FOR

Conditionally perform a command several times.

```
syntax-FOR-Files
      FOR %%parameter IN (set) DO command

syntax-FOR-Files-Rooted at Path
      FOR /R [[drive:]path] %%parameter IN (set) DO command

syntax-FOR-Folders
      FOR /D %%parameter IN (folder_set) DO command

syntax-FOR-List of numbers
      FOR /L %%parameter IN (start,step,end) DO command

syntax-FOR-File contents
      FOR /F ["options"] %%parameter IN (filenameset) DO command

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

syntax-FOR-Command Results
      FOR /F ["options"] %%parameter IN ('command to process') DO command
```

The operation of the FOR command can be summarised as...

- Take a set of data
- Make a FOR Parameter `%%G` equal to some part of that data
- Perform a command (optionally using the parameter as part of the command).
- Repeat for each item of data

If you are using the FOR command at the command line rather than in a batch program, use just one percent sign: `%G` instead of `%%G`.

FOR Parameters

The first parameter has to be defined using a single character, for example the letter G.

```
FOR %%G IN ...
```

In each iteration of a FOR loop, the `IN ( ....)` clause is evaluated and %%G set to a different value

If this clause results in a single value then %%G is set equal to that value and the command is performed.

If the clause results in a multiple values then extra parameters are implicitly defined to hold each. These are automatically assigned in alphabetical order `%%H %%I %%J ...` (implicit parameter definition)

If the parameter refers to a file, then enhanced variable reference may be used to extract the filename/path/date/size.

You can of course pick any letter of the alphabet other than %%G.

`%%G` is a good choice because it does not conflict with any of the pathname format letters (a, d, f, n, p, s, t, x) and provides the longest run of non-conflicting letters for use as implicit parameters.
G > H > I > J > K > L > M
Format letters are case sensitive in Windows 2000 and above, so using a capital letter is also a good way to avoid conflicts `%%A` rather than `%%a`.

Examples

```
FOR /F "tokens=1-5" %%G IN ("This is a long sentence") DO @echo %%G %%H %%J
```

will result in the output: `This is long`

Create a set of 26 folders, one for each letter of the alphabet:

```
FOR %%G IN (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) DO (md C:\demo\%%G)
```

Running multiple commands in a FOR loop

Within a FOR loop, variables are expanded at the start of the loop and don't update until the entire DO section has completed.
The following example counts the files in the current folder, but %count% always returns 1:

```
@echo off
SET count=1
 FOR /f "tokens=*" %%G IN ('dir /b') DO (
 echo %count%:%%G
 set /a count+=1 )
```

To update variables within each iteration of the loop we must either use EnableDelayedExpansion or else use the CALL :subroutine mechanism as shown below:

```
@echo off
SET count=1
FOR /f "tokens=*" %%G IN ('dir /b') DO (call :subroutine "%%G")
GOTO :eof

:subroutine
 echo %count%:%1
 set /a count+=1
 GOTO :eof
```

Nested FOR commands

FOR commands can be nested `FOR %%G... DO (for %%U... do ...)`
when nesting commands choose a different letter for each part. you can then refer to both parameters in the final DO command.

If Command Extensions are disabled, the FOR command will only support the basic syntax with no enhanced variables:
FOR %%parameter IN (set) DO command [command-parameters]

FOR is an internal command.

"Those who cannot remember the past are condemned to repeat it" - George Santayana

Related:

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 items in a text file
FOR /F - Loop through the output of a command
Parameters/arguments %~ options
FORFILES - Batch process multiple files
GOTO - Direct a batch program to jump to a labelled line
IF - Conditionally perform a command
Powershell: ForEach-Object - Loop for each object in the pipeline
Equivalent bash command (Linux): for var in [list]; do - Expand *list*, and execute *commands*