Оглавление

ВВЕДЕНИЕ

Цель проведения практикумов — выработка практических навыков работы с операционными системами (ОС) посредством командной строки, а также взаимодействие с инструментами проектирования, создания и сопровождения информационных систем, основанных на технологиях баз данных.

В качестве системы управления базами данных (СУБД) для проведения компьютерных практикумов используется среда Oracle 11gR2 XE, каждому обучающемуся должен быть предоставлен индивидуальный доступ к собственному экземпляру учебной базы данных. Это могут быть локально установленные в компьютерном классе экземпляры Oracle 11gR2 XE с выделенными под каждого обучающегося схемами или обучающийся может получить бесплатный доступ к облачной платформе Oracle (URL: http://apex.oracle.com) и там создать свой экземпляр учебной базы.

Каждый компьютерный практикум состоит из двух частей: сначала преподаватель объясняет тему занятия, демонстрирует примеры на собственном экземпляре учебной базы данных, затем обучающиеся выполняют задания для самостоятельной работы, используя собственные экземпляры учебной базы данных.

1. СРЕДСТВА РАБОТЫ С КОМАНДНОЙ СТРОКОЙ В WINDOWS

1.1. Командная строка в Windows

Консоль командной строки присутствует во всех версиях операционных систем Windows. Ранние версии ОС поддерживали режим MS-DOS напрямую, что позволяло выполнять простые команды прямо из консоли. Представители семейства NT, такие как Windows XP, Windows 7 или Windows Server 2008, работают по другим принципам, однако MS-DOS в них тоже поддерживается, но через виртуальную машину (NT Virtual DOS Machine, NTVDM), что позволяет контролировать и администрировать системные ресурсы прямо из консоли командного режима.

Для работы с командной строкой есть встроенный **интерпретатор команд**, который используется для выполнения вводимых с клавиатуры команд. В технической литературе, посвященной работе с командной строкой, можно встретить другие названия интерпретатора команд, такие как *«командный интерпретатор»*, *«командный процессор»*, *«командная строка»*, *«командная оболочка»*.

При стандартной установке командный интерпретатор хранится на диске под именем ***cmd.exe*** в папке ***\Windows\System32***. Размер файла в Windows 7 равен 295 Кб, в Windows XP SP3 — 387 Кб.

Значимость командной строки состоит в том, что некоторые возможности ОС Windows доступны только при использовании командной строки. Кроме того, ключи командной строки перекрывают параметры Реестра Windows.

1.2. Перенаправление стандартных ввода, вывода и ошибок

По умолчанию команды получают ввод из параметров, указываемых при вводе команды в командной строке, и направляют свой вывод, включая и сообщения об ошибках, на экран монитора. Однако иногда возникают ситуации, в которых ввод нужно получить не с клавиатуры, а из другого источника, а вывод направить в файл или на другое устройство вывода, например на принтер. Кроме того, сообщения об ошибках иногда желательно направлять в файл, а не в окно консоли. Для перенаправления ввода и вывода используется синтаксис, представленный в табл. 1.

Таблица 1

Синтаксис ввода, вывода

Синтаксис перенаправления	Описание
Команда1 | Команда2	Вывод первой команды служит вводом для второй
Команда < [путь]имя_файла	Ввод команды поступает из заданного файла
Команда > [путь]имя_файла	Вывод команды направляется в заданный файл. При этом, если указанный файл не существует, то он создается, в противном случае — перезаписывается
Команда >> [путь]имя_файла	Вывод команды направляется в заданный файл. При этом, если указанный файл не существует, то он создается, в противном случае вывод дописывается в конец файла
Команда < [путь]имя_файла1 > [путь]имя_файла2	Ввод команды поступает из заданного первого заданного файла, а вывод направляется во второй с перезаписью
Команда < [путь]имя_файла1 >> [путь]имя_файла2	Ввод команды поступает из заданного первого заданного файла, а вывод дозаписывается во второй файл
Команда 2> [путь]имя_файла	Создается заданный файл, в который направляется вывод сообщений об ошибках. Если такой файл уже существует, то он перезаписывается
Команда 2>&1 [путь]имя_файла	Сообщения об ошибках и стандартный вывод записываются в один и тот же файл

Перенаправление вывода команды в качестве ввода другой команды принято называть *конвейеризацией.* Вывод можно последовательно перенаправлять неоднократно. Чаще всего конвейеризация используется для двух команд — FIND и MORE.

Команда FIND ищет строки в файлах или в тексте ввода и выводит строки, соответствующие условию, которое задается в виде подстроки, заключенной кавычками. Например, чтобы получить список всех файлов с расширением .bmp в каталоге c:\windows, можно воспользоваться следующим вариантом перенаправления и обработки ввода:

dir c:\windows | find ".bmp"

```
D:\>dir c:\windows | find ".bmp"
15.04.2008  16:00             1a272 Голубые кружева 16.bmp
15.04.2008  16:00            26a582 Зеленый камень.bmp
15.04.2008  16:00            17a062 Кофейня.bmp
15.04.2008  16:00            17a336 На рыбалку.bmp
15.04.2008  16:00            65a954 Ночной ковыль.bmp
15.04.2008  16:00             9a522 Паркет.bmp
15.04.2008  16:00            65a978 Пузыри.bmp
15.04.2008  16:00            17a362 Рододендрон.bmp
15.04.2008  16:00            16a730 Сиреневый пух.bmp
15.04.2008  16:00            65a832 Штукатурка.bmp
15.04.2008  16:00            26a680 Японский мотив.bmp

D:\>
```

Команда MORE принимает вывод других команд и разбивает его на части, каждая из которых умещается в окне консоли. Например, если нужно просмотреть список всех файлов с расширением .dll каталога c:\windows\system32, можно воспользоваться командами FIND и MORE следующим образом:

dir c:\windows\system32 | find ".dll" | more

После полного заполнения экрана информацией вывод приостанавливается, и затем после нажатия клавиши ENTER на экран будет выводиться одна новая строка.

```
C:\Documents and Settings\lektor>dir c:\windows\system32 | find ".dll" | more
12.02.2010  08:35           100a864 6to4svc.dll
15.04.2008  16:00            26a112 aaaamon.dll
15.04.2008  16:00           136a192 aaclient.dll
07.03.2002  01:00           287a256 AbaleZip.dll
15.04.2008  16:00            66a048 acctres.dll
15.04.2008  16:00           130a560 acledit.dll
15.04.2008  16:00           117a760 aclui.dll
16.10.2006  20:49            17a440 acrotls.dll
15.04.2008  16:00           193a536 activeds.dll
15.04.2008  16:00            98a304 actxprxy.dll
08.03.2009  05:32            72a704 admparse.dll
15.04.2008  16:00            26a112 adptif.dll
15.04.2008  16:00           175a616 adsldp.dll
15.04.2008  16:00           143a360 adsldpc.dll
15.04.2008  16:00            68a096 adsmsext.dll
15.04.2008  16:00           162a304 adsnds.dll
15.04.2008  16:00           263a680 adsnt.dll
-- More  --
```

```
15.04.2008  16:00            162a304 adsnds.dll
15.04.2008  16:00            263a680 adsnt.dll
15.04.2008  16:00            123a392 adsnw.dll
-- More  --
```

Подробную справку о возможностях использования рассмотренных команд можно получить, введя в командной строке **find/?** или **more/?**.

Один из эффективных и часто используемых методов перенаправления — получение входных данных для команды из файла и помещение вывода команды в файл.

Например, вывод списка файлов каталога d:\test\data в файл d:\test\list.txt можно оформить следующей командой:

dir d:\test\data > d:\test\list.txt

Поиск файлов, имена которых начинаются с символов pr, в списке файлов, хранящемся в файле c:\test\list.txt, организуется следующим образом:

find “pr” d:\test\list.txt

```
C:\Documents and Settings\lektor>dir d:\test\data > d:\test\list.txt

C:\Documents and Settings\lektor>copy d:\test\list.txt con
 Том в устройстве D не имеет метки.
 Серийный номер тома: 8649-530E

 Содержимое папки d:\test\data

10.11.2013  16:56    <DIR>          .
10.11.2013  16:56    <DIR>          ..
10.11.2013  16:56                10 primer.bat
10.11.2013  16:55                 0 proba.bat
10.11.2013  16:56                10 test.bat
               3 файлов             20 байт
               2 папок  26a069a512a192 байт свободно
Скопировано файлов:         1.

C:\Documents and Settings\lektor>find "pr" d:\test\list.txt

---------- D:\TEST\LIST.TXT
10.11.2013  16:56                10 primer.bat
10.11.2013  16:55                 0 proba.bat

C:\Documents and Settings\lektor>_
```

Поставленная выше задача решается при помощи организации конвейерной передачи вывода команды **dir** на вход команды **find**:

```
C:\Documents and Settings\lektor>dir d:\test\data | find "pr"
10.11.2013  16:56                10 primer.bat
10.11.2013  16:55                 0 proba.bat
```

Ниже представлен еще один вариант решения поставленной задачи, включающий промежуточное создание файла с выводом команды **dir**:

```
D:\>dir d:\test\data > d:\test\list.txt | find d:\test\list.txt "pr"

---------- D:\TEST\LIST.TXT
10.11.2013  16:56                10 primer.bat
10.11.2013  16:55                 0 proba.bat

D:\>
```

1.3. Создание цепочек и группирование команд

Как правило, при работе с командной строкой пользователь вводит текст команды и нажатием клавиши ENTER запускает ее на выполнение. Однако командная строка позволяет выполнять не только одиночные команды, но и список команд, используя специальные управляющие символы “&” и “|”. Пользователю предоставляется возможность создавать цепочку команд и выполнять их последовательно, а также определять условия выполнения команд в зависимости от успеха или неудачи выполнения предыдущих команд. Можно группировать наборы команд, выполняемых по условию. В табл. 2 представлены основные схемы для создания цепочек и группирования команд.

Таблица 2

Группирование и цепочки команд

Символ	Синтаксис	Описание
&	Команда1 & Команда2	Последовательно выполняются команды Команда1 и Команда2
&&	Команда1 && Команда2	Команда2 выполняется, если Команда1 выполнена успешно
||	Команда1 || Команда2	Команда2 выполняется, если Команда1 не выполнена успешно
()	(Команда1 & Команда2) && (Команда3)	Команды Команда1 и Команда2 объединяются в группу, а Команда3 выполняется в случае успешного завершения этих команд
	(Команда1 & Команда2) || (Команда3)	Команды Команда1 и Команда2 объединяются в группу, а Команда3 выполняется в случае неудачного завершения этих команд

Цепочки команд используются, когда для решения задачи команды нужно выполнить в определенной последовательности. Пусть требуется перейти в определенный каталог и получить список файлов, отсортированный по дате. Использование цепочки позволяет решить эту задачу, введя всего одну строку:

cd c:\test\dir_cont & dir /o:d

Иногда требуется выполнить какую-либо операцию, если предыдущая операция потерпела неудачу. Пусть есть группа рабочих станций, на части которых существует каталог c:\test\data, а на других — каталог c:\data. Необходимо обеспечить возможность копирования всех данных из каталога docs внешнего накопителя D: в каталог data, независимо от конфигурации рабочей станции. Используя механизм цепочек команд, решить поставленную задачу можно следующим образом:

cd c:\test\data || cd c:\data & copy d:\docs*.*

Выполняя указанную цепочку команд, система попытается сначала перейти в каталог c:\test\data, если такого каталога нет, — в каталог c:\data. Затем, независимо от того, какой каталог станет текущим, система скопирует в него все файлы из каталога d:\docs.

Часто возникает ситуация, когда запуск последующей команды зависит от того, как завершилась предыдущая команда — успешно или неудачно. Пусть требуется переместить файл test.bat из каталога c:\test в каталог d:\arhiv, только если указанный файл существует. Задача может быть решена вводом следующей строки:

dir c:\test\test.bat && move c:\test\test.bat d:\arhiv

Группирование требуется при выполнении нескольких команд, чтобы избежать конфликтов между ними, обеспечить правильный порядок их выполнения и объявления вывода нескольких команд общим при помещении результатов в файл. Для группирования команд используются скобки.

Рассмотрим пример. Предположим, что нужно поместить в файл info.txt сведения об имени вычислительной системы и об использующейся ОС. Для этих целей воспользуемся следующей конструкцией:

hostname & ver > info.txt

Однако при выполнении команд в файл info.txt попадет только информация об операционной системе, а имя компьютера будет выдано на экран. Это вполне объяснимо. Команды выполняются последовательно, для первой команды стандартный вывод не переопределен и направляется на экран. Для второй команды стандартный вывод перенаправлен в файл. Чтобы в файл попадал вывод обеих команд, их нужно сгруппировать:

(hostname & ver) > info.txt

Теперь в файл indo.txt попала вся необходимая информация.

1.4. Основы сценариев командной строки

Рассмотренные выше возможности работы с командной строкой широко используются при создании ее сценариев.

Сценарии командной строки — текстовые файлы с командами, которые необходимо выполнять последовательно, часто в автоматическом режиме. Сценарии можно создавать и редактировать подобно любому текстовому файлу, используя текстовые редакторы, например Блокнот. Каждая команда или группа команд, которые нужно выполнять совместно, должны размещаться в отдельной строке. Командная строка не требует специального символа завершения помимо символа конца строки. Файл, в котором сохраняется сценарий командной строки, должен иметь расширение **.bat** или **.cmd.**

При создании сценариев командной строки очень часто используются шесть простейших команд: **cls**, **rem**, **echo**, **@**, **title**, **color**.

Рассмотрим подробнее назначение этих команд.

Команда **cls** очищает консольное окно и перемещает курсор в верхний левый угол экрана. При этом весь текст в буфере экрана тоже очищается.

Команда **rem** позволяет добавлять в сценарий строки комментариев. Текст комментария помещается через пробел после имени команды.

Также команда **rem** может быть использована для предотвращения выполнения команды или группы команд. В этом случае достаточно поместить команду **rem** в начало строки.

Команда **echo** служит двум целям: записи текста в вывод и включения/выключения эхо-отображения команд. Обычно при выполнении команд сценария сами команды и вывод этих команд отображаются в консольном окне. Это называется *эхо-отображением команд*. Чтобы отключить эхо-отображение, нужно ввести команду **echo off**. Чтобы узнать, включено ли эхо-отображение команд или нет, достаточно ввести команду **echo**.

Для возобновления эхо-отображения используется команда **echo on**. Чтобы использовать команду **echo** для отображения текста, нужно указать текст после пробела после команды.

Для того чтобы вывести пустую строку, нужно сразу после команды поставить точку.

При этом **между командой и точкой пробела быть не должно**!

Команда **@** предотвращает эхо-отображение одной текущей строки. Эту возможность команды **@** используют для отключения эхо-отображения команды **echo off**.

Команда **title** позволяет отобразить в заголовке окна консоли команд любого текста. Этой возможностью можно пользоваться для отображения хода выполнения сценария.

Команда **color** позволяет изменять цвета фона и текста окна консоли непосредственно при выполнении сценария. По умолчанию консольное окно отображает белый текст на черном фоне. Изменить цвета можно, указав в качестве параметра команды **color** двухразрядный шестнадцатеричный код, первая цифра которого определяет цвет фона, а вторая — цвет текста. В табл. 3 представлены значения кодов цветов окна командной оболочки.

Таблица 3

Значения кодов цветов

Код	Цвет		Код	Цвет	
0	Black	Черный	8	Gray	Серый
1	Blue	Синий	9	Bright Blue	Ярко-синий
2	Green	Зеленый	A	Bright Green	Салатовый
3	Aqua	Бирюзовый	B	Bright Aqua	Голубой
4	Red	Красный	C	Bright Red	Ярко-красный
5	Purple	Фиолетовый	D	Bright Purple	Ярко-фиолетовый
6	Yellow	Желтый	E	Bright Yellow	Ярко-желтый
7	White	Белый	F	Bright White	Ярко-белый

При запуске сценария на выполнение можно передать сценарию необходимую информацию путем задания значения аргументов. Каждое значение, передаваемое сценарию, задает значение одного из формальных параметров. Имя самого сценария сохраняется в параметре %0. Значение первого аргумента сохраняется в параметре %1, второго — в параметре %2 и т.д. до %9 — для девятого аргумента. Если при вызове сценария указывается более 9 аргументов, то дополнительные аргументы не теряются, а сохраняются в специальном параметре %*. Получить доступ к дополнительным аргументам можно с помощью команды **shift**. Если команда **shift** используется без аргумента, то значения параметров сценария сдвигаются на 1, т.е. значение параметра %1 заменяется значением параметра %2 и т.д.

При необходимости можно указать, с какого параметра начинается сдвиг, обозначив номер первого из заменяемых параметров в качестве аргумента команды **shift**. Например, если указать **shift /2**, то значение параметра %2 будет заменено значением параметра %3 и т.д. Значения параметров %0 и %1 при этом останутся без изменения.

Наряду с параметрами в сценариях командной строки можно использовать переменные, которые принято называть *переменными окружения* или *переменными среды*. Переменные окружения бывают двух типов: *встроенные системные* и *встроенные пользовательские.*

Встроенные системные переменные являются ресурсами ОС или формируются драйверами аппаратуры. Такие переменные доступны всем Windows-процессам, даже если никто не вошел в систему в интерактивном режиме. Значения встроенных системных переменных выбираются из реестра Windows.

Встроенные пользовательские переменные создаются при входе в систему какого-либо пользователя и **существуют только в течение рабочего сеанса пользователя**.

Список всех переменных среды, доступных в текущем экземпляре командной строки, можно получить с помощью команды **set.**

Команда имеет следующий формат:

set [переменная=[строка]],

где переменная — имя переменной среды;

строка — строка символов, присваиваемая указанной переменной.

Имена переменных не чувствительны к регистру и могут содержать латинские буквы, цифры и практически все символы клавиатуры, кроме символов, зарезервированных в командной строке: @ < > & | ^.

На практике принято присваивать переменным информативные имена, например:

System_name

Широкое распространение получила стандартная схема наименования переменных, в соответствии с которой имя переменной, состоящее из нескольких слов, пишется слитно, причем первая буква первого слова является строчной, а первые буквы каждого последующего слова являются прописными, например:

userName

В отличие от многих языков программирования, командная строка игнорирует типы данных. **Значения всех переменных хранятся как символьные строки**.

Для доступа к значениям переменных используются два варианта метода подстановки.

В первом случае переменная сравнивается с каким-либо действительным значением:

if %errorlevel%=="2" echo "Код возврата равен 2"

В приведенном примере определяется, содержит ли переменная *errorlevel* значение 2, при положительном результате отображается текст сообщения. Символы процента, окружающие имя переменной, предписывают сравнивать с символьным значением «2» значение переменной, а не ее имени.

Второй способ подстановки заключается в замене переменной ее реальным значением. Например, команда перехода в каталог **system32**, расположенный в корневом системном каталоге, может иметь следующий вид:

cd %systemroot%\system32,

где *systemroot* — это имя встроенной системной переменной, хранящей имя корневого системного каталога Windows.

Подстановку можно использовать и при присвоении значений переменным, например:

systemPath = %SystemRoot%\System32

Изменения значений переменных в командной оболочке с помощью команды **set** локализованы, т.е. применяются только к текущему экземпляру командной оболочки и недоступны другим системным процессам. После выхода из командной оболочки, в которой были созданы переменные, они перестают существовать.

Чтобы удалить переменную без выхода из командной строки, достаточно присвоить переменной пустое значение:

set working_dir =

Обычно командная оболочка Windows исполняет сценарии построчно от начала файла до конца. Порядок выполнения команд можно изменить с помощью команды **if,** команды безусловной передачи управления **goto** и команды вызова одного сценария из другого **call.**

Ветвление по условию позволяет организовать команда **if**, синтаксис которой приведен ниже:

if [not] условие (оператор1) [else (оператор2)]

Оператор1 и оператор2 могут быть одной командой или несколькими объединенными в цепочку, конвейер или группу. Условие — это любое выражение, возвращающее булево значение истина или ложь.

Команда работает так: если условие возвращает истина, выполняется Оператор1. В противном случае выполняется Оператор2, если он указан.

Пример

if %OS%==“Windows_NT” echo “Windows NT” else echo “Not Windows NT”

По правилам командной строки после каждого условия указывается только одна команда. Если же нужно выполнить не одну команду, а несколько, то необходимо воспользоваться конвейеризацией, созданием цепочки или группировкой команд:

if not errorlevel 0 (echo Обнаружена ошибка) & (exit)

Для проверки наличия переменных используется команда **if [not] defined.**

Пример

if defined kolServers (echo Количество серверов: % kolServers %)

Если переменная kolServers задана, то в вывод заносится информация о количестве серверов, в противном случае сценарий переходит к следующей команде.

Этот способ передачи управления обычно используется, когда требуется не выполнять достаточно большой фрагмент сценария или организовать бесконечный цикл.

Передача управления выполняется командой **goto**, параметром которой является метка, на которую нужно выполнить переход. Чтобы создать метку, следует на отдельной строке ввести двоеточие и имя метки. После передачи управления командой **goto** выполнение сценария продолжается с команды, следующей за указанной меткой.

Пример создания бесконечного цикла:

:start
.
goto start

Пример обхода фрагмента сценария:

goto start
.
: start

Иногда при выполнении пакетного файла возникает необходимость запустить другой пакетный файл. Причем в некоторых случаях выполнение основного пакетного файла должно быть приостановлено, пока выполняется вспомогательный файл, а в других вспомогательный файл должен работать вместо основного.

Для примера создадим два bat-файла. Один файл с именем 1.bat содержит следующие команды:

```
@echo off
cls
echo работает файл 1.bat
call 2.bat
echo снова работает файл 1.bat
```

Во второй файл с именем 2.bat поместим команды:

```
@echo off
echo запущен файл 2.bat
echo файл 2.bat свою работу закончил
```

Теперь запустим файл 1.bat. В результате на экране появится последовательность сообщений:

```
работает файл 1
запущен файл 2.bat
файл 2.bat свою работу закончил
снова работает файл 1
C:\bat>_
```

Таким образом, вызов из одного пакетного файла другого при помощи команды **call** останавливает исполнение пакетного файла до тех пор, пока не завершится выполнение пакетного файла, вызванного командой **call**. При этом после завершения вызываемого файла управление возвращается в исходный файл на команду, непосредственно следующую за командой **call**.

Теперь изменим файл 1.bat , заменив команду call простым указанием имени файла 2.bat:

```
@echo off
cls
echo работает файл 1.bat
2.bat
echo снова работает файл 1.bat
```

Снова запустим файл 1.bat на выполнение и получим результат:

```
работает файл 1
запущен файл 2.bat
файл 2.bat свою работу закончил
C:\bat>_
```

Видно, что возврата в исходный файл в этом случае не произошло.

1.5. Операторы циклов в командной строке

Довольно часто необходимо выполнить какое-либо действие для группы файлов. Для этих целей используется команда **for**, которую можно использовать и в сценариях, и в командной строке.

В первом случае используется синтаксис:

for %%переменная in (множество) do команда,

а во втором случае:

for %переменная in (множество) do команда

Параметр %%переменная или %переменная представляет подставляемую переменную. Команда **for** заменяет эту переменную текстом каждой строки в заданном множестве, пока команда после ключевого слова do не обработает все файлы. «Множество» задает один или более файлов или текстовых строк, которые необходимо обработать с помощью заданной команды (скобки обязательны), «команда» — команду, выполняемую для каждого включенного в множество файла, «параметры» — параметры данной команды (если она их использует). in и do — это не параметры, а обязательные ключевые слова команды **for**.

Ниже представлен пример использования команды **for** для последовательного вывода на экран монитора содержимого всех пакетных файлов каталога bat диска c:

```
@echo off
cls
for %%f in (*.bat) do copy %%f con
```

Если включена поддержка расширенной обработки команд, то командой **for** можно обрабатывать не только файлы, но и каталоги. В этом случае команда **for** используется с ключом **/d.**

Пример. Необходимо вывести на экран список всех подкаталогов первого уровня корневого системного каталога Windows (рис. 1).

for /d %B in (%SystemRoot%\s*) do echo %B

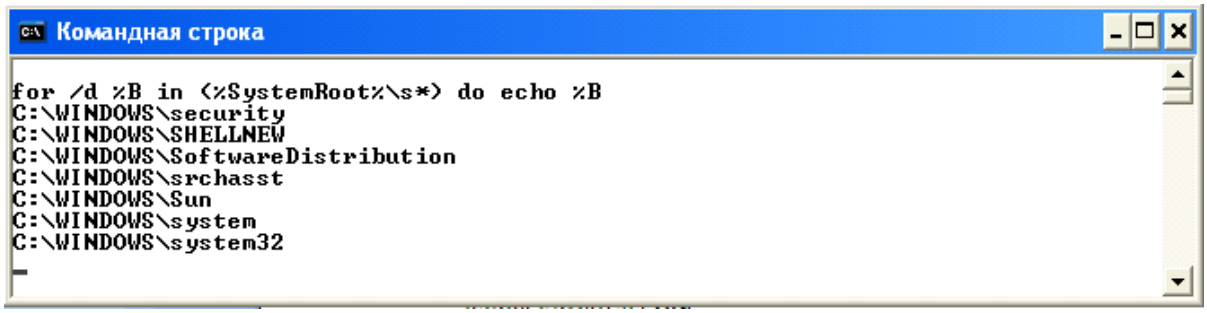

Рис. 1. Результат выполнения сценария с командой for для каталогов

Описанные выше возможности широко используются в сценариях, разрабатываемых для автоматизации администрирования вычислительной системы, работающей под управлением ОС Windows.

Задания для самостоятельной работы

Необходимо написать пакетный файл, соответствующий следующему описанию.

Пакетный файл запускается с указанием параметра, значение которого проверяется на совпадение со строкой IFO.

Если параметр при запуске пакетного файла не задан, пользователю выдается сообщение о необходимости указать параметр и после нажатия любой клавиши работа пакетного файла заканчивается.

Если указанное пользователем значение параметра не совпадает со строкой IFO, пользователю выдается сообщение вида "*Нет заданий для студентов специальности* <.....>". И после нажатия любой клавиши работа пакетного файла заканчивается.

Если параметр указан и его значение совпало с контрольной строкой, на экран выводится список курсов с 1-го по 3-й, и пользователю предлагается указать курс, на котором он обучается.

Если номер курса не равен 2, то пользователю выдается сообщение вида "*Для студентов 1-го курса заданий нет*", и после нажатия любой клавиши вновь выводится экран с запросом номера курса.

При вводе значения, содержащего курс 2, необходимо осуществить следующие действия и проверки:

– если в текущем каталоге нет каталога ARHIV, то его нужно создать;

– поставить в соответствие каталогу ARHIV логический диск E:;

– если в каталоге ARHIV есть файл, имя и тип которого совпадают с именем и типом выполняющегося пакетного файла, то файл в архиве должен быть переименован с заменой типа файла-копии на *bak*. При этом все существующие в каталоге ARHIV файлы с типом *bak* должны быть предварительно удалены;

– текущий пакетный файл должен быть скопирован в каталог ARHIV с сохранением имени и типа. При выполнении операций копирования на экран не должно выводиться системных сообщений.