Лабораторная работа №2

Сервер сценариев Windows Script Host

Теоретические сведения

С помощью Windows Script Host (WSH), предназначен для выполнения сценариев, написанные, в принципе, на любом языке (при условии, что для этого языка установлен соответствующий модуль (scripting engine), поддерживающим технологию ActiveX Scripting. В качестве стандартных языков поддерживаются Visual Basic Script Edition (VBScript) и JScript.
WSH предъявляет минимальные требования к объему оперативной памяти, и является очень удобным инструментом для автоматизации повседневных задач пользователей и администраторов операционной системы Windows. Используя сценарии WSH, можно непосредственно работать с файловой системой компьютера, а также управлять работой других приложений (серверов автоматизации). При этом возможности сценариев ограничены только средствами, которые предоставляют доступные серверы автоматизации.

1.1 Создание и запуск простейших сценариев WSH

Простейший WSH-сценарий, написанный на языке JScript или VBScript — это обычный текстовый файл с расширением js или vbs соответственно, создать его можно в любом текстовом редакторе, способном сохранять документы в формате "Только текст".
Размер сценария может изменяться от одной до тысяч строк, предельный размер ограничивается лишь максимальным размером файла в соответствующей файловой системе.
В качестве первого примера создадим JScript-сценарий, выводящий на экран диалоговое окно с надписью "Привет!". Для этого достаточно с помощью, например, стандартного Блокнота Windows (notepad.exe) создать файл First.js, содержащий всего одну строку:

```
WScript.Echo("Привет!");
```

Тот же самый сценарий на языке VBScript, естественно, отличается синтаксисом и выглядит следующим образом:

```
WScript.Echo "Привет!"
```

1.2 Запуск сценария из командной строки в консольном режиме

Можно выполнить сценарий из командной строки с помощью консольной версии WSH cscript.exe. Например, чтобы запустить сценарий, записанный в файле C:\Script\First.js, нужно загрузить командное окно и выполнить в нем команду

```
cscript C:\Script\First.js
```

Сценарий можно выполнить из командной строки с помощью (оконной) графической версии WSH wscript.exe. Для нашего примера в этом случае нужно выполнить команду

```
wscript C:\Script\First.js
```

1.3 Установка и изменение свойств сценариев

В случае необходимости для сценариев можно задавать различные параметры, влияющие на ход их выполнения. Для консольной (cscript.exe) и графической (wscript.exe) версий сервера сценариев эти параметры задаются по-разному.
Если сценарий запускается в консольном режиме, то его исполнение контролируется с помощью параметров командной строки для cscript.exe (см. табл. 1.1), которые включают или отключают различные опции WSH (все эти параметры начинаются с символов "//").

Таблица 1.1. Параметры командной строки для cscript.exe	
Параметр	Описание
`//I`	Выключает пакетный режим (по умолчанию). При этом на экран будут выводиться все сообщения об ошибках в сценарии
`//B`	Включает пакетный режим. При этом на экран не будут выводиться никакие сообщения
`//T:nn`	Задает тайм-аут в секундах, т. е. сценарий будет выполняться nn секунд, после чего процесс прервется. По умолчанию время выполнения не ограничено
`//Logo`	Выводит (по умолчанию) перед выполнением сценария информацию о версии и разработчике WSH
`//Nologo`	Подавляет вывод информации о версии и разработчике WSH
`//H:CScipt` или `//H:Wscript`	Делает cscript.exe или wscript.exe приложением для запуска сценариев по умолчанию. Если эти параметры не указаны, то по умолчанию подразумевается wscript.exe
`//S`	Сохраняет установки командной строки для текущего пользователя
`//?`	Выводит встроенную подсказку для параметров командной строки
`//E:engine`	Выполняет сценарий с помощью модуля, заданного параметром engine
`//D`	Включает отладчик
`//X`	Выполняет программу в отладчике
`//Job:<JobID>`	Запускает задание с индексом `JobID` из многозадачного WS-файла (структура WS-файлов будет описана ниже)
`//U`	Позволяет использовать при перенаправлении ввода-вывода с консоли кодировку Unicode

Например, команда
cscript //Nologo C:\Script\First.js
запустит сценарий First.js без информации о версии WSH.
Если сценарий запускается в графическом режиме (с помощью wscript.exe), то свойства сценария можно устанавливать с помощью вкладки **Сценарий** (**Script**) диалогового окна, задающего свойства файла в Windows.

1.4 Собственная объектная модель WSH

`WScript`. Это главный объект WSH, который служит для создания других объектов или связи с ними, содержит сведения о сервере сценариев, а также позволяет вводить данные с клавиатуры и выводить информацию на экран или в окно Windows.
`WshArguments`. Обеспечивает доступ ко всем параметрам командной строки запущенного сценария или ярлыка Windows.
`WshNamed`. Обеспечивает доступ к именным параметрам командной строки запущенного сценария.
`WshUnnamed`. Обеспечивает доступ к безымянным параметрам командной строки запущенного сценария.
`WshShell`. Позволяет запускать независимые процессы, создавать ярлыки, работать с переменными среды, системным реестром и специальными папками Windows.
`WshSpecialFolders`. Обеспечивает доступ к специальным папкам Windows.
`WshShortcut`. Позволяет работать с ярлыками Windows.
`WshUrlShortcut`. Предназначен для работы с ярлыками сетевых ресурсов.

`WshEnvironment`. Предназначен для просмотра, изменения и удаления переменных среды.
`WshNetwork`. Используется при работе с локальной сетью: содержит сетевую информацию для локального компьютера, позволяет подключать сетевые диски и принтеры.
`WshScriptExec`. Позволяет запускать консольные приложения в качестве дочерних процессов, обеспечивает контроль состояния этих приложений и доступ к их стандартным входным и выходным потокам.
`WshController`. Позволяет запускать сценарии на удаленных машинах.
`WshRemote`. Позволяет управлять сценарием, запущенным на удаленной машине.
`WshRemoteError`. Используется для получения информации об ошибке, возникшей в результате выполнения сценария, запущенного на удаленной машине.
`FileSystemObject` обеспечивающий доступ к файловой системе компьютера.

1.5 Объект WScript

Свойства объекта `WScript` позволяют получить полный путь к использующемуся серверу сценариев (wscript.exe или cscript.exe), параметры командной строки, с которыми запущен сценарий, режим его работы (интерактивный или пакетный). Кроме этого, с помощью свойств объекта `WScript` можно выводить информацию в стандартный выходной поток и читать данные из стандартного входного потока. Также `WScript` предоставляет методы для работы внутри сценария с объектами автоматизации и вывода информации на экран (в текстовом режиме) или в окно Windows.
Отметим, что в сценарии WSH объект `WScript` можно использовать сразу, без какого-либо предварительного описания или создания, так как его экземпляр создается сервером сценариев автоматически. Для использования же всех остальных объектов нужно использовать либо метод `CreateObject`.
Свойства объекта WScript представлены в табл. 1.2.

Таблица 1.2. Свойства объекта WScript

Свойство	Описание
`Application`	Предоставляет интерфейс `Idispatch` для объекта WScript
`Arguments`	Содержит указатель на коллекцию `WshArguments`, содержащую параметры командной строки для исполняемого сценария
`FullName`	Содержит полный путь к исполняемому файлу сервера сценариев (в Windows XP обычно это C:\WINDOWS\SYSTEM32\CSCRIPT.EXE или C:\WINDOWS\SYSTEM32\WSCRIPT.EXE)
`Name`	Содержит название объекта Wscript (Windows Script Host)
`Path`	Содержит путь к каталогу, в котором находится cscript.exe или wscript.exe (в Windows XP обычно это C:\WINDOWS\SYSTEM32)
`ScriptFullName`	Содержит полный путь к запущенному сценарию
`ScriptName`	Содержит имя запущенного сценария
`StdErr`	Позволяет запущенному сценарию записывать сообщения в стандартный поток для ошибок
`StdIn`	Позволяет запущенному сценарию читать информацию из стандартного входного потока
`StdOut`	Позволяет запущенному сценарию записывать информацию в стандартный выходной поток
`Version`	Содержит версию WSH

Опишем более подробно некоторые свойства объекта `WScript`.

Свойство Arguments

В следующем примере с помощью цикла `For Each ... Next` на экран выводятся все параметры командной строки, с которыми был запущен сценарий.

```
'**************************************************************
' Имя: Args.vbs
' Язык: VBScript
' Описание: Работа с аргументами запущенного сценария
'**************************************************************
Option Explicit

Dim i,objArgs,Arg
Set objArgs = WScript.Arguments  ' Создаем объект WshArguments
For Each Arg In objArgs
  WScript.Echo Arg  ' Формируем строки со значениями аргументов
Next

WScript.Echo s    ' Выводим сформированные строки
```

Свойства StdErr, StdIn, StdOut

Доступ к стандартным входным и выходным потокам с помощью свойств `StdIn`, `StdOut` и `StdErr` можно получить только в том случае, если сценарий запускался в консольном режиме с помощью cscript.exe. Если сценарий был запущен с помощью wscript.exe, то при попытке обратиться к этим свойствам возникнет ошибка `"Invalid Handle"`.

Таблица 1.3. Методы для работы с потоками

Метод	Описание
`Read(n)`	Считывает из потока `StdIn` заданное параметром n число символов и возвращает полученную строку
`ReadAll()`	Читает символы из потока `StdIn` до тех пор, пока не встретится символ конца файла ASCII 26 (<Ctrl>+<Z>), и возвращает полученную строку
`ReadLine()`	Возвращает строку, считанную из потока `StdIn`
`Skip(n)`	Пропускает при чтении из потока `StdIn` заданное параметром `n` число символов
`SkipLine()`	Пропускает целую строку при чтении из потока `StdIn`
`Write(string)`	Записывает в поток `StdOut` или `StdErr` строку `string` (без символа конца строки)
`WriteBlankLines(n)`	Записывает в поток `StdOut` или `StdErr` заданное параметром `n` число пустых строк
`WriteLine(string)`	Записывает в поток `StdOut` или `StdErr` строку `string` (вместе с символом конца строки)

Кроме этого, с помощью методов, работающих с входным потоком StdIn, можно организовывать диалог с пользователем, то есть создавать интерактивные сценарии.

```
'**************************************************************
'* Имя: Interact.vbs
'* Язык: VBScript
'* Описание: Ввод/вывод строк в консольном режиме
'**************************************************************
Dim  s
' Выводим строку на экран
WScript.StdOut.Write "Введите число: "
' Считываем строку
```

```
s = WScript.StdIn.ReadLine
' Выводим строку на экран
WScript.StdOut.WriteLine "Вы ввели число " & s
'*************  Конец ******************************************
```

1.6 Методы объекта WScript

Объект `WScript` имеет несколько методов:

Таблица 1.4. Методы объекта WScript

Метод	Описание
`CreateObject(strProgID [, strPrefix])`	Создает объект, заданный параметром `strProgID`
`ConnectObject(strObject, strPrefix)`	Устанавливает соединение с объектом `strObject`, позволяющее писать функции-обработчики его событий (имена этих функций должны начинаться с префикса `strPrefix`)
`DisconnectObject(obj)`	Отсоединяет объект `obj`, связь с которым была предварительно установлена в сценарии
`Echo([Arg1] [, Arg2] [,…])`	Выводит текстовую информацию на консоль или в диалоговое окно
`GetObject(strPathname [, strProgID], [strPrefix])`	Активизирует объект автоматизации, определяемый заданным файлом (параметр `strPathName`) или объект, заданный параметром `strProgID`
`Quit([intErrorCode])`	Прерывает выполнение сценария с заданным параметром `intErrorCode` кодом выхода. Если параметр `intErrorCode` не задан, то объект WScript установит код выхода равным нулю
`Sleep(intTime)`	Приостанавливает выполнения сценария (переводит его в неактивное состояние) на заданное параметром `intTime` число миллисекунд

Метод CreateObject

Строковый параметр `strProgID`, указываемый в методе `CreateObject`, называется программным идентификатором объекта (Programmic Identifier, ProgID).

Если указан необязательный параметр `strPrefix`, то после создания объекта в сценарии можно обрабатывать события, возникающие в этом объекте (естественно, если объект предоставляет интерфейсы для связи с этими событиями). Когда объект сообщает о возникновении определенного события, сервер сценариев вызывает функцию, имя которой состоит из префикса `strPrefix` и имени этого события. Например, если в качестве `strPrefix` указано `"MYOBJ_"`, а объект сообщает о возникновении события `"OnBegin,"` то будет запущена функция `"MYOBJ_OnBegin"`, которая должна быть описана в сценарии.

В следующем примере метод `CreateObject` используется для создания объекта `WshNetwork`:

```
set WshNetwork = WScript.CreateObject("WScript.Network")
```

Метод ConnectObject

Объект, соединение с которым осуществляется с помощью метода `ConnectObject`, должен предоставлять интерфейс к своим событиям.

В следующем примере в переменной `MyObject` создается абстрактный объект `"SomeObject"`, затем из сценария вызывается метод `SomeMetod` этого объекта. После этого устанавливается связь с переменной `MyObject` и задается префикс `"MyEvent"` для процедур обработки события этого объекта. Если в объекте возникнет событие с именем

"Event", то будет вызвана функция MyEvent_Event. Метод DisconnectObject объекта WScript производит отсоединение объекта MyObject.

```
var MyObject = WScript.CreateObject("SomeObject");
MyObject.SomeMethod();
WScript.ConnectObject(MyObject,"MyEvent");

function MyEvent_Event(strName){
  WScript.Echo(strName);
  }
WScript.DisconnectObject(MyObject);
```

Метод Echo

Параметры Arg1, Arg2, ... метода Echo задают аргументы для вывода. Если сценарий был запущен с помощью wscript.exe, то метод Echo направляет вывод в диалоговое окно, если же для выполнения сценария применяется cscript.exe, то вывод будет направлен на экран (консоль).

```
'*********************************************************************
'* Имя: EchoExample.vbs
'* Язык: VBScript
'* Описание: Использование метода WScript.Echo
'*********************************************************************
WScript.Echo            'Выводим пустую строку
WScript.Echo 1,2,3      'Выводим числа
WScript.Echo "Привет!"  'Выводим строку
'*************  Конец ***************************************
```

Метод Sleep

В следующем примере сценарий переводится в неактивное состояние на 5 секунд:

```
'*********************************************************************
'* Имя: SleepExample.vbs
'* Язык: VBScript
'* Описание: Использование метода WScript.Sleep
'*********************************************************************
WScript.Echo "Сценарий запущен, отдыхаем…"
WScript.Sleep 5000
WScript.Echo "Выполнение сценария завершено"
'*************  Конец ***************************************
```

1.7 Объект WshShell

С помощью объекта WshShell можно запускать новый процесс, создавать ярлыки, работать с системным реестром, получать доступ к переменным среды и специальным папкам Windows. Создается этот объект следующим образом:

```
var WshShell=WScript.CreateObject("WScript.Shell");
```

Таблица 1.5. Свойства объекта WshShell

Свойство	Описание
CurrentDirectory	Здесь хранится полный путь к текущему каталогу (к каталогу, из которого был запущен сценарий)
Environment	Содержит объект WshEnvironment, который обеспечивает доступ к переменным среды операционной системы для Windows NT/2000/XP или к переменным среды текущего командного окна для Windows 9x
SpecialFolders	Содержит объект WshSpecialFolders для доступа к специальным папкам Windows (рабочий стол, меню Пуск (Start) и т. д.)

Опишем теперь методы, имеющиеся у объекта WshShell.

Таблица 1.6. Методы объекта WshShell

Метод	Описание
`AppActivate(title)`	Активизирует заданное параметром `title` окно приложения. Строка `title` задает название окна (например, `"calc"` или `"notepad"`) или идентификатор процесса (`Process ID, PID`)
`CreateShortcut(strPathname)`	Создает объект `WshShortcut` для связи с ярлыком Windows (расширение lnk) или объект `WshUrlShortcut` для связи с сетевым ярлыком (расширение url). Параметр `strPathname` задает полный путь к создаваемому или изменяемому ярлыку
`Environment(strType)`	Возвращает объект `WshEnvironment`, содержащий переменные среды заданного вида
`Exec(strCommand)`	Создает новый дочерний процесс, который запускает консольное приложение, заданное параметром `strCommand`. В результате возвращается объект `WshScriptExec`, позволяющий контролировать ход выполнения запущенного приложения и обеспечивающий доступ к потокам `StdIn`, `StdOut` и `StdErr` этого приложения
`ExpandEnvironmentStrings(strString)`	Возвращает значение переменной среды текущего командного окна, заданной строкой `strString` (имя переменной должно быть окружено знаками `"%"`)
`LogEvent(intType, strMessage[,strTarget])`	Протоколирует события в журнале Windows NT/2000/XP или в файле WSH.log. Целочисленный параметр `intType` определяет тип сообщения, строка `strMessage` — текст сообщения. Параметр `strTarget` может задаваться только в Windows NT/2000/XP, он определяет название системы, в которой протоколируются события (по умолчанию это локальная система). Метод `LogEvent` возвращает `true`, если событие записано успешно и `false` в противном случае
`Popup(strText,[nSecToWait], [strTitle], [nType])`	Выводит на экран информационное окно с сообщением, заданным параметром `strText`. Параметр `nSecToWait` задает количество секунд, по истечении которых окно будет автоматически закрыто, параметр `strTitle` определяет заголовок окна, параметр `nType` указывает тип кнопок и значка для окна
`RegDelete(strName)`	Удаляет из системного реестра заданный параметр или раздел целиком
`RegRead(strName)`	Возвращает значение параметра реестра или значение по умолчанию для раздела реестра
`RegWrite(strName, anyValue[,strType])`	Записывает в реестр значение заданного параметра или значение по умолчанию для раздела
`Run(strCommand, [intWindowStyle], [bWaitOnReturn])`	Создает новый независимый процесс, который запускает приложение, заданное параметром `strCommand`
`SendKeys(string)`	Посылает одно или несколько нажатий клавиш в активное окно (эффект тот же, как если бы вы нажимали эти клавиши на клавиатуре)

`SpecialFolders(strSpecFolder)`	Возвращает строку, содержащую путь к специальной папке Windows, заданной параметром `strSpecFolder`

Метод CreateShortcut

Этот метод позволяет создать новый или открыть уже существующий ярлык для изменения его свойств. Приведем пример сценария, в котором создаются два ярлыка — на сам выполняемый сценарий (объект `oShellLink`) и на сетевой ресурс (`oUrlLink`).

```
'*************************************************************
'* Имя: MakeShortcuts.vbs
'* Язык: VBScript
'* Описание: Создание ярлыков из сценария
'*************************************************************
Dim WshShell,oShellLink,oUrlLink
' Создаем объект WshShell
Set WshShell=WScript.CreateObject("WScript.Shell")
' Создаем ярлык на файл
Set oShellLink=WshShell.CreateShortcut("Current Script.lnk")
' Устанавливаем путь к файлу
oShellLink.TargetPath=WScript.ScriptFullName
' Сохраняем ярлык
oShellLink.Save

' Создаем ярлык на сетевой ресурс
Set oUrlLink = WshShell.CreateShortcut("Microsoft Web Site.URL")
' Устанавливаем URL
oUrlLink.TargetPath = "http://www.microsoft.com"
' Сохраняем ярлык
oUrlLink.Save
'*************  Конец **************************************
```

Метод Environment

Параметр `strType` задает вид переменных среды, которые будут записаны в коллекции `WshEnvironment`; возможными значениями этого параметра являются `"System"` (переменные среды операционной системы), `"User"` (переменные среды пользователя), `"Volatile"` (временные переменные) или `"Process"` (переменные среды текущего командного окна).

```
'*************************************************************
'* Имя: ShowEnvir.vbs
'* Язык: VBScript
'* Описание: Получение значений некоторых переменных среды
'*************************************************************
Dim WshShell,WshSysEnv
' Создаем объект WshShell
Set WshShell = WScript.CreateObject("WScript.Shell")
' Создание коллекции WshEnvironment
Set WshSysEnv = WshShell.Environment("SYSTEM")
WScript.Echo WshSysEnv("OS")
WScript.Echo WshShell.Environment.Item("NUMBER_OF_PROCESSORS")
'*************  Конец **************************************
```

Метод Run

Параметр `intWindowStyle` устанавливает вид окна для запускаемого приложения.

Таблица 1.7. Типы окна (intWindowStyle)

Параметр	Константа Visual Basic	Описание
0	`vbHide`	Прячет текущее окно и активизирует другое окно (показывает его и передает ему фокус)
1	`vbNormalFocus`	Активизирует и отображает окно. Если окно было минимизировано или максимизировано, система восстановит его первоначальное положение и размер. Этот

		флаг должен указываться сценарием во время первого отображения окна
2	vbMinimizedFocus	Активизирует окно и отображает его в минимизированном (свернутом) виде
3	vbMaximizedFocus	Активизирует окно и отображает его в максимизированном (развернутом) виде
4	vbNormalNoFocus	Отображает окно в том виде, в котором оно находилось последний раз. Активное окно при этом остается активным
5		Активизирует окно и отображает его в текущем состоянии
6	vbMinimizedNoFocus	Минимизирует заданное окно и активизирует следующее (в Z-порядке) окно
7		Отображает окно в свернутом виде. Активное окно при этом остается активным
8		Отображает окно в его текущем состоянии. Активное окно при этом остается активным
9		Активизирует и отображает окно. Если окно было минимизировано или максимизировано, система восстановит его первоначальное положение и размер. Этот флаг должен указываться, если производится восстановление свернутого окна (его нельзя использовать в методе Run)
10		Устанавливает режим отображения, опирающийся на режим программы, которая запускает приложение

Необязательный параметр bWaitOnReturn является логической переменной, дающей указание ожидать завершения запущенного процесса. Если этот параметр не указан или установлен в false, то после запуска из сценария нового процесса управление сразу же возвращается обратно в сценарий (не дожидаясь завершения запущенного процесса). Если же bWaitOnReturn установлен в true, то сценарий возобновит работу только после завершения вызванного процесса. При этом если параметр bWaitOnReturn равен true, то метод Run возвращает код выхода вызванного приложения. Если же bWaitOnReturn равен false или не задан, то метод Run всегда возвращает ноль.

```
'*********************************************************************
'* Имя: RetCode.vbs
'* Язык: VBScript
'* Описание: Вывод кода выхода запущенного приложения
'*********************************************************************
' Создаем объект WshShell
Set WshShell = WScript.CreateObject("WScript.Shell")
' Запускаем Блокнот и ожидаем завершения его работы
Return = WshShell.Run("notepad " + WScript.ScriptFullName, 1, true)
' Печатаем код возврата
WScript.Echo "Код возврата:", Return
'*************  Конец ***********************************************
```

Метод SendKeys

Каждая клавиша задается одним или несколькими символами. Например, Для того чтобы задать нажатие друг за другом букв А, Б и В нужно указать в качестве параметра для SendKeys строку "АБВ": string="АБВ".

Несколько символов имеют в методе SendKeys специальное значение: +, ^, %, ~, (,). Для того чтобы задать один из этих символов, их нужно заключить в фигурные скобки ({}). Например, для задания знака плюс используется {+}. Квадратные скобки ([]) хотя и не имеют в методе SendKeys специального смысла, их также нужно заключать в фигурные

скобки. Кроме этого, для задания самих фигурных скобок следует использовать следующие конструкции: {{} (левая скобка) и {}} (правая скобка).

Таблица 1.8. Коды специальных клавишей для SendKeys			
Названия клавиши	Код	Названия клавиши	Код
<Backspace>	{BACKSPACE}, {BS} или {BKSP}	<→>	{RIGHT}
<Break>	{BREAK}	<F1>	{F1}
<Caps Lock>	{CAPSLOCK}	<F2>	{F2}
<Del> или <Delete>	{DELETE} или {DEL}	<F3>	{F3}
<End>	{END}	<F4>	{F4}
<Enter>	{ENTER} или ~	<F5>	{F5}
<Esc>	{ESC}	<F6>	{F6}
<Home>	{HELP}	<F7>	{F7}
<Ins> или <Insert>	{INSERT} или {INS}	<F8>	{F8}
<Num Lock>	{NUMLOCK}	<F9>	{F9}
<Page Down>	{PGDN}	<F10>	{F10}
<Page Up>	{PGUP}	<F11>	{F11}
<Print Screen>	{PRTSC}	<F12>	{F12}
<Scroll Lock>	{SCROLLLOCK}	<F13>	{F13}
<Tab>	{TAB}	<F14>	{F14}
<↑>	{UP}	<F15>	{F15}
<←>	{LEFT}	<F16>	{F16}
<↓>	{DOWN}		

Таблица 5.8. Коды клавиш <Shift>, <Ctrl> и <Alt>	
Клавиша	Код
<Shift>	+
<Ctrl>	^
<Alt>	%

Для того чтобы задать комбинацию клавиш, которую нужно набирать, удерживая нажатыми клавиши <Shift>, <Ctrl> или <Alt>, нужно заключить коды этих клавиш в скобки. Например, если требуется сымитировать нажатие клавиш G и S при нажатой клавише <Shift>, следует использовать последовательность "+(GS)". Для того же, чтобы задать одновременное нажатие клавиш <Shift>+<G>, а затем <S> (уже без <Shift>), используется "+GS".

В методе SendKeys можно задать несколько нажатий подряд одной и той же клавиши. Для этого необходимо в фигурных скобках указать код нужной клавиши, а через пробел — число нажатий. Например, {LEFT 42} означает нажатие клавиши <?> 42 раза подряд; {h 10} означает нажатие клавиши h 10 раз подряд.

```
'*************************************************************
'* Имя: RunCalc.vbs
'* Язык: VBScript
'* Описание: Активизация приложения с помощью имени окна
'*************************************************************
Dim WshShell
' Создаем объект WshShell
Set WshShell=WScript.CreateObject("WScript.Shell")
```

```
' Запускаем Калькулятор
WshShell.Run "calc"
' Приостанавливаем сценарий на 0,1 секунды
WScript.Sleep 100
' Активизируем Калькулятор
WshShell.AppActivate "Calculator"
' Приостановка сценария на 0,1 секунды
WScript.Sleep(100)
' Посылаем нажатия клавиш в Калькулятор
WshShell.SendKeys "1{+}"
WScript.Sleep 500
WshShell.SendKeys "2"
WScript.Sleep 500
WshShell.SendKeys "~"
WScript.Sleep 2500
```

1.9 Объект WshArguments

Объект WshArguments содержит коллекцию всех параметров командной строки запущенного сценария или ярлыка Windows. Этот объект можно создать только с помощью свойства Arguments объектов WScript и WshShortcut.

С помощью объекта WshArguments можно также выделять и отдельно обрабатывать аргументы сценария, у которых имеются имена (например, /Name:Andrey) и безымянные аргументы. Ясно, что использование именных параметров более удобно, так как в этом случае нет необходимости запоминать, в каком порядке должны быть записаны параметры при запуске того или иного сценария.

Для доступа к именным и безымянным аргументам используются соответственно два специальных свойства объекта WshArguments: Named и Unnamed. Свойство Named содержит ссылку на коллекцию WshNamed, свойство Unnamed — на коллекцию WshUnnamed.

Таким образом, обрабатывать параметры командной строки запущенного сценария можно тремя способами:

- просматривать полный набор всех параметров (как именных, так и безымянных) с помощью коллекции WshArguments;
- выделить только те параметры, у которых есть имена (именные параметры) с помощью коллекции WshNamed;
- выделить только те параметры, у которых нет имен (безымянные параметры) с помощью коллекции WshUnnamed.

```
'**********************************************************************
' Имя: Args.vbs
' Язык: VBScript
' Описание: Работа с аргументами запущенного сценария
'**********************************************************************
Option Explicit

Dim i,Arg,objArgs,s,objNamedArgs,objUnnamedArgs  ' Объявляем переменные

Set objArgs = WScript.Arguments  ' Создаем объект WshArguments
' Определяем общее количество аргументов
s="Всего аргументов: " & objArgs.Count() & vbCrLf
For Each Arg In objArgs
  s=s & Arg & vbCrLf  ' Формируем строки со значениями аргументов
Next

Set objUnnamedArgs=objArgs.Unnamed  ' Создаем объект WshUnnamed
' Определяем количество безымянных аргументов
s=s & vbCrLf & "Безымянных аргументов: " & objUnnamedArgs.length & vbCrLf
For Each Arg In objUnnamedArgs
```

```
  ' Формируем строки со значениями безымянных аргументов
  s=s & Arg & vbCrLf
Next

Set objNamedArgs=objArgs.Named  ' Создаем объект WshNamed
' Определяем количество именных аргументов
s=s & vbCrLf & "Именных аргументов: " & objNamedArgs.Length & vbCrLf
' Проверяем, существует ли аргумент /Имя:
If objNamedArgs.Exists("Имя") Then
  s=s & objNamedArgs("Имя") & vbCrLf
End If
' Проверяем, существует ли аргумент /Comp:
If objNamedArgs.Exists("Comp") Then
  s=s & objNamedArgs("Comp") & vbCrLf
End If

WScript.Echo s   ' Выводим сформированные строки
'*************  Конец *********************************************
```

1.10 Объект WshEnvironment

Объект WshEnvironment позволяет получить доступ к коллекции, содержащей переменные среды заданного типа (переменные среды операционной системы, переменные среды пользователя или переменные среды текущего командного окна). Этот объект можно создать с помощью свойства Environment объекта WshShell или одноименного его метода:

```
Set WshShell=WScript.CreateObject("WScript.Shell")
Set WshSysEnv=WshShell.Environment
Set WshUserEnv=WshShell.Environment("User")
```

Объект WshEnvironment имеет свойство Length, в котором хранится число элементов в коллекции (количество переменных среды), и методы Count и Item. Для того чтобы получить значение определенной переменной среды, в качестве аргумента метода Item указывается имя этой переменной в двойных кавычках.

```
'*********************************************************************
' Имя: Environment.vbs
' Язык: VBScript
' Описание: Работа с переменными среды
'*********************************************************************
Dim WshShell, WshSysEnv
Set WshShell=WScript.CreateObject("WScript.Shell")
Set WshSysEnv=WshShell.Environment
WScript.Echo "Системный путь:",WshSysEnv.Item("PATH")
```

Можно также просто указать имя переменной в круглых скобках после имени объекта:

```
WScript.Echo "Системный путь:",WshSysEnv("PATH")
```

Кроме этого у объекта WshEnvironment имеется метод Remove(strName), который удаляет заданную переменную среды.

1.11 Объект WshSpecialFolders

При установке Windows всегда автоматически создаются несколько специальных папок (например, папка для рабочего стола (Desktop) или папка для меню Пуск (Start)), путь к которым впоследствии может быть тем или иным способом изменен. Объект WshSpecialFolders обеспечивает доступ к коллекции, содержащей пути к специальным папкам Windows; задание путей к таким папкам может понадобиться, например, для создания непосредственно из сценария ярлыков на рабочем столе. В Windows XP поддерживаются следующие имена специальных папок:

1. Desktop;

2. Favorites;
3. Fonts;
4. MyDocuments;
5. NetHood;
6. PrintHood;
7. Programs;
8. Recent;
9. SendTo;
10. StartMenu;
11. Startup;
12. Templates;
13. AllUsersDesktop;
14. AllUsersStartMenu;
15. AllUsersPrograms;
16. AllUsersStartup.

Объект `WshSpecialFolders` создается с помощью свойства `SpecialFolders` объекта `WshShell`:

```
var WshShell=WScript.CreateObject("WScript.Shell"),
    WshSpecFold=WshShell.SpecialFolders;
```

Сценарий, формирующий список всех имеющихся в системе специальных папок.

```
'*****************************************************************
' Имя: SpecFold1.vbs
' Язык: VBScript
' Описание: Вывод названий всех специальных папок Windows
'*****************************************************************
Option Explicit

Dim WshShell, WshFldrs, SpecFldr, s  ' Объявляем переменные
' Создаем объект WshShell
Set WshShell = WScript.CreateObject("Wscript.Shell")
' Создаем объект WshSpecialFolders
Set WshFldrs = WshShell.SpecialFolders
s="Список всех специальных папок:" & vbCrLf & vbCrLf
' Перебираем все элементы коллекции WshFldrs
For Each SpecFldr In WshFldrs
  ' Формируем строки с путями к специальным папкам
  s=s & SpecFldr & vbCrLf
Next
WScript.Echo s
```

Объект `WshSpecialFolders` также позволяет получить путь к конкретно заданной специальной папке.

```
'*****************************************************************
' Имя: SpecFold2.vbs
' Язык: VBScript
' Описание: Вывод названий заданных специальных папок Windows
'*****************************************************************
Option Explicit

Dim WshShell, WshFldrs, s   ' Объявляем переменные
' Создаем объект WshShell
Set WshShell = WScript.CreateObject("Wscript.Shell")
' Создаем объект WshSpecialFolders
Set WshFldrs = WshShell.SpecialFolders
' Формируем строки с путями к конкретным специальным папкам
s="Некоторые специальные папки:" & vbCrLf & vbCrLf
s=s+"Desktop:"+WshFldrs("Desktop") & vbCrLf
s=s+"Favorites:"+WshFldrs("Favorites") & vbCrLf
s=s+"Programs:"+WshFldrs("Programs")
WScript.Echo s   ' Выводим сформированные строки на экран
```

```
'*************  Конец ********************************************/
```

1.12 Сценарии WSH для доступа к файловой системе. Объектная модель FileSystemObject

Для работы с файловой системой из сценариев WSH предназначены восемь объектов, главным из которых является `FileSystemObject`. С помощью методов объекта `FileSystemObject` можно выполнять следующие основные действия:

1. копировать или перемещать файлы и каталоги;
2. удалять файлы и каталоги;
3. создавать каталоги;
4. создавать или открывать текстовые файлы;
5. создавать объекты `Drive`, `Folder` и `File` для доступа к конкретному диску, каталогу или файлу соответственно.

С помощью свойств объектов `Drive`, `Folder` и `File` можно получить детальную информацию о тех элементах файловой системы, с которыми они ассоциированы. Объекты `Folder` и `File` также предоставляют методы для манипулирования файлами и каталогами (создание, удаление, копирование, перемещение); эти методы в основном копируют соответствующие методы объекта `FileSystemObject`.

Кроме этого имеются три объекта-коллекции: `Drives`, `Folders` и `Files`. Коллекция `Drives` содержит объекты `Drive` для всех имеющихся в системе дисков, `Folders` — объекты `Folder` для всех подкаталогов заданного каталога, `Files` — объекты `File` для всех файлов, находящихся внутри определенного каталога.

Наконец, из сценария можно читать информацию из текстовых файлов и записывать в них данные. Методы для этого предоставляет объект `TextStream`.

Таблица 1.9. Выполнение основных файловых операций

Операция	Используемые объекты, свойства и методы
Получение сведений об определенном диске (тип файловой системы, метка тома, общий объем и количество свободного места и т.д.)	Свойства объекта `Drive`. Сам объект `Drive` создается с помощью метода `GetDrive` объекта `FileSystemObject`
Получение сведений о заданном каталоге или файле (дата создания или последнего доступа, размер, атрибуты и т.д.)	Свойства объектов `Folder` и `File`. Сами эти объекты создаются с помощью методов `GetFolder` и `GetFile` объекта `FileSystemObject`
Проверка существования определенного диска, каталога или файла	Методы `DriveExists`, `FolderExists` и `FileExists` объекта `FileSystemObject`
Копирование файлов и каталогов	Методы `CopyFile` и `CopyFolder` объекта `FileSystemObject`, а также методы `File.Copy` и `Folder.Copy`
Перемещение файлов и каталогов	Методы `MoveFile` и `MoveFolder` объекта `FileSystemObject`, или методы `File.Move` и `Folder.Move`
Удаление файлов и каталогов	Методы `DeleteFile` и `DeleteFolder` объекта `FileSystemObject`, или методы `File.Delete` и `Folder.Delete`
Создание каталога	Методы `FileSystemObject.CreateFolder` или `Folders.Add`
Создание текстового файла	Методы `FileSystemObject.CreateTextFile` или

	`Folder.CreateTextFile`
Получение списка всех доступных дисков	Коллекция `Drives`, содержащаяся в свойстве `FileSystemObject.Drives`
Получение списка всех подкаталогов заданного каталога	Коллекция `Folders`, содержащаяся в свойстве `Folder.SubFolders`
Получение списка всех файлов заданного каталога	Коллекция `Files`, содержащаяся в свойстве `Folder.Files`
Открытие текстового файла для чтения, записи или добавления	Методы `FileSystemObject.CreateTextFile` или `File.OpenAsTextStream`
Чтение информации из заданного текстового файла или запись ее в него	Методы объекта `TextStream`

Получение сведений о диске

В сценарий DriveInfo.vbs, который выводит на экран некоторые свойства диска C.

```
'*********************************************************************
' Имя: DriveInfo.vbs
' Язык: VBScript
' Описание: Вывод на экран свойств диска C
'*********************************************************************
'Объявляем переменные
Dim FSO,D,TotalSize,FreeSpace,s
'Создаем объект FileSystemObject
Set FSO = WScript.CreateObject("Scripting.FileSystemObject")
'Создаем объект Drive для диска C
Set D = FSO.GetDrive("C:")
s = "Информация о диске C:" & VbCrLf
'Получаем серийный номер диска
s = s &"Серийный номер: " & D.SerialNumber & VbCrLf
'Получаем метку тома диска
s = s & "Метка тома: " & D.VolumeName & VbCrLf
'Вычисляем общий объем диска в килобайтах
TotalSize = D.TotalSize/1024
s = s & "Объем: " & TotalSize & " Kb" & VbCrLf
'Вычисляем объем свободного пространства диска в килобайтах
FreeSpace = D.FreeSpace/1024
s = s & "Свободно: " & FreeSpace & " Kb" & VbCrLf
'Выводим свойства диска на экран
WScript.Echo s
'*************  Конец *********************************************
```

Получение сведений о каталоге

В сценарии FolderInfo.vbs на экран выводятся свойства каталога, из которого был запущен сценарий.

```
'*********************************************************************
' Имя: FolderInfo.vbs
' Язык: VBScript
' Описание: Вывод на экран даты создания текущего каталога
'*********************************************************************
Dim FSO,WshShell,FoldSize,s  'Объявляем переменные

'Создаем объект FileSystemObject
Set FSO = WScript.CreateObject("Scripting.FileSystemObject")
'Создаем объект WshShell
Set WshShell = WScript.CreateObject("WScript.Shell")

'Определяем каталог, из которого был запущен сценарий
'(текущий каталог)
Set Folder = FSO.GetFolder(WshShell.CurrentDirectory)
'Получаем имя текущего каталога
```

```
s = "Текущий каталог: " & Folder.Name & VbCrLf
'Получаем дату создания текущего каталога
s = s & "Дата создания: " & Folder.DateCreated & VbCrLf
'Вычисляем размер текущего каталога в килобайтах
FoldSize=Folder.Size/1024
s = s & "Объем: " & FoldSize & " Kb" & VbCrLf
'Выводим информацию на экран
WScript.Echo s
'*************  Конец ********************************************
```

Получение сведений о файле

В сценарий FileInfo.vbs, в котором на экран выводятся некоторые свойства файла C:\boot.ini.

```
'*********************************************************************
' Имя: FileInfo.vbs
' Язык: VBScript
' Описание: Вывод на экран некоторых свойств файла
'*********************************************************************
Dim FSO,F,s   'Объявляем переменные

'Создаем объект FileSystemObject
Set FSO = WScript.CreateObject("Scripting.FileSystemObject")
'Создаем объект File
Set F = FSO.GetFile("C:\boot.ini")

'Получаем имя файла
s = "Файл: " & F.Name & VbCrLf
'Получаем дату создания файла
s = s & "Дата создания: " & F.DateCreated & VbCrLf
'Получаем тип файла
s = s & "Тип: " & F.Type & VbCrLf
'Выводим информацию на экран
WScript.Echo s
'*************  Конец ********************************************
```

Проверка существования диска, каталога или файла

В сценарий IsExistsFile.vbs, в котором проверяется наличие на диске C файла boot.ini.

```
'*********************************************************************
' Имя: IsExistsFile.vbs
' Язык: VBScript
' Описание: Проверка существования файла
'*********************************************************************
Dim FSO,FileName 'Объявляем переменные

'Создаем объект FileSystemObject
Set FSO = WScript.CreateObject("Scripting.FileSystemObject")

FileName = "C:\boot.ini"
if  FSO.FileExists(FileName)  Then
  'Выводим информацию на экран
  WScript.Echo "Файл " & FileName & " существует"
else
  'Выводим информацию на экран
  WScript.Echo "Файл " & FileName & " не существует"
end if
'*************  Конец ********************************************
```

1.13 Получение списка всех имеющихся дисков

Каждому из дисков компьютера (включая подключенные сетевые диски и дисководы со сменными носителями) соответствует элемент коллекции `Drives` (объект `Drive`). Таким образом, для построения списка дисков компьютера нужно в цикле перебрать все элементы коллекции `Drives`.

В сценарий ListDrives.vbs, в котором на экран выводятся сведения обо всех доступных дисках.

```
'********************************************************************
' Имя: ListDrives.vbs
' Язык: VBScript
' Описание: Получение списка всех имеющихся дисков
'********************************************************************
'Объявляем переменные
Dim FSO,s,ss,Drives,D

'Создаем объект FileSystemObject
Set FSO = WScript.CreateObject("Scripting.FileSystemObject")
'Создаем коллекцию дисков, имеющихся в системе
Set Drives = FSO.Drives
s = ""

'Перебираем все диски в коллекции
For Each D In Drives
  'Получаем букву диска
  s = s & D.DriveLetter
  s = s & " - "
  if (D.DriveType = 3) then  'Проверяем, не является ли диск сетевым
    'Получаем имя сетевого ресурса
    ss = D.ShareName
  else
    'Диск является локальным
    if (D.IsReady) then  'Проверяем готовность диска
      'Если диск готов, то получаем метку тома для диска
      ss = D.VolumeName
    else
      ss = "Устройство не готово"
    end if
  s = s & ss & VbCrLf
  end if
Next

'Выводим полученные строки на экран
WScript.Echo s
'*************  Конец ***********************************************
```

Получение списка всех подкаталогов заданного каталога

Для построения списка всех подкаталогов определенного каталога можно воспользоваться коллекцией `Folders`, которая хранится в свойстве `SubFolders` соответствующего объекта `Folder` и содержит объекты `Folder` для всех подкаталогов.

В сценарий ListSubFold.vbs, в котором на экран выводятся названия всех подкаталогов каталога C:\Program Files.

```
'********************************************************************
' Имя: ListSubFold.vbs
' Язык: VBScript
' Описание: Получение списка всех подкаталогов заданного каталога
'********************************************************************
'Объявляем переменные
Dim FSO,F,SFold,SubFolders,Folder,s

'Создаем объект FileSystemObject
Set FSO = WScript.CreateObject("Scripting.FileSystemObject")
'Путь к каталогу
SFold = "C:\Program Files"
s = "Каталог " & SFold & VbCrLf
s = s & "Подкаталоги:" & VbCrLf
'Создаем объект Folder для каталога C:\Program Files
Set F=FSO.GetFolder(SFold)

'Создаем коллекцию подкаталогов каталога C:\Program Files
Set SubFolders = F.SubFolders
```

```
'Цикл по всем подкаталогам
For Each Folder In SubFolders
  'Добавляем строку с именем подкаталога
   s = s & Folder & VbCrLf
Next

'Выводим полученные строки на экран
WScript.Echo s
'*************  Конец ******************************************/
```

Получение списка всех файлов заданного каталога

В свойстве `Files` объекта `Folder`, соответствующего определенному каталогу, хранится коллекция находящихся в этом каталоге файлов (объектов `File`). В сценарии ListFiles.vbs, выводящий на экран названия всех файлов, которые содержатся в специальной папке Мои документы.

```
'*********************************************************************
' Имя: ListFiles.vbs
' Язык: VBScript
' Описание: Получение списка всех файлов заданного каталога
'*********************************************************************
'Объявляем переменные
Dim FSO,F,File,Files,WshShell,PathList,s

'Создаем объект FileSystemObject
Set FSO = WScript.CreateObject("Scripting.FileSystemObject")
'Создаем объект WshShell
Set WshShell = WScript.CreateObject("Wscript.Shell")
'Создаем объект WshSpecialFolders
Set WshFldrs = WshShell.SpecialFolders
'Определяем путь к папке Мои документы
PathList = WshFldrs.item("MyDocuments") & "\"
'Создаем объект Folder для папки Мои документы
Set F = FSO.GetFolder(PathList)
'Создаем коллекцию файлов каталога Мои документы
Set Files = F.Files
s = "Файлы из каталога " & PathList & VbCrLf
'Цикл по всем файлам
For Each File In Files
  'Добавляем строку с именем файла
  s = s & File.Name & VbCrLf
Next

'Выводим полученные строки на экран
WScript.Echo s
'*************  Конец ******************************************
```

Создание каталога

Создать новый каталог на диске можно либо с помощью метода `CreateFolder` объекта `FileSystemObject`, либо с помощью метода `Add` коллекции `Folders`. Оба эти метода используются в сценарии MakeFolder.vbs для создания в каталоге C:\Мои документы подкаталогов Новая папка и Еще одна новая папка.

```
'*********************************************************************
' Имя: MakeFolder.vbs
' Язык: VBScript
' Описание: Создание нового каталога
'*********************************************************************
'Объявляем переменные
Dim FSO, F, SubFolders

'Создаем объект FileSystemObject
Set FSO = WScript.CreateObject("Scripting.FileSystemObject")
'Создаем каталог C:\Program Files\Новая папка
```

```
FSO.CreateFolder("C:\Program Files\Новая папка")
0'Создаем объект Folder для каталога C:\Program Files
Set F = FSO.GetFolder("C:\Program Files")
'Создаем коллекцию подкаталогов каталога C:\Program Files
Set SubFolders = F.SubFolders
'Создаем каталог C:\Program Files\Еще одна новая папка
SubFolders.Add "Еще одна новая папка"
'*************  Конец ******************************************
```

Создание текстового файла

Для создания текстового файла используется метод `CreateTextFile` объекта `FileSystemObject`, который имеет один обязательный текстовый параметр (путь к создаваемому файлу) и два необязательных логических параметра (`Overwrite` и `Unicode`).

Параметр `Overwrite` имеет значение в том случае, когда создаваемый файл уже существует. Если `Overwrite` равно `True`, то такой файл перепишется (старое содержимое будет утеряно), если же в качестве `Overwrite` указано `False`, то файл переписываться не будет. Если этот параметр вообще не указан, то существующий файл также не будет переписан.

```
'*********************************************************************
' Имя: CreateTempFile.vbs
' Язык: VBScript
' Описание: Создание временного файла со случайным именем
'*********************************************************************
Dim FSO,FileName,F,s  'Объявляем переменные
'Создаем объект FileSystemObject
Set FSO = WScript.CreateObject("Scripting.FileSystemObject")
'Генерируем случайное имя файла
FileName = FSO.GetTempName
'Создаем файл с именем FileName
Set F = FSO.CreateTextFile(FileName, true)
'Закрываем файл
F.Close
'Сообщаем о создании файла
WScript.Echo "Был создан файл " & FileName
'*************  Конец ******************************************
```

Операции чтения и записи строк для текстового файла

Последовательный (строка за строкой) доступ к текстовому файлу обеспечивает объект `TextStream`. Методы этого объекта позволяют читать информацию из файла и записывать ее в него. Создается объект `TextStream` при открытии или создании текстового файла с помощью следующих методов:

1. CreateTextFile объектов FileSystemObject и Folder;
2. OpenTextFile объекта FileSystemObject;
3. OpenAsTextStream объекта File.

Таблица 1.10. Параметр Iomode

Константа	Значение	Описание
`ForReading`	1	Файл открывается только для чтения, записывать информацию в него нельзя
`ForWriting`	2	Файл открывается для записи. Если файл с таким именем уже существовал, то при новой записи его содержимое теряется
`ForAppending`	8	Файл открывается для добавления. Если файл уже существовал, то информация будет дописываться в конец этого файла

Таблица 1.11. Параметр Format

Константа	Значение	Описание
TristateUseDefault	-2	Файл открывается в формате, используемом системой по умолчанию
TristateTrue	-1	Файл открывается в формате Unicode
TristateFalse	0	Файл открывается в формате ASCII

```
'*****************************************************************
' Имя: TextFile.vbs
' Язык: VBScript
' Описание: Запись строк в текстовый файл и чтение из него
'*****************************************************************
Dim FSO,F,TextStream,s  'Объявляем переменные
' Инициализируем константы
Const ForReading = 1, ForWriting = 2, TristateUseDefault = -2

' Создаем объект FileSystemObject
Set FSO=WScript.CreateObject("Scripting.FileSystemObject")
' Создаем в текущем каталоге файл test1.txt
FSO.CreateTextFile "test1.txt"
' Создаем объект File для файла test1.txt
set F=FSO.GetFile("test1.txt")
' Создаем объект TextStream (файл открывается для записи)
Set TextStream=F.OpenAsTextStream(ForWriting, TristateUseDefault)
' Записываем в файл строку
TextStream.WriteLine "Это первая строка"
' Закрываем файл
TextStream.Close
' Открываем файл для чтения
Set TextStream=F.OpenAsTextStream(ForReading, TristateUseDefault)
' Считываем строку из файла
s=TextStream.ReadLine
' Закрываем файл
TextStream.Close
' Отображаем строку на экране
WScript.Echo "Первая строка из файла test1.txt:" & vbCrLf & vbCrLf & s
'*************  Конец **************************************
```

Копирование и перемещение файлов и каталогов

Для копирования файлов/каталогов можно применять метод CopyFile/CopyFolder объекта FileSystemObject или метод Copy соответствующего этому файлу/каталогу объекта File/Folder. Перемещаются файлы/каталоги с помощью методов MoveFile/MoveFolder объекта FileSystemObject или метода Move соответствующего этому файлу/каталогу объекта File/Folder.

В сценарий CopyFile.vbs, иллюстрирующий использование метода Copy. В этом сценариях на диске C создается файл TestFile.txt, который затем копируется на рабочий стол.

```
'*****************************************************************
' Имя: CopyFile.vbs
' Язык: VBScript
' Описание: Создание и копирование файла
'*****************************************************************
'Объявляем переменные
Dim FSO,F,WshShell,WshFldrs,PathCopy

'Создаем объект FileSystemObject
Set FSO = WScript.CreateObject("Scripting.FileSystemObject")
'Создаем файл
Set F = FSO.CreateTextFile("C:\TestFile.txt", true)
'Записываем в файл строку
F.WriteLine "Тестовый файл"
'Закрываем файл
```

```
F.Close

'Создаем объект WshShell
Set WshShell = WScript.CreateObject("Wscript.Shell")
'Создаем объект WshSpecialFolders
Set WshFldrs = WshShell.SpecialFolders
'Определяем путь к рабочему столу
PathCopy = WshFldrs.item("Desktop")+"\"
'Создаем объект File для файла C:\TestFile.txt
Set F = FSO.GetFile("C:\TestFile.txt")
'Копируем файл на рабочий стол
F.Copy PathCopy
'*************  Конец ************************************************
```

Удаление файлов и каталогов

Для удаления файлов/каталогов можно применять метод `DeleteFile/DeleteFolder` объекта `FileSystemObject` или метод `Delete` соответствующего этому файлу/каталогу объекта `File/Folder`. Отметим, что при удалении каталога неважно, является ли он пустым или нет — удаление будет произведено в любом случае. Если же заданный для удаления файл/каталог не будет найден, то возникнет ошибка.

В сценарий DeleteFile.vbs, в котором производится удаление предварительно созданного файла C:\TestFile.txt.

```
'*********************************************************************
' Имя: DeleteFile.vbs
' Язык: VBScript
' Описание: Создание и удаление файла
'*********************************************************************
'Объявляем переменные
Dim FSO,F,FileName

'Создаем объект FileSystemObject
Set FSO = WScript.CreateObject("Scripting.FileSystemObject")
'Задаем имя файла
FileName="C:\TestFile.txt"
'Создаем файл
Set F = FSO.CreateTextFile(FileName, true)
'Записываем в файл строку
F.WriteLine "Тестовый файл"
'Закрываем файл
F.Close
WScript.Echo "Файл создан"
FSO.DeleteFile FileName
WScript.Echo "Файл удален"
'*************  Конец ************************************************
```

2. Практическое задание

Создать сценарий реализующий в консольном режиме диалог с пользователем в виде меню. Сценарий должен выполняться циклически пока не выбран пункт «Выход». Первый пункт меню должен выводить информацию о создателе (ФИО, группа) и краткое описание выполняемых действий, остальные пункты реализуют действия указанные в таблице в соответствии с вариантом. Все параметры задаются в результате диалога с пользователем. При выполнении задания А, допускается использование командных файлов рассмотренных в первой лабораторной работе. Сценарий запускается в **консольной** версии WSH.
Отчет должен содержать краткие теоретические сведения о **использованных** объектах, методах и свойствах.

Вариант	Задание
1	А)Создание паки в указанном месте. Б)Открытие блокнота и сохранение в нем заданного сообщения.
2	А)Копирование файлов с указанного места в заданное. Б)Создание ярлыка для вызова заданной программы и помещение его на рабочий стол.
3	А)Удаление файлов заданного расширения в заданной папке. Б)Создание ярлыка для просмотра содержимого заданной папки и помещение его на рабочий стол.
4	А)Удаление содержимого заданной папки. Б)Создание ссылки на заданный сетевой ресурс и помещение его на рабочий стол.
5	А)Создание файла со списком папок в указанном месте. Б)Открытие блокнота и сохранение в нем имени пользователя.
6	А)Копирование файлов заданного расширения с указанного места в папку «*BackUp*», на указанном диске. Б)Вывод на экран пути к заданной специальной папке.
7	А)Перенос файлов заданного расширения с указанного места в папку «*BackUp*», на заданном диске. Б)Сохранение в блокноте параметра реестра: *HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\BuildLab*
8	А)Создание файла со списком системных файлов в указанном месте. Б)Сохранение в блокноте списка дисков с их размером.
9	А)Создание файла со списком скрытых файлов в указанном месте. Б)Сохранение в блокноте списка дисков с значением свободного места .
10	А)Перенос файлов с указанного места в заданное. Б)Сохранение в блокноте даты создания и размера заданной папки.
11	А)Переименование заданной папки. Б)Сохранение в блокноте даты создания и размера заданного файла.
12	А)Архивирование заданной папки. Б)Сохранение в текстовом файле списка папок в заданном каталоге.
13	А)Архивирование файлов заданного расширения в заданной папке. Б)Сохранение в текстовом файле списка файлов в заданной папке.
14	А)Создание файла со списком файлов заданного расширения на заданном диске. Б)Сохранение в текстовом файле списка специальных папок.
15	А)Копирование всех фалов с заданным расширением с заданного диска в указанную папку. Б)Сохранение в текстовом файле списка текстовых файлов в папке «*мои документы*» с датой их создания.

Пример

Создать сценарий реализующий в консольном режиме диалог с пользователем в виде меню реализующий: создание файла со списком файлов с расширением .ini", создание ярлыка на заданный сетевой ресурс.

```
'* Имя: Interact.vbs
'* Язык: VBScript
'* Описание: пример лабораторной работы
'*********************************************************************
Dim  s
' Выводим строку на экран
do
 WScript.StdOut.WriteLine "МЕНЮ:"
 WScript.StdOut.WriteLine "--------------------------"
 WScript.StdOut.WriteLine "1. Информация о авторе"
 WScript.StdOut.WriteLine "2. Создание файла со списком файлов с расширением
.ini"
 WScript.StdOut.WriteLine "3. Создание ярлыка на заданный сетевой ресурс"
 WScript.StdOut.WriteLine "4. Выход"
 WScript.StdOut.Write "Выберите пункт меню:"
 ' Считываем строку
 s = WScript.StdIn.ReadLine
 ' Создаем объект WshShell
 Set WshShell = WScript.CreateObject("WScript.Shell")

 if (s="1") Then
      WScript.StdOut.WriteLine "ФИО, группа"

 elseif(s="2") Then
      WScript.StdOut.Write "Введите имя файла:"
  f = WScript.StdIn.ReadLine

      ' Запускаем командный файл и ожидаем окончания ее работы
  Code=WshShell.Run("%COMSPEC% /c 02.cmd >"+f ,0,true)

      elseif(s="3") Then
      WScript.StdOut.Write "Введите имя ярлыка:"
  f = WScript.StdIn.ReadLine
  ' Создаем ярлык на сетевой ресурс
  Set oUrlLink = WshShell.CreateShortcut(f+".URL")
      WScript.StdOut.Write "Введите имя ресурса:"
  f = WScript.StdIn.ReadLine
  ' Устанавливаем URL
  oUrlLink.TargetPath = f
  ' Сохраняем ярлык
  oUrlLink.Save
End if

loop until (s="4")

'*************  Конец ********************************************
```

Командный файл 02.cmd

```
@ECHO OFF
CLS
FOR /R c:\ %%f IN (*.ini) DO echo %%f
```