

What Are You Waiting FOR?

This article will introduce the FOR command and illustrate basic usage that will expand your scripting possibilities and hopefully make your work a bit more enjoyable. I'm assuming you are running Windows NT or later. There is a version of the FOR command on Windows 9x, but it is not nearly as powerful.

At a command prompt type FOR /? to see basic help information.

In a nutshell, the FOR command allows us to perform some action on a set of files. The command says that FOR every X (the variable) IN some set of files, then DO some command and pass any other command parameters as needed. The variable is usually a letter such as %i. Be careful, as variable names are case sensitive. The rest of the command is not case sensitive, but I have used upper case for emphasis and clarity.

Let's begin with a very basic example. At a command prompt type:

```
FOR %i IN (*.*) DO echo %i
```

Our command is that FOR every file in the current directory (*.*), assign it a variable %i and then simply echo that variable to the console. This will list all files in the current directory. It is comparable to the dir /b command but listing only files. The command to be run, echo in this case, can be any OS command, executable or batch file. You'll notice that when you run this example you get output like the following:

```
E:\scripts>echo xlshares.vbs
xlshares.vbs
```

```
E:\scripts>echo xltest.vbs
xltest.vbs
```

```
E:\scripts>echo test2k.vbs
test2k.vbs
```

To clean up the output, use the @ symbol before your command:

```
FOR %i IN (*.*) DO @echo %i
```

Now let's try another example. Suppose you have a batch file that will run a virus scan on a document, compress it using PKZIP, and copy it to an archive server or ZIP drive. I know this may not be the most practical example, but you get the idea. Let's call the batch file SAVEIT.BAT and it takes the filename as a parameter, SAVEIT expenses.doc. Obviously, you want to run this batch file on many documents without intervention. At a command prompt you type:

```
FOR %i IN (*.doc) DO @C:\SCRIPTS\SAVEIT %i
```

What we are asking is that for every .doc file in the current directory, we want to run SAVEIT and pass it the filename in the set as the parameter. The set (*.doc) assumes the existing directory. If you wanted to run SAVEIT.BAT on files in another directory, simply use the full path for the set.

```
FOR %i IN ("e:\my documents\april 99\reports\*.doc") DO @c:\SCRIPTS
\SAVEIT %i
```

Of course, you don't want to have to be at a console manually typing all of this. You want to integrate it this into a batch file. The only thing you need to remember to change is to use %%i for the variable, as opposed to %i.

```
@echo off
```

```

::ADMIN.BAT
For %%i in ("e:\my documents\april 99\reports\*.doc") DO @C:\SCRIPTS
\SAVEIT %%i
For %%i in ("e:\your documents\*.*)" DO @C:\SCRIPTS\SAVEIT %%i
For %%i in ("f:\public\inventory reports\*.xls") DO @C:\SCRIPTS\SAVEIT
%%i

```

We schedule ADMIN.BAT to run and the specified documents will be scanned, compressed and archived.

Now let's look at some advanced uses of the FOR command and demonstrate how to use this powerful command to enhance your command line scripts.

If you type FOR /? at a command prompt, you will get a lengthy help screen. One of the switches is /L which will perform a series of commands based on a number sequence. Here is the help screen for this particular switch:

```
FOR /L %variable IN (start,step,end) DO command [command-parameters]
```

The set is a sequence of numbers from start to end, by step amount. So (1,1,5) would generate the sequence 1 2 3 4 5 and (5,-1,1) would generate the sequence (5 4 3 2 1)

To see this in action, at a command prompt, type FOR /L %i in (1,1,5) do @echo %i and you should see:

```

1
2
3
4
5

```

You might at first think this isn't very useful, but supposed your servers are named Server01, Server02 up to Server07. And you have a process you want to run against the servers. You could create a text list and use the /F switch (to be discussed in a later article) to execute your script, but why not do it on the fly?

```
FOR /L %i in (1,1,7) do @c:\scripts\monthly.bat Server0%i
```

You'll notice that /L won't use leading zeroes so I have to include one for my script to work. Or perhaps you do need a large text list but don't want to have to type a hundred computer or server names. I recently needed a text list of 80 computers. I simply ran:

```
FOR /L %i in (1,1,80) do @echo SYSPC%i >>syspc.txt
```

I had to manually edit the first 9 entries to correct the leading zero, but that was certainly easier than cutting and pasting or anything else. If you use this technique, be sure to use the >> redirection symbols. You want to append each entry to the file. If you only use >, your file will only have the last entry, SYSPC80.

This technique would also work for populating user accounts, perhaps you are testing Active Directory and want to quickly setup 1000 user accounts. Assuming you are using a script to create the accounts, such as Addusers.vbs, use the following command then go get a cup of coffee while it runs:

```
FOR /L %i in (10,1,1000) do @cscript addusers.vbs USR%i
```

If you want to see what variable will be passed, execute the command again using do @echo USR%i.

Finally, you can create a list of pseudo-Random numbers with the following command:

```
FOR /L %i in (1,3,30) do @set /a (%i*%RANDOM%)/7 && Echo.
```

Be sure to use the period (.) after the last echo command, otherwise your numbers will all run together. Of course, you can use whatever mathematical formula you would like and any seed pattern, in my case (1,3,30).

The last option we want to look at has three variations:

```
FOR /F ["options"] %variable IN (file-set) DO command [command-parameters]
FOR /F ["options"] %variable IN ("string") DO command [command-parameters]
FOR /F ["options"] %variable IN ('command') DO command [command-parameters]
```

In the first variation you use a text file, perhaps a list of computers, as the file set. Let's create a text file called Machines.txt:

```
;Machines.txt
DC01
DC02
WebSrv01
```

By default, the file is read one line at a time from beginning to end (blank lines are skipped). Your server list file may begin with a title or other comments which you don't want processed. You can use the EOL option to skip those lines.

With this file you would run: for "eol=;" %i in (machines.txt) do @c:\scripts\myscript.vbs %i. You can use any character you want, but you are limited to one. You could also have used the SKIP option to start at line 2: for "skip=1" %i in (machines.txt) do @c:\scripts\myscript.vbs %i.

The FOR command not only allows you to read a line from a file, but to parse it as well. To do this we use the DELIMS option. The default delimiters are the space and tab, but you can use anything you want, such as a comma. To tell FOR which element, or elements, of the line you want to extract, you use the TOKENS option. Now we have a csv file of new users:

```
`first,last,samid,dept
Alice,Jones,ajones,Engineering
Bob,Smith,bsmith,Sales
Chris White,cwhite,Marketing
```

We can use the following command (all on one line) to parse the file as parameters for DSADD to create new users in Active Directory.

```
For /f "eol=` delims=, tokens=1-4" %a in (users.txt) do@dsadd user
"CN=%a %b, OU = Employees,DC=Mycompany,DC=local" -samid %c -fn %a -ln %b -
pwd P@ssw0rd -dept %d -mustchgpwd yes
```

The FOR command lets us skip the first line since it is an EOL delimiter. We specify a comma as the delimiter and that we want to use all the values, identified as %a, %b, %c and %d. DSADD creates new users in the Employees OU, defines the user's department, sets the default password and forces the user to change password at next logon. You could also put this in a batch file, just remember that %a becomes %%a.

Using a text file is handy for routine and planned events, or where you know what your "file set" will be. What if you want to run something on the fly? You can use the /F switch to parse through a command sequence like

this:

```
For /f "tokens=1 delims=." %i in ('dir /b') do @echo %i
```

Let's say you have a script that requires the file name as a parameter and you want to run it on every exe file on the c: drive.

```
For /f "tokens=*" %x in ('dir /s /b *.exe') do @c:\scripts\fixfile.vbs  
%x
```

One final note on parsing command strings, if you are using a special character, such as |, <, or >, you need to put a caret (^) before it so that the FOR command processes properly. Without it you'll probably get messages like "| was unexpected at this time."

There are many great command line utilities in Windows Support Tools and the Resource Kits. Not to mention your own scripting solutions. As you've seen, the FOR command is a veritable Swiss Army knife utility. I hope you'll find room for it in your tool belt.

copyright 2004 JDH Information Technology Solutions, Inc.

All Rights reserved

<http://www.jdhitsolutions.com>