**Краткий учебник основ Powershell**

Несколько лет назад мною писались заметки по **Powershell**.

В данном документе я их систематизирую и актуализирую.

На дворе конец 2016 года.

1. **Знакомство**

Основная конструкция языка глагол-существительное.

**Простые примеры:**

Get-Help – вывести справку.

Get-Process – вывести список процессов.

Get-Command – вывести команды.

Передача параметров происходит через дефис «-».

**Вывод команд с глаголом «Get»:**

Get-Command –Verb Get

**Вывод команд с существительным «Event»:**

Get-Command –Noun event

**Неявная передача параметра команде Get-help (справка по команде):**

Get-Help Get-Event

Get-Help Get-Event –Detailed

Get-Help Get-Event -Full

Get-Help Get-Event -Examples

**Список модулей в Powershell:**

Get-Module –ListAvailable

**Вывести все команды из модуля «NetTCPIP»:**

Get-Command –Module NetTCPIP

**Обновить справку(необходимо подключение к интернету и права администратора):**

Update-Help

2. **Конвейер и управление сервисами**

**Вывод порционно:**

Get-Command | more

**С сортировкой по столбцу Version:**

Get-Command | Sort-Object Version

**С сортировкой + вывод первых 50 строк:**

Get-Command | Sort-Object Version | Select-Object -First 50

**Вывести свойства и методы объектов:**

Get-Service | Get-Member

**Вывести только методы:**

Get-Service | Get-Member -MemberType Method

**Перенаправление вывода в файл:**

Get-Service | Out-File "C:\test.txt"

**Управление сервисом(с правами администратора):**

**Выбрали сервис:**

Get-Service -Name AdobeARMservice

**Что мы можем с ним делать:**

Get-Command -Noun service

**Рестартуем сервис:**

Restart-Service AdobeARMservice

3. **Работа с файловой системой**

**Какие есть команды для работы с файловой системой:**

Get-Command -Noun item

**Посмотреть содержимое директории(аналогично dir):**

Get-ChildItem "C:\Program Files\7-Zip\"

Посмотреть все файлы по маске .exe, Recurse – вернет объекты, Force – покажет скрытые.

Get-ChildItem "C:\Program Files\7-Zip\" -Include *.exe -Recurse –Force

**Удалить файл:**

Remove-Item "C:\test.txt"

**Создать папку:**

New-Item -Path 'C:\temp\testfolder' -ItemType "directory"

**Создать файл:**

New-Item -Path 'C:\temp\testfolder\testfile.txt' -ItemType "file"

**Удалить папку и содержимое:**

Remove-Item 'C:\temp\testfolder' –Recurse

**Работа со столбцами(оба примера равнозначны):**

**Для обработки используется конструкция @{}.**

**Ей передается аргумент в виде названия столбца n="Kbytes";**

**Далее происходит обработка столбца e={$PSItem.Length/1024}**

**$PSItem и $_ - ссылки на текущий объект.**

Get-ChildItem "C:\Program Files\7-Zip\" | Select-Object -Property Name, Length, @{n="Kbytes"; e={$PSItem.Length/1024}}

Get-ChildItem "C:\Program Files\7-Zip\" | Select-Object -Property Name, Length, @{n="Kbytes"; e={$_.Length/1024}}

**Вывод по маске в столбце:**

Get-ChildItem "C:\Program Files\7-Zip\" | Where-Object Name -Like "7z*"

**Операторы сравнения:**

-eq Equal Равно

-ne Not equal Не Равно

-ge Greater than or equal Больше или Равно

-gt Greater than Больше

-lt Less than Меньше

-le Less than or equal Меньше или Равно

-like Wildcard comparison Использование символов подстановки для поиска соответствия образцу

-notlike Wildcard comparison Использование символов подстановки для поиска несоответствия образцу

-match Regular expression comparison Использование регулярных выражений для поиска соответствия образцу

-notmatch Regular expression comparison Использование регулярных выражений для поиска несоответствия образцу

-replace Replace operator Заменяет часть или все значение слева от оператора

-contains Containment operator Определение, содержит ли значение слева от оператора значение справа. В отличие от предыдущих операторов, результатом является булево значение

-notcontains Containment operator Определение того, что значение слева от оператора не содержит значение справа. Результатом является булево значение

**Вывод, где размер больше или равен с поиском по маске:**

Get-ChildItem "C:\Program Files\7-Zip\" | Where-Object -FilterScript {$_.Length -ge 150000 -and $_.Name -Like "7z.*"}

**Вывод с общего занятого места объектами, с выводом максимального размера файла:**

Get-ChildItem "C:\Program Files\7-Zip\" | Measure-Object -Property Length -Sum –Maximum

**Вывод в форматированном виде, табуляция после Format-Table покажет варианты:**

Get-ChildItem "C:\Program Files\7-Zip\" | Format-Table FullName, Extension, Length, Attributes

4. **Программирование**

**Посмотреть список стандартных переменных:**

Get-Variable

**Вывести значение переменной:**

$PWD

**Типы переменных:**

[int] - целые числа

[single] - числа одинарной точности

[double] – числа двойной точности

[string] - текст

[char] - символ

[Boolean] – булево значение «Истина» \ «Ложь»

[datetime] - дата или время

**Зададим значения переменных и произведем действие:**

```
[int]$a = 10
[int]$b = 5
$a - $b
```

**Конструкция if (elseif и else – необязательные блоки) – проверка условия:**

```
[int]$a = 10
If ($a -eq 10){
   Write-Host "a = 10"
}
ELSEIF ($a -gt 10){
   Write-Host "a > 10"
   }
ELSE {
   Write-Host "a < 10"
}
```

**Конструкция switch – проверяет значение:**

```
[int]$a = 1
SWITCH ($a)
{
 0 {Write-Host "a = 0"}
 1 {Write-Host "a = 1"}
 2 {Write-Host "a = 2"}
 default {Write-Host "a - ?"}
}
```

**Цикл while – выполняется, пока условие верно:**

```
[int]$a = 1
WHILE ($a -lt 3){
   Write-Host $a
   $a = $a + 1
}
```

**Аналогичная задача с while для цикла for:**

```
FOR ([int]$a = 1; $a -lt 3; $a++){
   Write-Host $a
}
```

**Перебор с помощью цикла foreach:**

```
$Process = Get-Process
FOREACH ($proc In $Process){
   $proc.ProcessName + " - " + $proc.Id
}
```

**Массивы, создание, обращение, перебор:**

```
[int32[]]$Array = 1, 2, 3, 4
$Array[2]
FOR ([int]$i = 0; $i -lt $Array.count; $i++){
   $Array[$i]
}
```

Обработка исключений(ошибок), введите число или символ:

```
Try {
 [int]$num = Read-Host "Enter num"
 6 / $num
} catch {
 Write-Warning "Error"
}
```

**Функции:**

**Простой пример и вызов:**

```
Function Func{"func is work"}
Func
```

**Пример с аргументами:**

```
Function Func{"this is args: $Args"}
Func arg1 arg2
```

**Еще пример с аргументами:**

```
Function Func ($a, $b) {"this is args: $a $b"}
Func arg1 arg2
```

**Пример с предопределенными значениями:**

```
Function Func ($a="a1", $b="a2") {"this is args: $a $b"}
Func
```