

# Batch equivalent of Bash backticks



When working with Bash, I can put the output of one command into another command like so:

86

```
my_command `echo Test`
```



would be the same thing as



18

```
my_command Test
```

(Obviously, this is just a non-practical example.)

I'm just wondering if you can do the same thing in Batch.

[bash](#)[batch-file](#)[backticks](#)

edited May 4 '10 at 20:18



Pascal Cuoq

68k 6 128 236

asked May 4 '10 at 20:07



MiffTheFox

15.8k 12 59 88

possible duplicate of [How do I get the result of a command in a variable in windows?](#) – Cristian Ciupitu Jul 1 '14 at 16:55

## 5 Answers



You can do it by redirecting the output to a file first. For example:

51

```
echo %* > file.txt
```

Join **Stack Overflow** to learn, share knowledge, and build your career.

Email Sign Up

OR SIGN IN WITH



Google

Facebook



Joey

273k

67

583

612



zvrba

21.3k

3

46

63

14 It requires you to find a place where you have write access to store the temporary file; you have to clean up after yourself; this very example only enables you to read the very first line of input. For all practical purposes the `for /f` variant is a much better one. – Joey May 4 '10 at 21:33

4 @Joey: `%TEMP%` is a good place for that. :) – gravity May 27 '11 at 14:01

@gravity - along with `%random%` – bacar Aug 26 '11 at 11:34

@joey It does read each line, but how to concatenate to a command line argument in a *single* command. – user877329 Mar 19 '14 at 16:47

how does it handle newline(s)? – andrewrk Oct 16 '17 at 13:43

You can get a similar functionality using `cmd.exe` scripts with the `for /f` command:

96

```
for /f "usebackq tokens=*" %%a in (`echo Test`) do my_command %%a
```

Yeah, it's kinda non-obvious (to say the least), but it's what's there.

See `for /?` for the gory details.

Sidenote: I thought that to use `" echo "` inside the backticks in a `" for /f "` command would need to be done using `" cmd.exe /c echo Test "` since `echo` is an internal command to `cmd.exe`, but it works in the more natural way. Windows batch scripts always surprise me somehow (but not usually in a good way).

answered May 4 '10 at 20:15



Michael Burr

288k

41

446

680

2 You usually have to do this when executing shell-builtins from external programs that don't automatically spawn a shell. I.e. C's `system()` was fine, iirc, since it starts a shell in any case but .NET's `Process.Start` needs to explicitly invoke the shell. Something like that, iirc. In any case, I consider this to be the better answer than the accepted one :) – Joey May 4 '10 at 22:19

Join Stack Overflow to learn, share knowledge, and build your career.

Email Sign Up


OR SIGN IN WITH



Google

Facebook

[Jonathan](#) Apr 21 '14 at 21:09

- 3 @Evan: tokens=\* is important. Otherwise, if the result of the command includes spaces, only the first word is output. Run this at the prompt: `for /f %a in ("echo foo bar") do echo %a` . It will print "foo" . – [Dan Dascalescu](#) Nov 28 '14 at 5:59 

Read the documentation for the "for" command: `for /?`

27

Sadly I'm not logged in to Windows to check it myself, but I think something like this can approximate what you want:

```
for /F %i in ('echo Test') do my_command %i
```

answered May 4 '10 at 20:15



[Weeble](#)

10.5k 2 38 59

- 4 In case someone else stumbles on this, when the command is executed from a batch file (\*.bat), both %i variables need to be double percent sign: %%i. – [mgouin](#) Jun 2 '17 at 14:28

You could always run [Bash](#) inside Windows. I do it all the time with [MSYS](#) (much more efficient than [Cygwin](#)).

3

edited Jun 19 '14 at 19:28



[Peter Mortensen](#)

14.2k 19 88 114


answered May 4 '10 at 20:15



[davr](#)

13.2k 16 68 96

Unfortunately MSYS is barely maintained now, to get an up-to-date bash you need to install it separately – [Ed Randall](#) Apr 14 '15 at 6:48

- 1 BusyBox is even smaller (~432KB). Not a full Bash though, just Ash. Get the unofficial Windows port here: [frippy.org/busybox](http://frippy.org/busybox) – [Martin](#) Dec 14 '15 at 16:09 
- 1 @EdRandall - [msys2.org](http://msys2.org) seems ok, however – [sdbbs](#) Feb 14 at 11:35

Join Stack Overflow to learn, share knowledge, and build your career.

Email Sign Up

OR SIGN IN WITH



Google



3

complexity command:

```
SET VV=some_command -many -arguments && another_command -requiring -the-other -command |
handling_of_output | more_handling
for /f "usebackq tokens=*" %%a in (`%VV%`) do mycommand %%a
```

By putting your full and complex command in a variable first, then putting a reference to the variable in the limit rather than putting the complex command directly into the limit of the for loop, you can avoid syntax interpretation issues. Currently if I copy the exact command I have set to the `vv` variable in the example above into where it's used, `%VV%`, it causes syntax errors.

edited Aug 21 '14 at 12:12

answered Jan 28 '14 at 22:49



zb226

5,920

4

30

54



mtalexan

414

1

3

16

- 1 This didn't work for me, but I found that you can do piping directly within the for command if you escape it with a caret (^): `for /f "usebackq tokens=*" %%a in (`command ^| command`) do command %%a` – [Heptite](#) Jan 26 '15 at 2:16

@Heptite Card escaping didn't work for me, but surrounding command line with double quotes did. – [Alexandr Zarubkin](#) Sep 9 '16 at 13:35

@Heptite In my case I didn't know the series of commands I was going to need to run beforehand, they were extracted out of a file in one case, and passed in in another. With the escaping you need to know how many times to escape the command based on how many interpreters it's going to run through, and in my use cases that wasn't a consistent number of interpreters. – [mtalexan](#) Dec 6 '17 at 1:44

Join **Stack Overflow** to learn, share knowledge, and build your career.

Email Sign Up

OR SIGN IN WITH



Google

Facebook