE\]](#) Of course! I don't blame you at all for having trouble understanding, I explained it poorly (if at all), and will have to update it soon.

Remember that 'game' (<https://hackforums.net/showthread.php?tid=4885997>) a while ago, where the 'enemy' would move, even if you do not press a key? Multi-threading was crucial then. This is because usually, the enemy will not move unless you move.

What the multi-threading did was essentially let 2 pieces of code run, with what I like to call the 'displayer' and the 'input source'.

The input source would run, and then all it would do is wait for a keypress, and then update a file. Nothing fancy there.

The displayer had two roles. For every few sub-seconds that passed, it would check if the file was updated, and then update the display/screen with the new movements. And then, regardless of whether the file was updated or not, it would move the enemy.

As White Light demonstrated, this technique could also be applied to chat rooms.

Of course, non-blocking inputs are not the only function of multithreading. It can be used to pseudo-parallel process code, as White Light used in his quicksort, which is now unfortunately expired ?.

Did I explain it well??

I'm curious what kind of games could be made with it. And yes you did lol