FOR /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`

   Filenameset    A set of one or more files. Wildcards may be used.
                  If (filenameset) is a period character (.) then FOR will
                  loop through every file in the folder.

   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 FOR variables is controlled via SETLOCAL EnableDelayedExpansion

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 cause the 3rd and all subsequent items on each line to be processed

Each token specified will cause a corresponding parameter letter to be allocated.

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 letters used for tokens are case sensitive, so you can specify up to 26 tokens (a-z) or 26 tokens (A-Z)

It is actually possible to assign up to 61 tokens by using the ASCII codes from this range: `? @ 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 [ \ ] ^ _ ` 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 {`
Note the caret character ^ must be escaped with a second caret: `^^`

FOR tokens variables (or parameter names) are global, so in complex scripts which call one FOR statement from within another FOR statement you can refer to both sets of parameters.

Delims

More than one delimiter may be specified so a string like 'abcd+efg+hijk+lmno;pqr;stu+vwzyz' 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.

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. Usebackq can be abbreviated to useback (undocumented.)

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 `For /f tokens^=*^ delims^=^ eol^= %%a in (file.txt) do...` (see forum for a discussion of this)

Examples

Extracting data from this text file:

```
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"

token1 ,	token2 ,	token3
%%G	<ignored>	%%H
January		02
February		15
March		25

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

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 command to create the list of filenames ~ and use this variant of the FOR command syntax.

Unicode

Many of the newer commands and utilities (e.g. WMIC) output text files in unicode format, these cannot be read by the FOR command which expects ASCII.
To convert the file format use the TYPE command.

FOR is an internal command.

"It's completely intuitive; it just takes a few days to learn, but then it's completely intuitive" - Terry Pratchett.

Related:

FOR - Loop commands
FOR - Loop through a set of files in one folder
FOR /R - Loop through files (recurse subfolders)
FOR /D - Loop through several folders
FOR /L - Loop through a range of numbers
FOR /F - Loop through the output of a command
FORFILES - Batch process multiple files
IF - Conditionally perform a command
SETLOCAL - Control the visibility of environment variables inside a loop
Powershell: ForEach-Object - Loop for each object in the pipeline
Equivalent bash command (Linux): for - Expand *words*, and execute *commands*