Reading the output of a command into a batch file variable

devblogs.microsoft.com/oldnewthing/20120731-00

July 31, 2012

Raymond Chen

It's Day Two of Batch File Week. Don't worry, it'll be over in a few days.

There is no obvious way to read the output of a command into a batch file variable. In unix-style shells, this is done via backquoting.

```
x=`somecommand`
```

The Windows command processor does not have direct backquoting, but you can fake it by abusing the `FOR` command. Here's the evolution:

The `/F` flag to the `FOR` command says that it should open the file you pass in parentheses and set the loop variable to the contents of each line.

```
for /f %%i in (words.txt) do echo [%%i]
```

The loop variable in the `FOR` command takes one percent sign if you are executing it directly from the command prompt, but two percent signs if you are executing it from a batch file. I'm going to assume you're writing a batch file, so if you want to practice from the command line, remember to collapse the double percent signs to singles.

I'm cheating here because I know that `words.txt` contains one word per line. By default, the `FOR` command sets the loop variable to the first word of each line. If you want to capture the entire line, you need to change the delimiter.

```
for /f "delims=" %%i in (names.txt) do echo [%%i]
```

There are other options for capturing say the first and third word or whatever. See the `FOR /?` online help for details.

Now, parsing files is not what we want, but it's closer. You can put the file name in single quotes to say "Instead of opening this file and reading the contents, I want you to run this *command* and read the contents." For example, suppose you have a program called

`printappdir` which outputs a directory, and you want a batch file that changes to that directory.

```
for /f "delims=" %%i in ('printappdir') do cd "%%i"
```

We ask the `FOR` command to run the `printappdir` program and execute the command `cd "%%i"` for each line of output. Since the program has only one line of output, the loop executes only once, and the result is that the directory is changed to the path that the `printappdir` program prints.

If you want to capture the output into a variable, just update the action:

```
for /f %%i in ('printappdir') do set RESULT=%%i
echo The directory is %RESULT%
```

If the command has multiple lines of output, then this will end up saving only the last line, since previous lines get overwritten by subsequent iterations.

But what if the line you want to save isn't the last line? Or what if you don't want the entire line?

If the command has multiple lines of output and you're interested only in a particular one, you can filter it in the `FOR` command itself…

```
for /f "tokens=1-2,14" %%i in ('ipconfig') do ^
    if "%%i %%j"=="IPv4 Address." set IPADDR=%%k
```

The above command asked to execute the `ipconfig` command and extract the first, second, and fourteenth words into loop variable starting with `%i`. In other words, `%i` gets the first word, `%j` gets the second word, and `%k` gets the fourteenth word. (Exercise: What if you want to extract more than 26 words?)

The loop then checks each line to see if it begins with "`IPv4 Address.`", and if so, it saves the fourteenth word (the IP address itself) into the `IPADDR` variable.

How did I know that the IP address was the fourteenth word? I counted!

```
   IPv4 Address. . . . . . . . . . . : 192.168.1.1
   ---- -------- - - - - - - - - - - - -----------
     1      2    3 4 5 6 7 8 9  11  13      14
                             10  12
```

That's also why my test includes the period after `Address`: The first dot comes right after the word `Address` without an intervening space, so it's considered part of the second "word".

Somebody thought having the eye-catching dots would look pretty, but didn't think about how it makes parsing a real pain in the butt. (Note also that the above script works only for US-English systems, since the phrase `IPv4 Address` will change based on your current language.)

Instead of doing the searching yourself, you can have another program do the filtering, which is important if the parsing you want is beyond the command prompt's abilities.

```
for /f "tokens=14" %%i in ('ipconfig ^| findstr /C:"IPv4 Address"') do ^
  set IPADDR=%%i
```

This alternate version makes the findstr program do the heavy lifting, and then saves the fourteenth word. (But this version will get fooled by the line `Autoconfiguration IPv4 Address`.)

Yes I know that you can do this in PowerShell

```
foreach ($i in Get-WmiObject Win32_NetworkAdapterConfiguration) {
  if ($i.IPaddress) { $i.IPaddress[0] }
}
```

You're kind of missing the point of Batch File Week.

Raymond Chen

Follow