Команда FOR - организация циклической обработки результатов выполнения других команд, списков, и строк в текстовых файлах

Команда **FOR** используется для выполнения команды, заданной в виде параметра, для каждого элемента из набора. В качестве элементов могут использоваться файлы, каталоги, наборы строк.

Формат командной строки:

FOR %переменная IN (набор) DO команда [параметры]

Параметры:

%переменная - Однобуквенный подставляемый параметр.

(набор) - Определяет набор, состоящий из одного или нескольких элементов, обрабатываемых командой.

команда - Команда, которую следует выполнить для каждого элемента набора.

параметры - Параметры для команды, выполняемой по отношению к элементам набора.

. В пакетных файлах для команды FOR используется запись

%%переменная вместо **%переменная**. Имена переменных учитывают регистр букв (%i отличается от %I).

Поддерживаются также дополнительные форма команды FOR:

FOR /D %переменная IN (набор) DO команда [параметры]

Ключ /D задает в качестве набора имена каталогов (не файлов).

FOR /R [[диск:]путь] %переменная IN (набор) DO команда [параметры]

Ключ /R задает выполнение команды для каталога [диск:]путь, а также для всех подкаталогов этого пути. Если после ключа /R не указано имя каталога, используется текущий каталог. Если набор - это одиночный символ точки (.), команда просто перечисляет дерево каталогов.

FOR /L %переменная IN (начало,шаг,конец) DO команда [параметры]

Ключ /L задает обработку набора из последовательности чисел с заданными началом, концом и шагом приращения. Так, набор (1,1,5) раскрывается в (1 2 3 4 5), а набор (5,-1,1) - в (5 4 3 2 1)

FOR /F ["ключи"] %переменная IN (набор-файлов) DO команда [параметры]

FOR /F ["ключи"] %переменная IN ("строка") DO команда [параметры]

FOR /F ["ключи"] %переменная IN ('команда') DO команда [параметры]

Ключ /F задает обработку файлов, строковых значений или результатов стандартного вывода другой команды. Набор файлов - содержит имена одного или нескольких файлов, которые по очереди открываются, читаются и обрабатываются. Обработка состоит в чтении файла, разбивке его на отдельные строки текста и разборе каждой строки в ноль или более подстрок. Затем вызывается тело цикла "for", при выполнении которого каждая найденная подстрока используется в качестве значения переменной. По умолчанию ключ /F выделяет из каждой строки каждого файла первую отделенную пробелами подстроку. Пустые строки в файле пропускаются. Необязательный параметр "ключи" служит для переопределения правил разбора по умолчанию. Он представляет собой заключенную в кавычки строку, содержащую одно или несколько ключевых слов для определения параметров разбора. Ключевые слова:

eol=символ - знак начала комментария в конце строки (признак конца обрабатываемых данных строки). Задается в виде одиночного символа.

skip=n - число пропускаемых при обработке строк от начала файла.

delims=xxx - набор разделителей между обрабатываемыми элементами строк. По умолчанию, в качестве разделителей используются пробелы и знаки табуляции.

tokens=x,y,m-n - номера подстрок из каждой строки, передаваемые в тело цикла "for" для каждой итерации. Например, для обычного текстового файла, подстроками будут слова, а разделителями подстрок - пробелы или знаки табуляции. При использовании этого ключа выделяются дополнительные имена переменных. Формат **m-n** представляет собой диапазон подстрок с номерами от m по n. Если последний знак в строке **tokens=** является звездочкой, то создается дополнительная переменная, значением которой будет весь оставшийся текст в строке после разбора последней подстроки.

usebackq - режим обработки кавычек. Строка, заключенная в обратные кавычки, выполняется как команда, строка, заключенная в прямые одиночные кавычки, является строкой символов, а двойные кавычки могут использоваться для задания имен файлов, содержащих пробелы.

Поясняющий пример:

FOR /F "eol=; tokens=2,3* delims=, " %i in (myfile.txt) do @echo %i %j %k

Выполняется разбор файла myfile.txt. Все строки, которые начинаются с символа точки с запятой (eol=;), пропускаются. Вторая и третья подстроки из каждой строки (tokens=2,3) передаются в тело цикла "for", причем подстроки разделяются запятыми и/или пробелами. В теле цикла переменная **%i** принимает значение второй подстроки, **%j** - третьей, а **%k** - все оставшееся поле до конца строки после третьей подстроки . Имена файлов, содержащие пробелы, необходимо заключать в двойные кавычки. Чтобы использовать двойные кавычки, необходимо использовать параметр usebackq, иначе двойные кавычки будут восприняты как определение строки-литерала для разбора.

В данном примере переменная **%i** явно объявлена в инструкции "for", а переменные **%j** и **%k** объявляются неявно с помощью ключа **tokens=** . Ключ **tokens=** позволяет извлечь из одной строки файла до 26 подстрок. Следует помнить, что имена переменных FOR являются **однобуквенными**, с учетом регистра, поэтому одновременно не может быть активно более 52 переменных, задаваемых как явно, так и неявно.

Команда **FOR /F** может также использоваться для обработки явно заданной строки, заключенной в одиночные кавычки и указанной в качестве параметра в скобках. Она будет разобрана так же, как одиночная строка, считанная из входного файла.

В качестве обрабатываемого набора, также, может быть использован вывод (выходные данные) другой команды. В этом случае используется в качестве параметра в скобках **строка в обратных одиночных кавычках** . Эта строка передается для выполнения дочернему обработчику команд CMD.EXE, а вывод этой команды сохраняется в памяти и разбирается так, как если бы это был файл. Пример:

FOR /F "usebackq delims==" %i IN (`set`) DO @echo %i,

Выполняется команда **SET**, отображающая значения переменных среды и команда **FOR /F** выведет их перечень с использованием команды **echo** .

В команде **FOR** возможно использование ссылок на переменные. Допускается применение следующих синтаксических конструкций:

%~I - из переменной %I удаляются обрамляющие кавычки (")
%~fI - переменная %I расширяется до полного имени файла
%~dI - из переменной %I выделяется только имя диска
%~pI - из переменной %I выделяется только путь к файлу
%~nI - из переменной %I выделяется только имя файла
%~xI - из переменной %I выделяется расширение имени файла
%~sI - полученный путь содержит только короткие имена
%~aI - переменная %I расширяется до атрибутов файла
%~tI - переменная %I расширяется до даты /времени файла
%~zI - переменная %I расширяется до размера файла
%~$path:I - проводится поиск по каталогам, заданным в переменной среды **path**, и переменная **%I** заменяется на полное имя первого найденного файла. Если переменная **path** не определена или в результате поиска не найден ни один файл, то этот модификатор заменяется на пустую строку.

При объединении нескольких операторов можно получить следующие результаты:

~dpI - переменная I раскрывается в имя диска и путь
~nxI - переменная I раскрывается в имя файла и его расширение
~fsI - переменная I раскрывается в полный путь с короткими именами
~dp$path:I - проводится поиск по каталогам, заданным в переменной среды **path**, и переменная **I** раскрывается в имя диска и путь к первому найденному файлу.
~ftzaI - переменная I раскрывается в строку, подобную выдаваемой командой DIR

В приведенных выше примерах переменные **I** и **path** можно заменить на другие допустимые значения. Синтаксическая конструкция с символами ~ заканчивается допустимым именем переменной цикла FOR.

Для имен переменных рекомендуется использовать заглавные буквы, например **-I**, что делает эту конструкцию более удобной для чтения и предотвращает ошибочное принятие их за модификаторы, которые не различают регистр.

При использовании команды **FOR** в командных файлах, если внутри цикла нужно выполнить более одной команды, то они заключаются в скобки:

```
FOR %переменная IN (набор) DO (
команда1 [параметры]
команда2
. . .
)
```

Пример:

```
@echo OFF
for /L %%I in (1,1,5) DO (
echo FIRST%%I
ECHO LAST%%I
)
```

Обычно, в командных файлах команда **FOR** используется не только для разбора данных, но и их обработки, что требует использования переменных внутри цикла **FOR**. И здесь возникает проблема - изменения значений переменных не происходит, т.е. их применение внутри скобок невозможно. Подобное явление вызвано не логическими предпосылками, а всего лишь определенными особенностями реализации командного процессора **CMD.EXE**, и это нужно обязательно учитывать при обработке переменных внутри циклов команд FOR и IF. Другими словами, использование значений переменных внутри скобок, требует изменения стандартного режима интерпретации командного процессора. Разработчиками предусмотрена возможность запуска CMD.EXE с параметром **/V:ON**, что включает разрешение отложенного расширения переменных среды с применением символа восклицательного знака (**!**) в качестве разделителя. То есть, параметр /V:ON разрешает использовать **!var!** в качестве значения переменной **var** во время выполнения внутри циклов команд **FOR** и **IF**. Но на практике чаще используется возможность локального включения данного режима внутри командного файла специальной директивой:

```
Setlocal EnableDelayedExpansion
```

После чего, можно обрабатывать принимаемые переменными значения внутри цикла, используя вместо знаков процента восклицательные знаки. Синтаксически, использование обоих разделителей допускается, но результаты этого использования будут разными, что наглядно демонстрируется следующим командным файлом:

```
Setlocal EnableDelayedExpansion
@ECHO OFF
set VAR=before
if "%VAR%" == "before" (
set VAR=after
if "!VAR!" == "after" @echo Со знаком процента=%VAR% , Со знаком
вопроса=!VAR!
)
```

Команда **set VAR=after** выполняется внутри подпрограммы, ограниченной скобками и, если убрать команду **Setlocal EnableDelayedExpansion** или не использовать для получения значения переменной **VAR** восклицательные знаки, ее значение останется старым (тем, что было установлено до входа в цикл команды **FOR**).

Данная особенность реализации командного процессора Windows нередко приводит к неожиданным результатам при использовании групп команд, объединенных скобками в конструкциях **FOR** и **IF** и тогда, когда значение какой-либо переменной изменяется внутри цикла с одной командой. Например, для получения списка файлов текущего каталога такой командный файл работать не будет:

```
set LIST=
for %%i in (*) do set LIST=%LIST% %%i
echo %LIST%
```

Вроде бы, логически все верно, но не учтена особенность обработки значений переменных. Значение переменной **LIST** внутри цикла команды **FOR** изменено не будет, оно останется пустым (задано командой SET LIST=), каким и было на начало цикла FOR. Команда **SET LIST= %LIST% %%I** должна в каждом цикле менять значение переменной LIST на текущее, плюс символ пробела, и плюс текущее значение переменной **I**, которое принимает значение имени файла в текущем каталоге. Синтаксически, команда верная, но из-за озвученной выше особенности реализации командного процессора - не работает, и значение переменной LIST не изменяется. Для того, чтобы это произошло, командный файл нужно изменить, таким же образом, как и в примере для группы команд:

```
Setlocal EnableDelayedExpansion
set LIST=
for %%i in (*) do set LIST=!LIST! %%i
echo %LIST%
```

Теперь, значение переменной **LIST** внутри цикла **FOR** будет изменяться, последовательно принимая значения имен файлов, разделенных пробелом (**set LIST=!LIST! %%i**).

Эту особенность реализации CMD нужно учитывать и при использовании значений системных переменных внутри циклов, как например, переменной ERRORLEVEL:

IF !ERRORLEVEL!==0 вместо **%ERRORLEVEL%==0**