МЕТОДИЧЕСКИЕ УКАЗАНИЯ

к лабораторным работам № 2-4 по курсу

СИСТЕМНОЕ ПО

«Программирование в Windows Script Host»

Сервер сценариев Windows Script Host

Сервер сценариев Windows Script Host (WSH) – это инструмент, позволяющий создавать специальные сценарии, работающие непосредственно в операционной системе Windows и использующие внешние объекты ActiveX. Различные версии сервера сценариев Windows Script Host (WSH) входят в стандартную поставку Windows. Сценарии WSH могут создаваться с помощью специализированных языков (например, Microsoft Visual Basic Script Edition (VBScript) или Microsoft JScript).

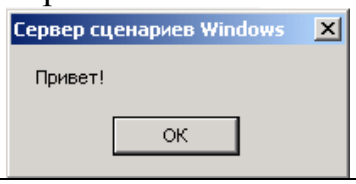
С помощью внутренних объектов WSH из сценариев можно выполнять следующие основные *задачи*:

- выводить информацию в стандартный выходной поток (на экран) или в диалоговое окно Windows;
- читать данные из стандартного входного потока (т.е. вводить данные с клавиатуры) или использовать информацию, выводимую другой командой;
- использовать свойства и методы внешних объектов, а также обрабатывать события этих объектов (т.е. которые генерируются этими объектами);
- запускать новые независимые процессы или активизировать уже имеющиеся;
- запускать дочерние процессы с возможностью контроля их состояния и доступ к их стандартным входным и выходным потокам;
- работать с локальной сетью: определять имя зарегистрировавшегося пользователя, подключать сетевые диски и принтеры;
- просматривать и изменять переменные среды;
- получать доступ к специальным папкам Windows;
- создавать ярлыки Windows;
- работать с системным реестром.

Первый скрипт

Простейший WSH-сценарий, написанный на языке JScript или VBScript, - это обычный текстовый файл с расширением ***js*** или ***vbs*** соответственно, создавать его можно в любом текстовом редакторе, способном сохранять документы в формате «Только текст».

В качестве первого примера создадим сценарий, выводящий на экран диалоговое окно с надписью "Привет!"




```
/*****************************************************/
/* Файл:  First.js                                   */
/* Язык:  JScript                                    */
/* Описание: Вывод на экран простого диалогового окна*/
/* Автор:  <Ваша фамилия>                            */
/*****************************************************/
```

```
WScript.Echo ("Привет!"); //Создание диалогового окна
                          //с надписью "Привет!"
/***  Конец ************************************/
```

Для запуска данного скрипта достаточно выполнить кликнуть на него или в командной строке набрать wscript (cscript – консольная версия) и имя скрипта.

Объекты WSH

1. **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.
2. **WshArguments**. Обеспечивает доступ к параметрам командной строки запущенного сценария или ярлыка Windows.
3. **WshNamed**. Обеспечивает доступ к именным параметрам командной строки запущенного сценария.
4. **WshUnnamed**. Обеспечивает доступ к безымянным параметрам командной строки запущенного сценария.
5. **WshShell**. Позволяет запускать независимые процессы, создавать ярлыки, работать с переменными среды, системным реестром и специальными папками Windows.
6. **WshSpecialFolders**. Обеспечивает доступ к специальным папкам Windows.
7. **WshShortcut**. Позволяет работать с ярлыками Windows.
8. **WshUrlShortcut**. Предназначен для работы с ярлыками сетевых ресурсов.
9. **WshEnvironment**. Предназначен для работы с переменными среды (для просмотра, изменения и удаления переменных среды).
10. **WshNetwork**. Используется при работе с локальной сетью: содержит сетевую информацию для локального компьютера, позволяет подключать сетевые диски и принтеры.
11. **WshScriptExec**. Позволяет запускать консольные приложения в качестве дочерних процессов, обеспечивает контроль этих приложений и доступ к их стандартным входным и выходным потокам.
12. **WshController**. Позволяет запускать сценарии на удаленных машинах.
13. **WshRemote**. Позволяет управлять сценарием, запущенным на удаленной машине.
14. **WshRemoteError**. Используется для получения информации об ошибке, возникшей в результате выполнения сценария, запущенного на удаленной машине.

Кроме этого, имеется объект **FileSystemObject**, обеспечивающий доступ к файловой системе компьютера.

Объект WScript

Объект *WScript* объект можно использовать сразу, без какого-либо предварительного описания или создания, т.к. его экземпляр создается сервером сценариев WSH автоматически. Для использования же других объектов нужно использовать либо метод **CreateObject**, либо определенное свойство другого объекта. Объект *WScript* поддерживает несколько свойств и методов:

Свойства объекта WScript

Свойство	Описание
Application	Предоставляет интерфейс IDispatch (интерфейс сервера сценариев) 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 обычно это C:\WINDOWS\system32\csript.exe для консольной версии WSH или C:\WINDOWS\system32\wsript.exe для оконной версии)
Name	Содержит название объекта *WScript* (Window Scripting Host)
Path	Содержит путь к каталогу, в котором находится wscript.exe или cscript.exe (в Windows 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

Методы объекта WScript

Метод	Описание
GreateObject (strProgID [, srtPrefix])	Создает объект, заданный параметром strProgID
ConnectObject (strObject, srtPrefix)	Устанавливает соединение с объектом strObject, позволяющее писать функции-обработчики его событий (имена этих должны начинаться с префикса srtPrefix)
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[,strPrefID],	Активизирует объект автоматизации,

[strPrefix])	определяемый заданным файлом (параметр strPathName) или объект, заданный параметром strPrefID
Quit ([intErrorCode])	Показывает 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

Выполнение основных операций с файловой системой

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

- создавать каталоги;
- создавать или открывать текстовые файлы;
- копировать или перемещать файлы и каталоги;
- удалять файлы и каталоги;
- 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 манипулирования файлами и каталогами (создание, удаление, копирование, перемещение); эти методы в основном копируют соответствующие методы объекта FileSystemObject.

Кроме того, 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.

В табл. кратко описано, какие объекты, свойства и методы могут использоваться для выполнения наиболее часто используемых файловых операций.

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

Задания

В соответствии с номером студента по списку в журнале преподавателя разработать скрипт (сценарий) (см. таблицу). В комментариях описать применение программы и сообщить информацию об авторе. Текущий каталог не изменять, если это специально не оговорено. Если производится копирование из одного места в другое, параметры источника и назначения передавать в командной строке.

В лабораторной работе № 2 необходимо:

1) вывести данные об авторе и задании (WScript.echo);

```
var s; //Объявление переменной
   // Формирование строки
s="Лабораторная работа N 2\n"+
   "Работа с файловой системой WSH\n"+
   "Выполнил ст-т гр. ПК-11Д\n";
WScript.Echo (s); // Печать строки
```

2) необходимые параметры передавать через командную строку (WScript.Arguments);

Пример на занесение параметров командной строки в массив args:

```
var i, n, objArgs, s, args;  // Объявляем переменные
objArgs = WScript.Arguments;  // Создаем объект WshArguments
  // Определяем общее количество аргументов
n = objArgs.Count();
s = "Всего аргументов: "+n+"\n";
args = new Array(n);
  // цикл по коллекции аргументов
for (i=0; i <= n-1; i++) {
  s += objArgs(i)+"\n";  // Формируем строки с аргументами
  args[i] = objArgs(i); // Записываем в массив
}
WScript.Echo(s);  // Выводим сформированные строки
```

3) выполнить файловую операцию по вариантам (FileSystemObject);

Пример на поиск в текущем каталоге файла наибольшего размера:

```
var FSO, Path, Fold, F, i = 0, CurSize, MaxSize = 0, MaxF, s;
FSO = WScript.CreateObject("Scripting.FileSystemObject");
    // получаем доступ к текущему каталогу
Path = FSO.GetParentFolderName(WScript.ScriptFullName);
Fold = FSO.GetFolder(Path);
    // Создаем коллекцию файлов
F = new Enumerator(Fold.Files);
    // Цикл по всем файлам
while ( ! F.atEnd()) {
    CurSize = F.item().Size; // размер текущего файла
      // сравниваем размер
    if (CurSize > MaxSize) {
        MaxSize = CurSize;  // новый максимум
        MaxF = F.item(); // файл наибольшего размера
    }
      // Переходим к следующему файлу
    F.moveNext();
```

```
    i++; // кол-во просмотренных файлов
} // while
s = "В текущем каталоге "+ Fold.Name + " "+i+" файлов.\n"+
    "Максимальный размер имеет "+MaxF.Name+" = "+
     Math.round(MaxSize/1024) + " Кб";
WScript.echo(s);
```

Пример на вывод дат создания (изменения, последнего обращения) и атрибутов файла, указанного в командной строке:

```
var file = "C:\\Autoexec.bat";  // файл автозагрузки
   // извлекам параметры командной строки - там имя файла
var objArgs = WScript.Arguments; // коллекция аргументов
if (objArgs.length > 0)          // кол-во аргументов > 0
    file = objArgs(0);           // получаем первый аргумент
    // Создание FileSystemObject для доступа к файловой системе
var FSO = WScript.CreateObject("Scripting.FileSystemObject");
   // проверка существования файла
if ( ! FSO.FileExists(file))
{
    WScript.Echo("Ошибка:", file, " не существует");
    WScript.Quit(1);
}
var F = FSO.GetFile(file);  // извлекаем в объект файл
   // Теперь извлекаем информацию о файле
var s = "Файл: \t\t" + F.Name + "\n";   // имя файла
     // даты создания, изменения, последнего обращения
var D1 = new Date(F.DateCreated); // создаем объект дата
     // конвертируем в удобоваримый формат
s += "Создан: \t\t" + D1.toLocaleDateString() + "\n";
var D2 = new Date(F.DateLastModified); // создаем объект дата
s += "Изменен: \t" + D2.toLocaleDateString() + "\n";
var D3 = new Date(F.DateLastAccessed); // создаем объект дата
s += "Обращение: \t" + D3.toLocaleDateString() + "\n";
    // расшифровка атрибутов
s += "Атрибуты: \t";
var attrib = F.Attributes;
if ((attrib & 0x01) != 0)   // Read-only
    s += "R ";
if ((attrib & 0x02) != 0)   // Hidden
    s += "H ";
if ((attrib & 0x04) != 0)   // System
    s += "S ";
if ((attrib & 0x20) != 0)   // Archive
    s += "A ";
if ((attrib & 0x800) != 0)  // Compressed (Windows NT)
    s += "C ";
WScript.Echo(s);  // вывод на экран
```

4) записать протокол выполненных дествий во внешний log-файл (объект TextStream).

```
var FSO = WScript.CreateObject("Scripting.FileSystemObject") ;
 // Создаем текстовый файл
var F = FSO.CreateTextFile("log.txt", true);
```

```
 // Записываем строку в файл
var D = new Date();
F.WriteLine("Протокол работы от :"+D.toLocaleDateString());
 // Закрываем файл
F.Close();
```

В работе № 3 необходимо:

1) предусмотреть поле ввода необходимых данных по вариантам (вызов функции InputBox языка VBScript в Jscript возможен при создании сценария WSF с XML-разметкой);

```
<job id="InputBox">
<script language="VBScript">
  Function InputName  ' Описываем функцию на языке VBScript
  ' Вводим имя в диалоговом окне
  InputName = InputBox("Введите Ваше имя:",
                       "Окно ввода VBScript")
  End Function
</script>
<script language="JScript">
    var s;
    s = InputName();  // Вызываем функцию InputName
     // Выводим значение переменной s на экран
    WScript.Echo("Здравствуйте, "+s+"!");
</script>
</job>
```

2) произвести контроль вводимых данных (на соответствие датам, строкам, числам и пр.);

```
var args = WScript.Arguments; // коллекция аргументов
var n = args.Count();
    // проверка на кол-во аргументов
if (n==0) {
WScript.echo("Задайте аргумент в командной строке !");
WScript.Quit(); // выход из сценария
}
var s = args(0);
if (isNaN(s))
{
WScript.echo("Аргумент - Not A Number !");
WScript.Quit(); // выход из сценария
}
m = parseInt(s);
if (! isFinite(m))
{
WScript.echo("Аргумент не является числом !");
WScript.Quit(); // выход из сценария
}
WScript.echo("Вы ввели число "+m);
```

3) создать диалог для подтверждения операций (MsgBox или Popup).

```
var WshShell, Res, Text, Title; // Объявление переменных
  // Инициализация констант для диалоговых окон
var vbOkCancel=1, vbOk=1;
  // Создание объекта WshShell
```

```
WshShell = WScript.CreateObject("WScript.Shell");
Text = "Вы действительно хотите удалить файлы ?";
Title = "Сообщение";
  // Вывод диалогового окна на экран
Res=WshShell.Popup(Text,0, Title, vbOkCancel);
  // Определение, какая из кнопок была нажата в диалоговом окне
if (Res==vbOk)
  WshShell.Popup ("Нажата кнопка OK");
else
  WshShell.Popup ("Нажата кнопка Отмена");
```

В работе № 4 необходимо:

1) Записать информацию о всех файлах (каталога) в таблицу Ms Word или Excel, состоящую из 4 колонок – порядковый номер, имя файла, расширение (дата, размер и т.п. – в зависимости от задания в 3-ей работе), выполненное действие. После заполнения таблицы должна быть произведена сортировка по 3-ей колонке. В следующем примере в таблицу Ms Word заносятся все подкаталоги текущего каталога, затем производится сортировка по их размеру:

```
var   WA,           // экземпляр объекта Application
      WD;           // экземпляр объекта Document
var n, WshShell,  F, Fold, fname, fsize, i1,Cur, put;
    i1= 0; // кол-во подкаталогов
  // создаем объект WshShell
WshShell = WScript.CreateObject("WScript.Shell");
  // создаем объект FileSystemObject
FSO = WScript.CreateObject("Scripting.FileSystemObject");
F = FSO.GetFolder(WshShell.CurrentDirectory);
Fold = new Enumerator(F.SubFolders);
n= F.SubFolders.Count;
WScript.Echo("Количество каталогов  "+n);
fname= new Array(n);
fsize= new Array(n);
put= F.Path;
for (;!Fold.atEnd();Fold.moveNext()) {
    // извлекаем текущий элемента коллекции
  Cur=Fold.item();
  fname[i1]=Cur.Name;
  fsize[i1]= parseInt(Cur.Size/1024);
i1++ }
var wdCell=12,wdAlignParagraphLeft=0,wdAlignParagraphCenter=1,
    wdWindowStateMaximize=1;
  // создаем объект Application
WA=WScript.CreateObject("Word.Application");
WD=WA.Documents.Add();// создаем новый документ
WA.Visible=true; // делаем окно Winword видимым
WA.WindowState=wdWindowStateMaximize;   //   Максимизируем   окно
var count,text1,text2 ;
WA.Selection.TypeText("   В   директории   "+put+   "   существуют
следующие   каталоги: ");
WA.Selection.TypeText("************************************");
```

```
WA.ActiveDocument.Tables.Add(WA.Selection.Range,3,2,1,1);
WA.Selection.Font.Color = 255;
WA.Selection.TypeText("Каталог");
WA.Selection.MoveRight(wdCell);
WA.Selection.Font.Color = 255;
WA.Selection.TypeText("Размер в Кб");
WA.Selection.MoveRight(wdCell);
count= 0;
while (count!= n) {
      text1=fname[count];
      text2=fsize[count]+"";
      WA.Selection.TypeText(text1);
      WA.Selection.MoveRight(wdCell);
      WA.Selection.TypeText(text2);
      WA.Selection.MoveRight(wdCell);
      count++;
      WScript.Sleep(100);
}
WA.Selection.Sort(0,"столбцам 2",1,0,"",0,0,"",0,0,0,1,0);
```

Вторая, третья и четвертая работа должны находится в одном исходном файле. Программа будет обрабатывать аргументы командной строки, либо, при их отсутствии, предлагать диалог ввода. Протокол работы ведется как в текстовом файле, так и во внешнем приложении – Ms Word или Excel.

№	Задание лб № 2	Задание лб № 3	Задание лб № 4
1	Создание списка студенческих каталогов	Поля ввода шифра группы и кол-ва студентов	Word
2	Удаление файлов с определенным атрибутом	Поля ввода атрибутов	Excel
3	Удаление файлов определенного размера	Поле ввода размера	Word
4	Удаление файлов определенного расширения	Поле ввода расширения	Excel
5	Удаление файлов определенной даты модификации	Поле ввода даты	Word
6	Удаление каталогов определенного размера	Поле ввода максимального размера	Excel
7	Удаление каталогов в зависимости от времени модификации	Поле ввода даты и времени	Word
8	Копирование файлов с определенным атрибутом	Поле ввода атрибутов	Excel
9	Копирование файлов с определенным расширением	Поле ввода расширения	Word
10	Копирование файлов с определенным размером	Поле ввода минимального размера	Excel
11	Копирование файлов с определенным временем модификации	Поле ввода даты и времени	Word
12	Установка атрибутов файлов	Поле ввода атрибутов	Excel
13	Переименование файлов с определенным расширением	Поля ввода расширений	Word
14	Переименование файлов в верхний или нижний регистр	Поля ввода регистра	Excel
15	Поиск файлов, имеющий заданную дату создания	Поле ввода даты	Word

	(модификации)		
16	Найти на диске N последних файлов и поместить их в новый каталог	Поля ввода диска и числа файлов N	Excel
17	Найти на диске файлы нулевого размера и удалить их	Поле выбора диска	Word
18	Найти на диске файлы за последний день, неделю, месяц и поместить их в новый каталог	Поле ввода диапазона дат	Excel
19	Найти на диске файлы максимального и минимального размера, подсчет среднего размера	Поле выбора диска	Word
20	Определить размер неиспользуемого дискового пространства (свободное место на диске минус суммарный объем всех файлов)	Поле выбора диска	Excel
21	Обновление программных файлов в зависимости от номера версии (GetFileVersion)	Поля выбора каталогов	Word
22	Добавление информации в текстовые файлы	Поля выбора файлов	Excel
23	Отчет об использовании дискового пространства	Поле выбора диска	Word
24	Получение и установка переменных среды	Поля ввода переменных среды	Excel
25	Поиск файлов по маске в имени файла	Поле ввода маски	Word
26	Подсчитать общий размер файлов с определенными расширениями (doc, mp2, avi и др.) в данном каталоге и его подкаталогах	Поле ввода расширений	Excel

В заданиях с дисками предусматривается полный перебор всех файлов (см. рекурсивную функцию LoopSubFolders в примере Попова Chapter05/DelTmp.js).

ЛИТЕРАТУРА

1. Попов А. В. Командные файлы и сценарии Windows Script Host
2. Попов А.В. Windows Script Host. – СПб.: БХВ-Петербург
3. Торрес Дж. Скрипты для администратора Windows. Специальный справочник
4. Борн Г. Руководство разработчика на Microsoft Windows Script Host 2.0. Мастер-класс (+CD)/ Пер. с англ
5. Экк Сценарии ADSI для системного администрирования Windows