**Лабораторная работа 19**

**Сценарии Windows PowerShell**

**Инструкция If … Elseif … Else**

Синтаксис инструкции If в общем случае имеет следующий вид:

if (условие1)

{блок\_кода1}

[elseif (условие2)

{блок\_кода2}]

[else

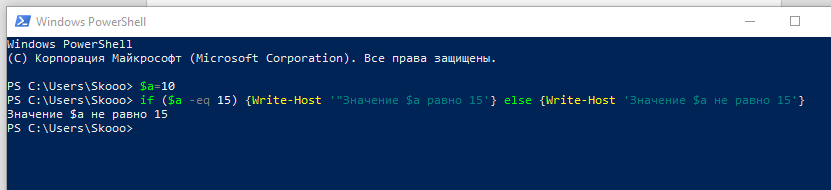
{блок\_кода3}]

Приведем пример использования инструкции If в интерактивном режиме работы. Запишем сначала в переменную $a число 10:

PS C:\> $a=10

Сравним теперь значение переменной $a с числом 15 (в нашем примере используется командлет Write-Host, который выводит информацию на экран):

PS C:\> if ($a -eq 15) {Write-Host '"Значение $a равно 15'} else {Write-Host 'Значение $a не равно 15'}



**Цикл While**

while (условие){блок\_команд}

while($val -ne 3)

{

$val++

Write-Host $val

}

В данном примере условие (значение переменной $val не равно 3) имеет значение true, пока $val равно 0, 1 или 2. При каждом повторении цикла значение $val увеличивается на 1 с использованием *унарного оператора* увеличения значения ++ ( $val++ ). При последнем выполнении цикла значением $val становится 3. При этом проверяемое условие принимает значение false, и цикл завершается.

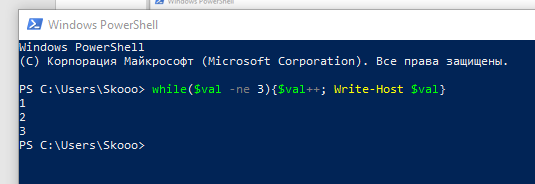
При интерактивной работе команды удобнее записывать в одной строке, например:

PS C:\> while($val -ne 3){$val++; Write-Host $val}

1

2

3



**Цикл Do … While**

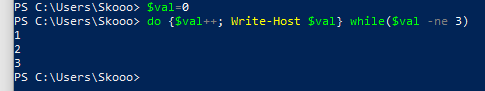
Цикл Do…While похож на цикл While, однако условие в нем проверяется не до блока команд, а после:

do{блок\_команд}while (условие)

Например:

PS C:\> $val=0

PS C:\> do {$val++; Write-Host $val} while($val -ne 3)

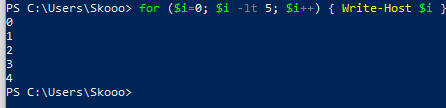


**Цикл For**

for (инициация; условие; повторение){блок\_команд}

Классический пример:

PS C:\> for ($i=0; $i -lt 5; $i++) { Write-Host $i }



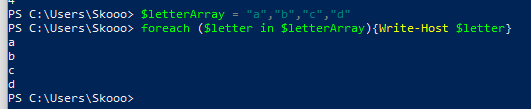
**Цикл Foreach**

foreach ($элемент in $коллекция){блок\_команд}

Например, цикл Foreach в следующем примере отображает значения в массиве с именем $letterArray:

PS C:\> $letterArray = "a","b","c","d"

PS C:\> foreach ($letter in $letterArray){Write-Host $letter}



PS C:\> $l = 0; foreach ($f in dir \*.txt) { $l += $f.length }

PS C:\> $l



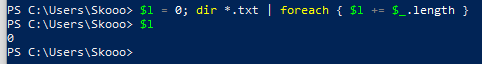
Синтаксис инструкции Foreach, применяемой внутри конвейера команд, в простейшем случае выглядит следующим образом:

команда | foreach {блок\_команд}

Рассмотренный выше пример с подсчетом суммарного размера текстовых файлов из текущего каталога для данного варианта инструкции Foreach примет следующий вид:

PS C:\> $l = 0; dir \*.txt | foreach { $l += $\_.length }

PS C:\> $l



Синтаксис псевдонима Foreach, используемого в конвейере команд с начальным, средним и конечным блоками команд выглядит следующим образом:

команда | foreach {начальный\_блок\_команд}

{средний\_блок\_команд}{конечный\_блок\_команд}

Для этого варианта инструкции Foreach наш пример можно записать следующим образом:

PS C:\> dir \*.txt | foreach {$l = 0}{ $l += $\_.length }{Write-Host $l}



**Запуск сценариев PowerShell**

Давайте запустим наш первый сценарий PowerShell. Для этого создадим с помощью Блокнота Windows текстовый файл test.ps1 и запишем в него одну строку:

Write-Host "Эта строка печатается из сценария PowerShell"

Сохраним этот файл в каталоге C:\Script. Как мы помним, сценарии WSH на языках VBScript или JScript можно было запускать из Проводника Windows, щелкая мышью на значках этих сценариев. Со сценариями PowerShell этот метод не работает – если дважды щелкнуть мышью на значке сценария PowerShell, то он не запустится, а откроется для редактирования в Блокноте (это позволяет предотвратить случайный запуск сценария).

Попробуем запустить созданный нами сценарий из оболочки PowerShell:

PS C:\> C:\script\test.ps1



Не удается загрузить файл C:\script\test.ps1, так как выполнение сценариев запрещено для данной системы. Введите "get-help about\_signing" для получения дополнительных сведений.

В строка:1 знак:18

+ C:\script\test.ps1 <<<<

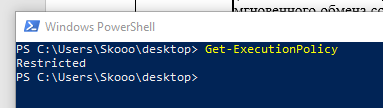
**Политики выполнения сценариев**

|  |  |
| --- | --- |
| Таблица 19.1. Политики выполнения PowerShell | |
| **Название политики** | **Описание** |
| Restricted | Данная политика выполнения используется по умолчанию, она запрещает выполнение сценариев и загрузку конфигурационных файлов (можно пользоваться только одиночными командами PowerShell в интерактивном режиме) |
| AllSigned | Выполнение сценариев PowerShell разрешено, однако все сценарии (как загруженные из Интернета, так и локальные) должны иметь цифровую подпись надежного издателя. Перед выполнением сценариев надежных издателей запрашивается подтверждение |
| RemoteSigned | Выполнение сценариев PowerShell разрешено, при этом все сценарии и файлы конфигураций, загруженные из Интернета (в том числе по электронной почте и с помощью программ мгновенного обмена сообщениями), должны иметь цифровую подпись надежного издателя, а локальные сценарии могут быть неподписанными. При запуске сценариев надежных издателей подтверждение не запрашивается |
| Unrestricted | Разрешается выполнение любых сценариев PowerShell без проверки цифровой подписи. При запуске сценариев и файлов конфигураций, загруженных из Интернета, выдается предупреждение |

Узнать, какая политика выполнения является активной, можно с помощью командлета Get-ExecutionPolicy:

PS C:\> Get-ExecutionPolicy

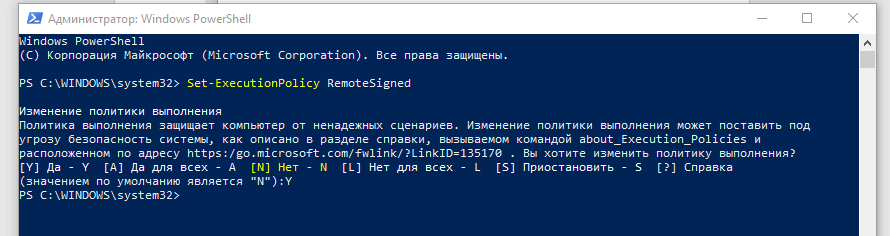
Restricted



Командлет Set-ExecutionPolicy позволяет сменить политику выполнения. Например, для установки политики выполнения RemoteSignedнужно выполнить следующую команду:

Set-ExecutionPolicy RemoteSigned

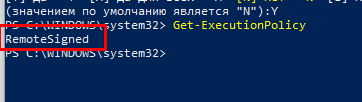




Проверим снова текущую политику:

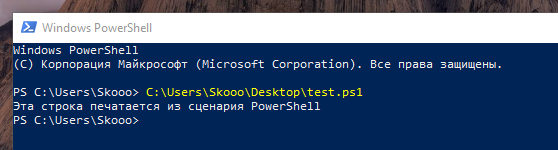
PS C:\> Get-ExecutionPolicy

RemoteSigned



Теперь вновь попробуем запустить наш сценарий:

PS C:\> C:\script\test.ps1



Эта строка печатается из сценария PowerShell

Как мы видим, сценарий test1.ps1 успешно выполнен.

При запуске сценариев PowerShell следует учесть, что путь к файлу с кодом нужно всегда указывать явно, даже если сценарий находится в текущем каталоге. В противном случае сценарий не будет выполнен, например:

PS C:\> cd script

PS C:\script> dir

Каталог: Microsoft.PowerShell.Core\FileSystem::C:\script

Mode LastWriteTime Length Name

---- ------------- ------ ----

-a--- 09.12.2016 17:37 59 test.ps1

