Batch files

<https://ss64.com/nt/for_f.html>

Začátek formuláře



Konec formuláře

[FOR](https://ss64.com/nt/for.html) /F

Loop command: against a set of files - conditionally perform a command against each item.

Syntax

FOR **/F** ["*options*"] %%*parameter* IN (*filenameset*) DO *command*

FOR **/F** ["*options*"] %%*parameter* IN ("Text string to process") DO *command*

Key

options:

delims=*xxx*   The delimiter character(s) (default = a space)

skip=*n* A number of lines to skip at the beginning of the file.

(default = 0)

eol=; Character at the start of each line to indicate a comment

The default is a semicolon ;

tokens=*n* Specifies which numbered items to read from each line

(default = 1)

usebackq Use the alternate quoting style:

- Use double quotes for long file names in "*filenameset*".

- Use single quotes for 'Text string to process'

- Use back quotes for `[command to process](https://ss64.com/nt/for_cmd.html)`

*Filenameset* A set of one or more files.

*command* The command to carry out, including any

command-line parameters.

%%*parameter* A replaceable parameter:

in a batch file use %%G (on the command line %G)

FOR /F processing of a text file consists of reading the file, one line of text at a time and then breaking the line up into individual items of data called 'tokens'. The DO command is then executed with the parameter(s) set to the token(s) found.  
  
By default, /F breaks up the line at each blank space " ", and any blank lines are skipped, this default parsing behavior can be changed by applying one or more of the "*options*" parameters. The option(s) must be contained within "a pair of quotes"

Within a FOR loop the visibility of variables is affected by SETLOCAL [EnableDelayedExpansion](https://ss64.com/nt/delayedexpansion.html), by default variable changes within the loop will not be visible until the loop completes.

usebackq

This option is useful when dealing with a *filenameset* that is a long filename containing spaces, it allows you to put double quotes around the filename. The backquote character ` is just below the ESC key on most keyboards.  
Filenames which don't contan spaces can still be referenced without using quotes.

Usebackq can be abbreviated to useback (undocumented.)

Skip

SKIP will skip processing a number of lines from the beginning of the file.  
SKIP includes empty lines, but after the SKIP is complete, FOR /F ignores (does not iterate) empty lines.

eol

The default end-of-line character is a semicolon ';' when the FOR command reads a text file (or even a character string), any line that STARTS with the eol character will be ignored. In other words it is treated as a comment.   
Use eol=X to change the eol character to X.

Often you will want to turn this feature off so that every line of your data file is processed, in theory "eol=" should turn this feature off, but in practice this fails to work correctly - it will set eol to whatever the next character is, often the quote or space character. One workaround is to set eol to some unusual character that you don’t expect to ever encounter in the data file e.g. "eol=€" or "eol=¬". Another method is to escape every [delimiter](https://ss64.com/nt/syntax-esc.html#delimiters) For /f tokens^=\*^ delims^=^ eol^= %%a in (file.txt) do... (see forum for a [discussion](https://ss64.org/viewtopic.php?id=1544) of this)

None of the options can be repeated, if you include say "eol=# eol=@" then only the second, "eol=@" is applied.

Delims

More than one delimiter can be specified so a string like 'abcd+efg+hijk;lmno;pqr' can be broken up using "delims=;+".

You can use any character as a delimiter, but they are case sensitive.  
If you don’t specify delims it will default to "delims=<tab><space>"  
  
n.b. some text editors will enter the TAB character as a series of spaces, specifying more than one delimiter has been known to cause problems with some data sets.

Tokens

tokens=2,4,6 will cause the second, fourth and sixth items on each line to be processed.  
  
tokens=2-6 will cause the second, third, fourth, fifth and sixth items on each line to be processed.  
  
tokens=\* will cause all items on each line to be processed.  
  
tokens=3\* will process the third token and the 4th + all subsequent items, this can also be written as tokens=3,\*  
  
Each token specified will cause a corresponding parameter letter to be allocated. The letters used for tokens are case sensitive.  
  
If the last character in the tokens= string is an asterisk, then additional parameters are allocated for all the remaining text on the line.

The following [ASCII characters](https://ss64.com/ascii.html) can be used as FOR tokens:  
ASCII 63 - 93 inclusive, 31 tokens: ? @ A B C D E F G H I J K L M N O P Q R S T U V W X Y Z [ \ ]  
ASCII 95-123 inclusive, 29 tokens: \_ ` a b c d e f g h i j k l m n o p q r s t u v w x y z {  
(there are [a few](https://stackoverflow.com/a/8520993/1720814) other characters that can be used, but require [escaping](https://ss64.com/nt/syntax-esc.html#escape))

A single FOR /F command can never parse more than 31 tokens, to use more requires a [workaround](https://stackoverflow.com/a/8520993/1720814) with multiple FOR commands.

The numbers specified in tokens= are automatically sorted, so for example tokens=5,7,1-3 and tokens=1,2,3,5,7 both produce the same result.

Matching the same token more than once (tokens=1,2,1) can give [unpredictable](https://stackoverflow.com/questions/25950181/why-for-f-sets-empty-values-for-repeated-numbers-in-the-rest-of-tokens) results. Token variables can of course be used multiple times: Echo %%G %%H %%G  
  
FOR tokens variables (or parameter names) are global, so in complex scripts which [call](https://ss64.com/nt/call.html) one FOR statement from within another FOR statement you can refer to both sets of parameters.

The precedence/priority of FOR command options is: usebackq > skip > delims > eol > tokens

**Examples**

Copy the files listed in a text file to a new destination:

FOR /f "delims=" %%G in (files.txt) DO copy "\\source\folder\%%G" "H:\destination\%%G"

This assumes that files.txt contains one filename on each line.

Extract data from a text file which contains characters and commas (but no spaces or other punctuation):

January,Snowy,02  
February,Rainy,15  
March,Sunny,25  
  
FOR /F "tokens=1,3 delims=," %%G IN (weather.txt) DO @echo %%G %%H  
  
The tricky part is splitting up each the line into the right tokens, in this case I'm splitting on the comma character ',' this splits the line into 3 chunks of text and we pull out the first and third items with "tokens=1,3"

|  |  |  |
| --- | --- | --- |
| token=1 (%%G) | token=2 | token=3 (%%H) |
|  | (ignored) |  |
| January |  | 02 |
| February |  | 15 |
| March |  | 25 |

%%G is declared in the FOR statement and %%H is implicitly declared via the tokens= option.

An alternative way to retrieve the same data would be:   
FOR /F "tokens=1,2,3 delims=," %%G IN (weather.txt) DO @echo %%G %%I

Splitting a string that includes spaces.

This can be done just as above, replacing "delims=," with "delims= "

It is possible to specify either the tokens= and/or delims= options in any order, but whenever both delims and tokens are specified, they must be separated by a space, this space will NOT count as a token. For this reason it is recommended to always place delims as the last option before the closing quotation, it is much easier to see what is happening with one space (or no spaces) at the end of the string.

Parse a text string

A string of text will be treated just like a single line of input from a file, the string must be enclosed in double quotes (or single quotes with usebackq).  
  
Echo just the date from the following string

FOR /F "tokens=4 delims=," %%G IN ("deposit,$4500,123.4,12-AUG-09") DO @echo Date paid %%G

Parse the output of a command:

FOR /F %%G IN ('"C:\program Files\command.exe"') DO ECHO %%G

Parse the contents of a file:

FOR /F "tokens=1,2\* delims=," %%G IN (C:\MyDocu~1\mytex~1.txt) DO ECHO %%G

FOR /F "usebackq tokens=1,2\* delims=," %%G IN ("C:\My Documents\my textfile.txt") DO ECHO %%G

Filenameset

To specify an exact set of files to be processed, such as all .MP3 files in a folder including subfolders and sorted by date - just use the [DIR /b](https://ss64.com/nt/dir.html)command to create the list of filenames ~ and use [this variant of the FOR command](https://ss64.com/nt/for_cmd.html) syntax.

Unicode

Many of the newer commands and utilities (e.g. [WMIC](https://ss64.com/nt/wmic.html)) output text files in unicode format, these cannot be read by the FOR command which expects [ASCII](https://ss64.com/ascii.html).   
To convert the file format use the [TYPE](https://ss64.com/nt/type.html) command.

Errorlevel

If no data was processed then FOR /F will return ERRORLEVEL = 1

FOR is an [internal](https://ss64.com/nt/syntax-internal.html) command.  
  
*“It's completely intuitive; it just takes a few days to learn, but then it's completely intuitive” - Terry Pratchett.*  
**Related:**  
  
[FOR](https://ss64.com/nt/for.html) - Loop commands  
[FOR](https://ss64.com/nt/for2.html) - Loop through a set of files in one folder  
[FOR /R](https://ss64.com/nt/for_r.html) - Loop through files (recurse subfolders)  [FOR /D](https://ss64.com/nt/for_d.html) - Loop through several folders  
[FOR /L](https://ss64.com/nt/for_l.html) - Loop through a range of numbers  
[FOR /F](https://ss64.com/nt/for_cmd.html) - Loop through the output of a command  
[FORFILES](https://ss64.com/nt/forfiles.html) - Batch process multiple files  
[IF](https://ss64.com/nt/if.html) - Conditionally perform a command   
[SETLOCAL](https://ss64.com/nt/setlocal.html) - Control the visibility of environment variables inside a loop  
Powershell: [ForEach-Object](https://ss64.com/ps/foreach-object.html) - Loop for each object in the pipeline  
  for example: Get-Content files.txt|Foreach{copy-item -path $\_.FullName -destination "H:\destination\"}  
Equivalent bash command (Linux): [for](https://ss64.com/bash/for.html) - Expand *words*, and execute *commands*

Copyright © [SS64.com](https://ss64.com/) 1999-2018  
Some rights reserved

# Replace string in file:

<https://www.tutorialspoint.com/batch_script/batch_script_replace_string.htm>

@echo off

set str=This message needs changed.

echo %str%

set str=%str:needs=has%

echo %str%

The key thing to note about the above program is, the example replaces the word ‘needs’ with the string ‘has’ via the statement %str:needs = has%

Output

The above command produces the following output.

This message needs changed.

This message has changed.