**Описание основных возможностей сценариев командной строки**

Далее кратко описаны полезные возможности, которые можно использовать в командных файлах (\*.bat;\*.cmd aka батники) операционных систем Windows 2000, XP, 2003 и более поздних.

В этих операционных системах введены существенные расширения возможностей сценариев командной строки, которые позволяют делать, например, следующие вещи:

* определение для файла полного и короткого пути к нему, имени, расширения, размера, времени модификации
* выборочная итерация по файлам и каталогам с применением различных команд
* расширенное перенаправление потоков ввода/вывода
* использование переменных, вычисление арифметических выражений, манипуляции со подстроками
* расширенная обработка условий
* ...многое другое

Более подробную помощь нежели в данном обзоре можно найти в стандартной справке в каталоге Windows.   
(Типичный путь к файлу справки: [C:\WINDOWS\Help\ntcmds.chm](file:///C:\WINDOWS\Help\ntcmds.chm). Если Windows установлена в каталог, отличный от C:\WINDOWS, то ссылки на справку из данной страницы работать не будут. Потребуется открыть справку самостоятельно.)  
В частности, крайне полезно почитать следующие разделы: [Общие сведения о командной оболочке](ms-its:C:\WINDOWS\Help\ntcmds.chm::/ntcmds_shelloverview.htm), [Использование пакетных файлов](ms-its:C:\WINDOWS\Help\ntcmds.chm::/batch.htm).

Справку по конкретной команде также всегда можно получить набрав в командной строке "<имя\_команды> /?".   
Например, очень познавательно почитать то, что выдается в результате команд **set /?**, **call /?**, **for /?**, **if /?**.

Помощь приводится на языке установленной операционной системы.

**Вывод**

Вывод на консоль осуществляется с помощью команды **[echo](ms-its:C:\\WINDOWS\\Help\\ntcmds.chm::/echo.htm)**. Например, "echo Здравствуй, читатель".   
При использовании в командных файлы команды echo on и echo off позволяют включать и выключать режим вывода на экран самих исполняемых команд.   
Если поставить перед командой знак @, то она на экране не будет выведена в любом случае. Будет видет только результат ее исполнения.   
Например, "@echo no echo".   
Команда "echo." позволяет вывести пустую строку.

**Текущий каталог**

Текущий каталог позволяет изменять команда **[cd](ms-its:C:\\WINDOWS\\Help\\ntcmds.chm::/chdir.htm)**.   
Рекомендуется всегда использовать с командой опцию /d для смены одновременно и текущего диска.   
Например, "cd /d d:\delta".   
Текущий каталог можно узнать также с помощью псевдопеременной %cd%.   
Например, "@echo Current directory is: %cd%".   
При операциях с путями точка "." обозначает текущий каталог, а две точки ".." - каталог уровнем выше.   
Например, команда "cd ..\.." указывает, что нужно перейти двумя каталогами выше.

**Параметры вызова сценариев**

Из пакетного файла переданные параметры можно узнать с помощью переменных %1, %2, %3 и т.д. по порядку.   
%0 расширяется до полного пути к данному командному файлу.   
%\* в пакетном файле обозначает все аргументы (например %1 %2 %3 %4 %5 ...).

Например, после вызова

**d:\delta\lngs\compile\_all.cmd sol.dpr .**

будем иметь следующую картину:

|  |  |
| --- | --- |
| **Параметр** | **Значение** |
| **%0** | **d:\delta\lngs\compile\_all.cmd** |
| **%1** | **sol.dpr** |
| **%2** | **.** *(точка)* |
| **%3,%4,%5 и т.д.** | *(пусты)* |
| **%\*** | **sol.dpr .** |

Современные ОС используют расширенный режим обработки параметров, в котором доступны следующие модификаторы:

|  |  |
| --- | --- |
| **Модификатор** | **Описание** |
| **%~1** | расширение %1 и удаление любых кавычек ("") |
| **%~f1** | замена %1 полным путем |
| **%~d1** | замена %1 именем диска |
| **%~p1** | замена %1 путем без диска и имени файла |
| **%~n1** | замена %1 именем файла без расширения |
| **%~x1** | замена %1 расширением имени файла |
| **%~s1** | замена полным путем, содержащим только короткие имена |
| **%~a1** | замена %1 атрибутами файла |
| **%~t1** | замена %1 датой и временем модификации файла |
| **%~z1** | замена %1 размером файла |
| **%~$PATH:1** | Поиск в каталогах, перечисленных в переменной среды PATH, замена %1 полным именем первого найденного файла. Если переменная среды не определена или поиск не обнаружил файлов, модификатор выдает пустую строку. |

Описанные модификаторы можно применять в различных сочетаниях для всех параметров. Например:

|  |  |
| --- | --- |
| **Модификатор** | **Описание** |
| **%~dp0** | полный путь (включая завершающий слэш) к каталогу выполняемого скрипта |
| **%~nx1** | имя файла первого параметра с расширением |
| **%~z2** | размер файла второго параметра |
| **%~ftza1** | замена %1 строкой, аналогичной результату работы программы **dir** |

См. также [Использование пакетных параметров командной строки](ms-its:C:\WINDOWS\Help\ntcmds.chm::/percent.htm).

**Проверка условий**

Проверка условий осуществляется с помощью команды **[if](ms-its:C:\\WINDOWS\\Help\\ntcmds.chm::/if.htm)**.   
Синтаксис:   
if [not] строка1==строка2 команда [else выражение]   
При обработке переменных важно учитывать возможность, что они окажутся пустыми. Поэтому, чтобы команда сравнения осталась корректной, операнды лучше заключать в кавычки: if "%1"=="input.txt".   
Расширенный синтаксис:   
if [/i] строка1 оператор\_сравнения строка2 команда [else выражение]   
Расширенный синтаксис позволяет использовать регистронезависимое (/i) сравнение со следующими операторами: EQU (равно), NEQ (не равно), LSS (меньше), LEQ (меньше или равно), GTR (больше), GEQ (больше или равно).

Проверка существования файла:   
if [not] exist имя\_файла команда [else выражение]   
Например: if exist $result$.txt del $result$.txt

Проверка кода возврата:   
if [not] errorlevel число команда [else выражение]   
Условие верно, если предыдущая обработанная команда завершилась с кодом, равным или большим значения число.   
Например:

**@echo off**

**rem данный скрипт копирует все файлы во всех подкаталогах**

**rem исходного каталога (%1) в другой каталог (%2)**

**xcopy %1 %2 /s /e**

**if errorlevel 4 echo Недостаточно памяти, задан недопустимый диск или ошибка в синтаксисе**

**if errorlevel 2 echo Нажата комбинация CTRL+C для остановки копирования**

**if errorlevel 0 echo Копирование прошло успешно**

**Использование меток**

Метки обозначаются идентификатором с двоеточием прямо перед ним.   
Для перехода по меткам используется команда **[goto](ms-its:C:\\WINDOWS\\Help\\ntcmds.chm::/goto.htm)**.   
Существует специальный вариант команды "goto :eof" для перехода на конец исполняемого сценария.

Есть также особый вариант команды **[call](ms-its:C:\\WINDOWS\\Help\\ntcmds.chm::/call.htm)** для повторного вызова текущего скрипта с заданной метки с определенными параметрами.

Пример:

**@rem Такой вот тест :)**

**if %cd%\==%~dp0 goto next**

**@echo Внимание: текущий каталог не совпадает с путем к данному скрипту**

**:next**

**if not exist $result$.txt (@echo $result$.txt не найден) else call :modified $result$.txt**

**goto :eof**

**:modified**

**@echo %1 был модифицирован %~t1**

**if /i ".txt" EQU "%~x1" @echo %1 является текстовым файлом**

**goto :eof**

**Запуск команд**

Запускать новые процессы из командной строки можно двумя способами.

Инструкция "**[cmd](ms-its:C:\\WINDOWS\\Help\\ntcmds.chm::/cmd.htm)** команда" запускает системный пакетный обработчик cmd.exe для выполнения команды команда.   
Если вызов происходит из консоли, то используется имеющаяся, если нет — создается новая.   
Ключи /c и /k указывают, что по завершении исполнения консоль нужно закрыть или оставить, соответственно.

Инструкция "**[start](ms-its:C:\\WINDOWS\\Help\\ntcmds.chm::/start.htm)** команда" запускает обработку команды команда в отдельном параллельном процессе.   
Соответствующие ключи позволяют изменять приоритет запускаемого процесса.   
Ключ /wait указывает, что запустившему процессу следует дождаться завершения запущенного.

Запускать новые командные сценарии из командных сценариев можно также двумя способами.   
Первый:

**@echo Эта строка выполнится перед запуском**

**call some\_script.cmd some params**

**@echo Эта строка выполнится после завершения some\_script.cmd**

Второй:

**@echo Эта строка выполнится перед запуском**

**some\_script.cmd some params**

**@echo Эта строка никогда не выполнится, т.к. управление из some\_script.cmd в вызвавший скрипт не вернется.**

**Код возврата команд**

После выполнения любой потенциально сбойной команды код возврата, свидетельствующий о степени успешности ее завершения, сохраняется в специальной псевдопеременной %errorlevel%. Нулевое значение значит, что команда была выполнена успешно, положительные значения свидетельствуют об ошибке, чем больше значение, тем серьезнее проблема.   
Значение кода возврата можно проверить с помощью специальной конструкции if [not] errorlevel число (см. **[if /?](ms-its:C:\\WINDOWS\\Help\\ntcmds.chm::/if.htm)**).   
Также, переменную %errorlevel% можно свободно использовать непосредственно.   
Например, "goto answer%errorlevel%" или "echo %errorlevel%".

Команда **[exit](ms-its:C:\\WINDOWS\\Help\\ntcmds.chm::/exit.htm)** число прерывает исполнение текущего сценария с возвратом запустившему его процессу заданного кода выхода. По умолчанию производится прерывание сразу всей цепочки вызовов пакетных сценариев. Если необходимо завершить только текущий сценарий и возвратить управление в сценарий, вызвавший его командой **call**, то нужно использовать ключ /b команды **exit**. Для выхода из сценария с передачей вызвавшему процессу текущего значения %errorlevel% необходимо использовать команду goto :eof.

Некоторые команды всегда устанавливают переменную %errorlevel%, некоторые модифицируют ее только при возникновении ошибок, а некоторые вообще ее никогда не меняют.   
Вот пример:

**md testdir**

**@echo %errorlevel%=0, успешно**

**rd testdir**

**@echo %errorlevel%=0, успешно**

**тест**

**@echo %errorlevel%=9009, недопустимая команда**

**md testdir**

**@echo %errorlevel%=0, успешно**

**md testdir**

**@echo %errorlevel%=1, каталог уже существует**

**rd testdir**

**@echo %errorlevel%=1, успешно, но сохранился код предыдущей ошибки**

**set errorlevel=0**

**@echo %errorlevel%=0, после явного сброса**

**rd con**

**@echo %errorlevel%=1, недопустимое имя, но оказывается, что rd errorlevel вообще никогда не меняет**

**@goto :eof**

**Копирование**

Команда **[copy](ms-its:C:\\WINDOWS\\Help\\ntcmds.chm::/copy.htm)** умеет копировать и склеивать файлы.   
Команда **[xcopy](ms-its:C:\\WINDOWS\\Help\\ntcmds.chm::/xcopy.htm)** умеет копировать файлы и каталоги.   
Команда **[ren (rename)](ms-its:C:\\WINDOWS\\Help\\ntcmds.chm::/rename.htm)** умеет переименовывать файлы и каталоги.

Проверить, удачно ли выполнилась команда можно по коду возврата %errorlevel%.

**Использование нескольких команд и символов условной обработки**

Можно выполнять несколько команд из одной командной строчки с помощью символов условной обработки, перечисленных в следующей таблице.

|  |  |  |
| --- | --- | --- |
| **Символ** | **Синтаксис** | **Определение** |
| **&** [...] | *команда1* **&** *команда2* | Выполняется первая команда, затем вторая команда. |
| **&&** [...] | *команда1* **&&** *команда2* | Вторая команда выполняется, только если первая была выполнена успешно. |
| **||** [...] | *команда1* **||** *команда2* | Вторая команда выполняется, только если первая не была выполнена (полученный код ошибки превышает ноль). |
| **( )** [...] | **(***команда1* **&** *команда2***)** | Используется для группировки или вложения команд. |
| **;** или **,** | *команда1* *параметр1***;***параметр2* | Используется для разделения параметров команды. |

Примечание:  
Амперсанд &, вертикальная черта | и скобки () являются специальными символами, которым должен предшествовать управляющий символ ^ или кавычки, если эти символы передаются в качестве аргументов.

Вот пример специализированного скрипта для сравнения файлов:

**@echo Производится сравнение файлов %1 и %2 с особой передачей кода возврата**

**fc /w /c /l %1 %2 >nul && @echo Файлы совпадают ^(с точностью до пробелов и регистра символов^)**

**if not %~z1==%~z2 (@echo Размеры файлов не равны) & exit /b 2**

**fc /b %1 %2 >nul || ((@echo Файлы побайтно не совпадают) & exit /b 1)**

**exit /b 0**

**Использование переменных среды**

В пакетных файлах можно использовать своеобразные переменные, которые задаются в виде ключ=*значение*.   
В дальнейшем при обработке команд каждое вхождение подстроки %ключ% заменяется на *значение*.   
Переменные среды бывают трех видов.

Глобальные переменные устанавливаются в свойствах операционной системы или специальной внешней утилитой **setx** и доступны с одними и теми же значениями при запуске всех пакетных файлов.

Локальные переменные устанавливаются и используются по-своему в каждом отдельном сценарии, сохраняются только во время его выполнения, а также передаются в дочерние сценарии при вложенных вызовах.

Псевдопеременные, такие как %CD%, %DATE%, %TIME%, %RANDOM%, %ERRORLEVEL%, обрабатываются особенным образом в зависимости от определенных условий.

В следующей таблице приведен выборочный список наиболее интересных переменных среды.

|  |  |  |
| --- | --- | --- |
| **Переменная** | **Тип** | **Описание** |
| %CD% | Локальная | Возвращает путь к текущей папке. |
| %COMPUTERNAME% | Системная | Возвращает имя компьютера. |
| %DATE% | Системная | Возвращает текущую дату. Использует тот же формат, что и команда **date /t**. |
| %ERRORLEVEL% | Системная | Возвращает код ошибки последней использовавшейся команды. |
|  |  |  |
| %PATH% | Системная | Указывает путь поиска для исполняемых файлов. |
| %RANDOM% | Системная | Возвращает произвольное десятичное число от 0 до 32767. |
| %SYSTEMROOT% | Системная | Возвращает размещение системного каталога Windows. |
| %TEMP% и %TMP% | Системная и пользовательская | Возвращает временные папки, по умолчанию используемые приложениями, которые доступны пользователям, выполнившим вход в систему. Некоторые приложения требуют переменную TEMP, другие — переменную TMP. |
| %TIME% | Системная | Возвращает текущее время. Использует тот же формат, что и команда **time /t**. |
| %USERNAME% | Локальная | Возвращает имя пользователя, выполнившего вход в систему. |

При использовании переменных можно также использовать выделение подстрок.

|  |  |
| --- | --- |
| **Конструкция** | **Назначение** |
| **%PATH:~10,5%** | расширит действие переменной среды PATH, затем использует только 5 символов, которые начинаются с 11-го символа (пропустив 10) расширенного результата. Если длина не указана, по умолчанию используется оставшееся значение переменной. Если оба значения (длина и число пропускаемых символов) отрицательны, следующим используемым значением будет длина значения переменной среды, добавленной к указанному значению пропуска или указанной длины. |
| **%PATH:~-10%** | извлечет последние 10 символов переменной PATH |
| **%PATH:~0,-2%** | извлечет все символы переменной PATH, за исключением 2-х последних |

Для установки значений переменных используется команда **[set](ms-its:C:\\WINDOWS\\Help\\ntcmds.chm::/set.htm)**.   
Установка значений:   
**set переменная=значение**   
Показ значений:   
**set переменная**   
Показ значений всех переменных:   
**set** (без параметров)   
Проверка наличия переменной с заданным именем:   
**if defined переменная команда**

**Вычисление арифметических выражений**

Команда **[set](ms-its:C:\\WINDOWS\\Help\\ntcmds.chm::/set.htm)** поддерживает особое расширение синтаксиса:   
**set /a выражение**

В следующей таблице представлены арифметические операторы, поддерживающие параметр **/a**, в порядке убывания приоритета.

|  |  |
| --- | --- |
| **Оператор** | **Действие оператора** |
| () | группировка |
| ! ~ - | унарные операторы |
| \* / % | арифметические операторы |
| + - | арифметические операторы |
| << >> | двоичный сдвиг |
| & | двоичное И |
| ^ | двоичное исключающее ИЛИ |
| | | двоичное ИЛИ |
| = \*= /= %= += -= &= ^= |= <<= >>= | присваивание |
| , | разделитель операторов |

При использовании символов & | % < > ^ заключайте выражение в кавычки.   
Любые нечисловые строки в выражении считаются именами переменной среды, чьи значения преобразуются в числа до обработки. Если указано имя переменной среды, которое не определено в текущем окружении, ей присваивается значение нуля, что позволяет выполнять арифметические операции со значениями переменной среды без использования % для получения значения.

Пример использования вычисления выражений:

**set /a "a=!5^5, b=2, a>>=2, c=a+b, d=2\*2"**

**@echo %a%=1, %b%=2, %c%=3, %d%=4**

**Потоки ввода/вывода**

Потоки используются в командных сценариях для унифицированного распределения и перенаправления ввода/вывода.

В следующей таблице представлены стандартные потоки.

|  |  |  |
| --- | --- | --- |
| **Поток** | **Номер дескриптора** | **Описание** |
| STDIN | 0 | Стандартный ввод (только чтение) |
| STDOUT | 1 | Стандартный вывод (только запись) |
| STDERR | 2 | Вывод ошибок (только запись) |

Каждый поток может быть связан с каким-либо устройством ввода/вывода.

В следующей таблице представлены стандартные устройства.

|  |  |
| --- | --- |
| **Устройство** | **Описание** |
| **CON** | Консоль (чтение с клавиатуры и вывод на экран) |
| **NUL** | Пустота |
| **PRN** | Принтер |

В качестве устройств могут также выступать файлы, заданные по имени.   
По умолчанию все стандартные потоки связаны с устройством **CON**.

В следующей таблице описаны операторы перенаправления, применимые к потокам и устройствам.

|  |  |
| --- | --- |
| **Оператор перенаправления** | **Описание** |
| **>** | Перенаправление вывода с затиранием. |
| **<** | Перенаправление ввода. |
| **>>** | Перенаправление вывода с добавлением. |
| **>&** | Считывает данные на выходе одного потока как входные данные для другого потока. |
| **<&** | Считывает входные данные одного потока как выходные данные другого потока. |
| **|** | Считывает выходные данные одной команды и записывает их на вход другой команды. Эта процедура известна под названием "канал". |

Некоторые команды, такие как find, sort, more, называются фильтрами, т.к. они занимаются специализированной трансформацией входного потока в выходной.

Далее приведены некоторые примеры.

Типичный вызов при тестировании программистских задач.   
(Ввод из input.txt, вывод в output.txt, вывод ошибок в никуда).

**solution.exe <input.txt >output.txt 2>nul**

Вызов скрипта компиляции с перенаправлением потоков вывода и ошибок на дописывание в файл compile.log.

**call compile\_all.cmd %SOLUTION% %TARGET% >>compile.log 2>&1**

Дописывание содержимого одного файла в конец другого.

**more <report.txt >>$result$.txt**

Поэкранный вывод списка элементов текущего каталога.

**dir | more**

Поэкранный вывод имен всех фидошников города Гомеля, отсортированных по алфавиту.

**(for /f "usebackq tokens=5 delims=, " %%i in (`find /i "Gomel" R45Point`) do @echo %%i) | sort | more**

**Организация циклов**

Команда **[for](ms-its:C:\\WINDOWS\\Help\\ntcmds.chm::/for.htm)** позволяет произвести запуск некоторой команды для каждого элемента из заданного множества.   
Синтаксис:   
**for** {**%***переменная*|**%%***переменная*} **in (***множество***)** **do***команда* [*ПараметрыКоманднойСтроки*]   
Параметры:

{**%***переменная*|**%%***переменная*}

Обязательный параметр. Замещаемый параметр. Используйте **%***переменная* для выполнения команды **for** из командной строки. В пакетных файлах для команды FOR используется запись %%переменная вместо %переменная. Имена переменных учитывают регистр букв (%i отличается от %I).

**(***множество***)**

Обязательный параметр. Задает один или несколько файлов, каталогов, диапазон значений или текстовых строк, подлежащих обработке заданной командой. Скобки являются обязательными.

*команда*

Обязательный параметр. Задает команду, которая будет выполнена для каждого файла, каталога диапазона значений или текстовой строки, включенной в указанный параметр **(***множество***)**.

*ПараметрыКоманднойСтроки*

Задает параметры командной строки, которые используются с указанной командой.

Подробнее команда описана в справке. Здесь приведены несколько примеров, иллюстрирующих ее возможности.

Перебор фиксированного множества параметров.

**for %%i in (DirInfo,File\_Id.diz,Descript.ion,Read.Me,ReadMe.\*,FILES.BBS) do @set descr=%%i & @if exist %%i goto found**

**goto notfound**

**:found**

**@echo В текущем каталоге доступно описание в файле %descr%**

**:notfound**

Итерация по файлам.

**@set sz=0**

**@for /r %temp% %%i in (\*.\*) do @set /a sz+=%%~zi & @echo %%i %%~zi**

**@echo Размер временного каталога: %sz%**

Итерация по каталогам.

**@echo off**

**echo Поддерживаемые компиляторы:**

**for /d %%i in (d:\delta\lngs\\*) do @echo %%~ni**

Численная итерация.

**@for /l %%i in (1,1,5) do @echo %%i**

**@echo ...вышел зайчик погулять.**

**Итерация и разбор файлов**

Команда **[for](ms-its:C:\\WINDOWS\\Help\\ntcmds.chm::/for.htm)** может также употребляться для чтения строк из файлов и их синтаксического разбора.   
В этом случае используется следующий синтаксис:   
FOR /F ["options"] %переменная IN (набор) DO команда [параметры]   
FOR /F ["options"] %variable IN ("literal string") DO command [command-parameters]   
FOR /F ["options"] %variable IN ('command') DO command [command-parameters]   
или, если использован параметр usebackq:   
FOR /F ["options"] %variable IN (filenameset) DO command [command-parameters]   
FOR /F ["options"] %variable IN ('literal string') DO command [command-parameters]   
FOR /F ["options"] %variable IN (`command`) DO command [command-parameters]

Набор содержит имена одного или нескольких файлов, которые по очереди открываются, читаются и обрабатываются. Обработка состоит в чтении файла, разбивки его на отдельные строки текста и выделения из каждой строки заданного числа подстрок (в том числе нуля). Затем найденная подстрока используется в качестве значения переменной при выполнении основного тела цикла. По умолчанию ключ /F выделяет из каждой строки файла первое слово, очищенное от окружающих его пробелов. Пустые строки в файле пропускаются. Необязательный параметр "options" служит для переопределения заданных по умолчанию правил обработки строк.   
Возможные ключевые слова для этого параметра:

|  |  |
| --- | --- |
| **Ключевое слово** | **Описание** |
| eol=c | Задает символ конца строки (только один символ). |
| skip=n | Задает число строк, пропускаемых в начале файла. |
| delims=xxx | Задает набор разделителей. Заменяет набор разделителей по умолчанию, состоящий из пробела и символа табуляции. |
| tokens=x,y,m-n | Задает элементы, передаваемые из каждой строки в тело цикла **for** при каждой итерации. В результате размещаются дополнительные имена переменных. Форма *m-n* задает диапазон, указывающий элементы с *m*-го по *n*-ый. Если последним символом строки **tokens=** является звездочка (\*), то размещается дополнительная переменная, в которую помещается остаток строки после разбора последнего элемента. |
| usebackq | Задает возможность использования кавычек для имен файлов в параметре *МножествоИменФайлов*. Задает исполнение строки, заключенной в обратные кавычки, как команды, а строки в одиночных кавычках — как команды в символьной строке. |

Поясняющий пример:   
FOR /F "eol=; tokens=2,3\* delims=, " %i in (myfile.txt) do @echo %i %j %k   
Эта команда обрабатывает файл myfile.txt, пропускает все строки, которые начинаются с символа точки с запятой, и передает вторую и третью подстроки из каждой строки в тело цикла, причем подстроки разделяются запятыми и/или пробелами. В теле цикла переменная %i используется для второй подстроки, %j - для третьей, а %k получает все оставшиеся подстроки после третьей. Имена файлов, содержащие пробелы, необходимо заключать в двойные кавычки. Для того чтобы использовать двойные кавычки, необходимо использовать параметр usebackq, иначе двойные кавычки будут восприняты как границы строки для обработки.

Синтаксис команды FOR /F также позволяет обработать отдельную строку, с указанием параметра filenameset, заключенным в одиночные кавычки. Строка будет обработана как единая строка из входного файла.

Наконец, команда FOR /F позволяет обработать строку вывода другой команды. Для этого следует ввести строку вызова команды в апострофах вместо набора имен файлов в скобках. Строка передается для выполнения обработчику команд CMD.EXE, а вывод этой команды записывается в память и обрабатывается так, как будто строка вывода взята из файла. Например, следующая команда:   
FOR /F "usebackq delims==" %i IN (`set`) DO @echo %i   
выведет перечень имен всех переменных среды, определенных в настоящее время в системе.

Еще один пример:

**@echo Разбор $result$.txt**

**for /f "usebackq delims=" %%i in ($result$.txt) do (@set val=%%i) & @goto next**

**:next**

**@set /a nval=val+0**

**if not "%val%"=="%nval%" (@echo Не число!) & @goto :eof**

**if %val% lss 0 (@echo Меньше нуля!) & @goto :eof**

**@echo Начислено %val% баллов.**

**Организация блокировок**

Иногда бывает необходимо обеспечить, чтобы какой-либо скрипт мог выполняться только в единственном экземпляре. Для этого нужно, чтобы скрипт мог за одну элементарную операцию одновременно попытаться установить блокировку и узнать, удалось ли это. На роль такой операции подходит команда создания каталога **md** (**mkdir**). При таком подходе наличие блокировки определяется по факту существования определенного каталога, а успех выполнения команды **md**можно проверить по коду возврата (errorlevel).   
Далее приведен иллюстрирующий пример.

**@rem Устанавливаем блокировку на повторный запуск**

**set lockdir=%tmp%\some\_lock**

**md %lockdir%**

**if not errorlevel 1 goto :work**

**@echo Aborting because of lock: %lockdir%**

**@echo (Other instance of %~nx0 has not finished yet)**

**@goto :eof**

**:work**

**@rem some work...**

**@rem Снимаем блокировку**

**rd %lockdir%**

**Установка цвета консоли**

Команда **[color](ms-its:C:\\WINDOWS\\Help\\ntcmds.chm::/color.htm)** позволяет менять цвет консоли.   
Например, "color 0A". Первый символ соответствует цвету фона, второй - текста.

0 = Черный 8 = Серый

1 = Синий 9 = Светло-синий

2 = Зеленый A = Светло-зеленый

3 = Голубой B = Светло-голубой

4 = Красный C = Светло-красный

5 = Лиловый D = Светло-лиловый

6 = Желтый E = Светло-желтый

7 = Белый F = Ярко-белый

Цвет создаваемой консоли позволяет менять также опция **/t** команды **[cmd](ms-its:C:\\WINDOWS\\Help\\ntcmds.chm::/Cmd.htm)**.