

08.02.2024

Курс:

Практическая работа к уроку № Lesson_7

--

Автоматизация Active Directory

Задание:

Перед выполнением домашнего задания создайте резервные копии (снапшоты) виртуальных машин с домен контроллерами

1. Узнайте IP и MAC адреса компьютера используя конвеер Powershell
2. Установите оффлайн модуль VMware.PowerCLI
3. С помощью Powershell отфильтруйте все запущенные процессы
4. Запустите, потом остановите процесс notepad (блокнот)
5. Отфильтруйте все события с кодом 7036 за последний месяц журнала Система
6. Создайте задание в планировщике
7. Посмотрите счетчики процессора при работе в пользовательском режиме
8. Добавьте роль ADDS, установите Контроллер домена
9. На компьютере работающем в режиме Core поднимите резервный контроллер
10. Создайте пользователя, группу, OU. Добавьте пользователя в группу.
11. Установите роль DHCP сервера, создайте область, настройте её (Диапазон адресов, адреса DNS, gate). Посмотрите настройки.
12. Включите созданную область
13. Экспортируйте настройки DHCP сервера

Serv1: win2019serv01 (gui)

Параметры сетевого адаптера	
Индекс адаптера	1
Описание	Intel(R) PRO/1000 MT Desktop Adapter
IP-адрес	192.168.56.16 fe80::f814:7e3c:885d:9286
Маска подсети	255.255.255.0
DHCP включен	Ложь
Шлюз по умолчанию	192.168.56.4
Основной DNS-сервер	192.168.56.1
Альтернативный DNS-сервер	8.8.8.8

Serv2: win2019core01 (non-gui)

Задания из Урока

Команды:

```
Get-Process
Get-Service
Get-Help -Category cmdlet

Get-Process -Name *p
Get-Process mspaint, notep* | Format-List *
Get-Process mspaint, notep* | Select-Object ProcessName, StartTime,
MainWindowTitle, Path, Company |ft
Get-Process | Where-Object {$_ mainWindowTitle} | Format-Table Id, Name,
mainWindowTitle
Get-Process -Name not* -IncludeUserName
Get-Process | Where-Object {$_ WorkingSet -GT 20000*1024} | select
processname,@{l="Used RAM(MB)"; e={$_ workingset / 1mb}} | sort "Used
RAM(MB)" -Descending

Get-Process | Get-Member
Get-help Get-Process -Examples
Start-Process Notepad
Start Notepad
Stop-Process -Name Notepad
Get-WindowsDriver -Online -all
Get-Command -Noun Process

Get-Service | Sort-Object -property Status
Get-Service | WHERE {$_ status -eq "Running"} | SELECT displayname

Start-Process -FilePath ping -ArgumentList "192.168.56.17"
Get-Process -ComputerName w2019serv02 | Form-Table -Property ProcessName,
ID, MachineName

#index, Time, InstanceID ...
Get-EventLog -LogName "System"
```

```

Get-EventLog -LogName "System" | Format-Table EntryType, TimeWritten,
Source, EventID, Category, Message
Get-EventLog -LogName 'System' -Before '10 февраля 2024'
Get-EventLog -LogName 'System' -After '1 января 2024' -Before '10 февраля
2024'
Get-EventLog -LogName 'System' -Newest 20

$event Get-EventLog -LogName 'System' -Newest 20
$event | Sort-Object -Property EventID, Message
...
$event | Where-Object {$__.EntryType -match 'error'}
...

#Установка ролей
Get-WindowsFeatures
Get-WindowsFeatures | Where-Object {$__. installstate -eq 'installed'} | ft
Name,Installstate
Get-WindowsFeatures -Name net-* | Where Installed

#Установка контроллера
Get-WindowsFeature -Name *ad*
Install-WindowsFeature -Name AD-Domain-Services -IncludeManagementTools
...

# Funktion
function Get-CPUPercent
...

$p = Get-Process
$p | SELECT Name, Path | Export-Csv -Path c:\out_process1.csv

$h = Get-Process
$h | ConvertTo-Html Property Name, Path | Export-Csv -Path
c:\out_process2.html

```

PowerShell

Process

```

PS C:\Users\Администратор> Get-Process mspaint, notep* | Format-List *

Name          : mspaint
Id            : 3044
PriorityClass : Normal
FileVersion   : 10.0.17763.1 (WinBuild.160101.0800)
HandleCount   : 306
WorkingSet    : 29966336
PagedMemorySize : 8921088
PrivateMemorySize : 8921088
VirtualMemorySize : 209121280
TotalProcessorTime : 00:00:00.4843750
SI            : 2
User          : 205

```

```
PS C:\Users\Администратор> Get-Process mspaint, notepad | Select-Object ProcessName, StartTime, MainWindowTitle, Path, Company | ft
ProcessName StartTime MainWindowTitle Path Company
----- ----- -----
mspaint 10.02.2024 15:57:54 Безымянный - Paint C:\Windows\system32\mspaint.exe Microsoft Corporation
notepad 10.02.2024 15:58:03 Безымянный - Блокнот C:\Windows\system32\notepad.exe Microsoft Corporation
```

```
PS C:\Users\Администратор> Get-Process | Where-Object {$_['mainwindowtitle']} | Format-Table Id, Name, mainWindowTitle
Id Name MainWindowTitle
-- -- --
3044 mspaint Безымянный - Paint
2844 notepad Безымянный - Блокнот
3700 powershell Администратор: Windows PowerShell
3988 ServerManager Диспетчер серверов
```

```
PS C:\Users\Администратор> Get-Process | Where-Object {$_['WorkingSet -gt 20000*1024} | select processname,@{l="Used RAM(MB)"; e={$_['WorkingSet / 1mb}} | sort "Used RAM(MB)" -Descending
ProcessName Used RAM(MB)
----- -----
powershell 161,59765625
MsMpEng 150,19140625
svchost 93,5546875
sme 87,09765625
explorer 78,58984375
SearchUI 75,72265625
Registry 69,73046875
svchost 65,14453125
ServerManager 61,8046875
dwm 53,87890625
LogonUI 44,7578125
ShellExperienceHost 40,5390625
svchost 34,40625
dwm 34,31640625
svchost 32,7109375
svchost 29,85546875
mspaint 28,6328125
svchost 27,089375
svchost 25,33984375
sihost 23,96484375
WmiPrvSE 23,171875
svchost 22,95703125
smartscreen 22,390625
svchost 19,8125
conhost 19,77734375
```

PowerShell ISE

Безымянный1.ps1*

```
function Get-CPUPercent
{
    $CPUPercent = @{
        Name = 'CPUPercent'
        Expression = {
            $TotalSec = (New-TimeSpan -Start $_.StartTime).TotalSeconds
            [Math]::Round( ($_.$CPU * 100 / $TotalSec), 2)
        }
    }
    Get-Process | Select-Object -Property Name, $CPUPercent, Description | Sort-Object -Property CPUPercent -Descending
}
Get-CPUPercent
```

PS C:\Users\Администратор> function Get-CPUPercent
{
 \$CPUPercent = @{
 Name = 'CPUPercent'
 Expression = {
 \$TotalSec = (New-TimeSpan -Start \$_.StartTime).TotalSeconds
 [Math]::Round((\$_.\$CPU * 100 / \$TotalSec), 2)
 }
 }
 Get-Process | Select-Object -Property Name, \$CPUPercent, Description | Sort-Object -Property CPUPercent -Descending |
}
Get-CPUPercent

Name	CPUPercent	Description
conhost	0	Хост окна консоли
sme	0	Windows Admin Center Windows Service
smss	0	
spoolsv	0	Диспетчер очереди печати
svchost	0	Хост-процесс для служб Windows

```
function Get-CPUPercent
{
```

```

$CPUPercent = @{
    Name = 'CPUPercent'
    Expression = {
        $TotalSec = (New-TimeSpan -Start $_.StartTime).TotalSeconds
        [Math]::Round( ($_.CPU * 100 / $TotalSec), 2)
    }
}
Get-Process | Select-Object -Property Name, $CPUPercent, Description |
Sort-Object -Property CPUPercent -Descending | Select-Object -First 20
}
Get-CPUPercent

```

PS C:\Users\Администратор> Start-Process -FilePath ping -ArgumentList "192.168.56.103"
PS C:\Users\Администратор> Start-Process -FilePath ping -ArgumentList "192.168.56.103" -wait -windowstyle Maximized

7-1.ps1* X

1 Get-Process -Name *p

Команды X

Модули: Все

Имя:

A:

- Add-AppvClientConnectionGroup
- Add-AppvClientPackage
- Add-AppvPublishingServer
- Add-AppxPackage
- Add-AppxProvisionedPackage
- Add-AppVolume
- Add-AttestationServiceInfo
- Add-BCDDataCacheExtension
- Add-BitsFile
- Add-CertificateEnrollmentPolicyServer
- Add-ClDatastore
- Add-ClusterSCSITargetServerRole
- Add-Computer
- Add-Content
- Add-CustomCertificate
- Add-DeployRule
- Add-DfsrConnection
- Add-DfsrMember

PS C:\Users\Администратор> Get-Process -Name *p

Handles	NPM(K)	PM(K)	WS(K)	CPU(s)	Id	SI	ProcessName
324	13	2480	10884	0,11	2588	2	rdpclip
147	8	1672	7104	0,03	2612	0	WMIADAP

PS C:\Users\Администратор>

Администратор: Windows PowerShell ISE

Файл Правка Вид Сервис Отладка Дополнительные компоненты Справка

7-1.ps1* Безымянный4.ps1* Безымянный5.ps1* Безымянный6.ps1*

```
1 cls
2 help
```

Для получения дополнительных сведений о командлете Update-Help необходимо ввести следующее:

```
Get-Help Update-Help -Online
```

или перейти по ссылке <http://go.microsoft.com/fwlink/?LinkID=210614>

GET-HELP

Командлет Get-Help отображает справку в командной строке, используя содержимое файлов справки, находящихся на компьютере. В отсутствие файлов справки Get-Help отображает базовые справочные сведения о командлентах и функциях. Get-Help также позволяет отображать справку по командлентам и функциям в Интернете.

Для получения справки по какому-либо командлету необходимо ввести следующее:

```
Get-Help <имя_командлета>
```

Для получения справки в Интернете необходимо ввести следующее:

```
Get-Help <имя_командлета> -Online
```

Названия разделов общих понятий начинаются с "About_".

Для получения справочных сведений о каком-либо понятии или элементе языка необходимо ввести следующее:

```
Get-Help About_<название_раздела>
```

Для поиска слова или фразы в файлах справки необходимо ввести следующее:

```
Get-Help <условие_поиска>
```

Администратор: Windows PowerShell ISE

Файл Правка Вид Сервис Отладка Дополнительные компоненты Справка

7-1.ps1* Безымянный4.ps1* Безымянный5.ps1* Безымянный6.ps1* Безымянный7.ps1* X

```
1 "Hello, World!" | Out-File C:\test.txt
2 C:\test.txt
3
```

Служба сенсорной клавиатуры и панели
Службы удаленных рабочих столов
Темы
Брокер времени
Диспетчер учетных веб-записей
Клиент отслеживания изменившихся связей
Установщик модулей Windows
Служба ведения журнала доступа пользователей
Перенаправитель портов пользовательского интерфейса
Диспетчер пользователей
Update Orchestrator Service
Служба времени Windows
Диспетчер подключений Windows
Узел системы диагностики
Служба проверки сети Windows Defender
Антивирусная программа "Защитника Windows"
Служба автоматического обнаружения вредоносного ПО
Инструментарий управления Windows
Служба удаленного управления Windows
Служба системы push-уведомлений Windows
Пользовательская служба push-уведомлений Windows
Центр обновления Windows

```
PS C:\Users\Администратор> "Hello, World!" | Out-File C:\test.txt
```

Администратор: Windows PowerShell ISE

Файл Правка Вид Сервис Отладка Дополнительные компоненты Справка

7-1.ps1* Безымянный4.ps1* Безымянный5.ps1* Безымянный6.ps1* Безымянный7.ps1* X

```
1 # > equal. | Out-File
2 "Hello, 2World!" > C:\test.txt
3 C:\test.txt
4
```

Служба проверки сети Windows нет
Антивирусная программа "Защитник"

```
1 Get-Service | Sort-Object -property Status
2
```

Status	Name	DisplayName
Stopped	AJRouter	Служба маршрутизатора AllJoyn
Stopped	RasAuto	Диспетчер автоматических подключений
Stopped	QWAVE	Quality Windows Audio Video Experience
Stopped	PushToInstall	Служба PushToInstall Windows
Stopped	WSearch	Windows Search
Stopped	PrintWorkflowUs...	PrintWorkflow_413cd
Stopped	PrintNotify	Расширения и уведомления для принтеров
Stopped	VaultEcs	Локальные хранилища

```
7-1.ps1  Безымянный4.ps1  Безымянный5.ps1
1 $var=615
2 $var
3 $b=55
4 $var+$b
<
PS C:\Users\Администратор> $var=615
$var
$b=55
$var+$b
615
670
PS C:\Users\Администратор>
```

```
1 $str = "Hello"
2 $str = $str + " World"
3 $str
<
PS C:\Users\Администратор> $str = "Hello"
$str = $str + " World"
$str
Hello World
```

```
7-1.ps1* Безымянный4.ps1* Безымянный5.ps1* Безымянн
1 $str = $str + 2024
2 $str
<
PS C:\Users\Администратор> $str = $str + 2024
$str
Hello World2024
```

```
7-1.ps1* Безымянный4.ps1* Безымянный5.ps1* Безымянн
1 $s = "is it a String?"
2 $s.GetType().FullName
<
PS C:\Users\Администратор> $s = "is it a String?"
$s.GetType().FullName
System.String
```

```
7-1.ps1* Безымянный4.ps1* Безымянный5.ps1* Безымянн
1 $str1 = "Abc"
2 $str2 = "abc"
3 $str1 -ceq $str2 #false
<
PS C:\Users\Администратор> $str1 = "Abc"
$str2 = "abc"
$str1 -ceq $str2 #false
False
```

```
1 $str1 = "Abc"
2 $str2 = "abc"
3 $str1 -eq $str2 #false
```

```
PS C:\Users\Администратор> $str1 = "Abc"
$str2 = "abc"
$str1 -ceq $str2 #false
False

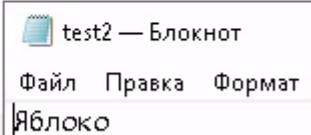
PS C:\Users\Администратор> $str1 = "Abc"
$str2 = "abc"
$str1 -eq $str2 #false
True
```

Безымянныи0.ps1 | Безымянныи1.ps1

```
1 $str = ""
2 if ( 1 -gt 2 ) { $str = "Апельсин" }
3 elseif ( $str -eq "Апельсин" ) { $str }
4 else { $str = "Яблоко"; $str }
5 $str > c:\test2.txt
6 c:\test2.txt
```

```
PS C:\Users\Администратор> $str = ""
if ( 1 -gt 2 ) { $str = "Апельсин" }
elseif ( $str -eq "Апельсин" ) { $str }
else { $str = "Яблоко"; $str }
$str > c:\test2.txt
c:\test2.txt
Яблоко

PS C:\Users\Администратор>
```



Безымянныи8.ps1* | Безымянныи9.ps1 | Безымянныи10.ps1*

```
1 $p = Get-Process
2 $p | SELECT Name, Path | Export-Csv -Path c:\out_process1.csv
```

```
PS C:\Users\Администратор> $p = Get-Process
$p | SELECT Name, Path | Export-Csv -Path c:\out_process1.csv
```

```
1 $h = Get-Process
2 $h | ConvertTo-Html Property Name, Path | Export-Csv -Path c:\out_process2.html

PS C:\Users\Администратор> $h = Get-Process
$h | ConvertTo-Html Property Name, Path | Export-Csv -Path c:\out_process2.html
```

безымянныиурс1 | безымянныиурс2 | безымянныиурс3 | безымянныиурс4 | безымянныиурс5 | безымянныиурс6 | Модули
Имя: А:

```
PS C:\Users\Администратор> Get-Content C:\out_process1.csv
#TYPE Selected.System.Diagnostics.Process
"Name","Path"
"conhost","C:\Windows\system32\conhost.exe"
"conhost","C:\Windows\system32\conhost.exe"
"csrss",
"csrss",
"csrss",
"ctfmon","C:\Windows\system32\ctfmon.exe"
"dwm","C:\Windows\system32\dwm.exe"
"dwm","C:\Windows\system32\dwm.exe"
"explorer","C:\Windows\Explorer.EXE"
"fontdrvhost","C:\Windows\system32\fontdrvhost.exe"
"fontdrvhost","C:\Windows\system32\fontdrvhost.exe"
"fontdrvhost","C:\Windows\system32\fontdrvhost.exe"
"Idle",
"LogonUI","C:\Windows\system32\LogonUI.exe"
"lsass","C:\Windows\System32\lsass.exe"
"msdtc","C:\Windows\System32\msdtc.exe"
"MsMpEng",
"MsSrv",
"powershell","C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe"
"powershell_isse","C:\Windows\System32\WindowsPowerShell\v1.0\PowerShell_ISE.exe"
"rdclip","C:\Windows\System32\rdclip.exe"
"Registry",
"RuntimeBroker","C:\Windows\System32\RuntimeBroker.exe"
"RuntimeBroker","C:\Windows\System32\RuntimeBroker.exe"
"SearchUI","C:\Windows\SystemApps\Microsoft.Windows.Cortana_cw5n1h2txyewy\SearchUI.exe"
"ServerManager","C:\Windows\system32\ServerManager.exe"
```

out_process1 — Блокнот

Файл Правка Формат Вид Справка

#TYPE Selected.System.Diagnostics.Process

"Name","Path"

"conhost","C:\Windows\system32\conhost.exe"

"conhost","C:\Windows\system32\conhost.exe"

"csrss",

"csrss",

"csrss",

"ctfmon","C:\Windows\system32\ctfmon.exe"

"dwm","C:\Windows\system32\dwm.exe"

"dwm","C:\Windows\system32\dwm.exe"

"explorer","C:\Windows\Explorer.EXE"

"fontdrvhost","C:\Windows\system32\fontdrvhost.exe"

"fontdrvhost","C:\Windows\system32\fontdrvhost.exe"

"fontdrvhost","C:\Windows\system32\fontdrvhost.exe"

"Idle",

"LogonUI","C:\Windows\system32\LogonUI.exe"

безымянныиурс1 | безымянныиурс2 |

```
1 $x=5
2 do
3 {
4     $x
5     $x = $x - 1
6 }
7 while ($x -gt 0)
```

безымянныиурс1 | безымянныиурс2 |

```
PS C:\Users\Администратор> $x=5
do
{
    $x
    $x = $x - 1
}
while ($x -gt 0)
5
4
3
2
1
```

безымянныиурс1 | безымянныиурс2 |

```
1 $x=0
2 do
3 {
4     $x
5     $x = $x + 1
6 }
7 until ($x -gt 6)
```

безымянныиурс1 | безымянныиурс2 |

```
PS C:\Users\Администратор> $x=0
do
{
    $x
    $x = $x + 1
}
until ($x -gt 6)
0
1
2
3
4
5
6
```

```
1  for ($i = 0; $i < 7; $i++)  
2  {  
3      $i  
4  }
```

```
PS C:\Users\Администратор> for ($i = 0; $i -lt 7; $i++)  
{  
    $i  
}  
0  
1  
2  
3  
4  
5  
6
```

```
1 $collection = "name1", "name2", "name3"
2 foreach($item in $collection)
3 {
4     $item
5 }
```

```
PS C:\Users\Администратор> $collection = "name1", "name2", "name3"
foreach($item in $collection)
{
    $item
}
name1
name2
name3
```

```
1 function sqr ($a)
2 {
3     return $a * $a
4 }
5 sqr 3
```

```
PS C:\Users\Администратор> function sqr ($a)
{
    return $a * $a
}
sqr 3
9
```

Задание_1:

Узнайте IP и MAC адреса компьютера используя конвеер Powershell

```
Get-WmiObject win32_networkadapterconfiguration | select description, macaddress
```

```
getmac /S w2019serv01 /FO CSV | ConvertFrom-Csv
```

```
PS C:\Users\Администратор.W2019SERV01> Get-WmiObject Win32_NetworkAdapterConfiguration | select Description, MACAddress  
Description  
-----  
Microsoft Kernel Debug Network Adapter  
Intel(R) PRO/1000 MT Desktop Adapter 08-00-27-DC-48-68
```

```
PS C:\Users\Администратор.W2019SERV01> getmac /S w2019serv01 /FO CSV | ConvertFrom-Csv  
Физический адрес Имя транспорта  
----- -----  
08-00-27-DC-48-68 \Device\Tcpip_{14E95590-E8F1-4FAF-A64F-5C4DAE0E1123}
```

Задание_2:

Установите оффлайн модуль VMware.PowerCLI

Serv1:

```
Get-Module -ListAvailable  
Find-Module -Name VMware.PowerCLI  
(Save-Module -Name VMware.PowerCLI -Path C:\path-to-folder)  
($env: PSModulePath)  
Install-Module -Name VMware.PowerCLI  
Get-ExecutionPolicy  
Import-Module VMware.PowerCLI  
Get-VICommand  
Get-Command  
Get-Command *vm
```

```
PS C:\Users\Администратор> Get-Module -ListAvailable  
  
Каталог: C:\Program Files\WindowsPowerShell\Modules  
  
ModuleType Version Name ExportedCommands  
---- -- -- --  
Script 1.0.1 Microsoft.PowerShell.Operation.W... {Get-OperationValidation, Invoke-OperationValidation}  
Binary 1.0.0.1 PackageManagement {Find-Package, Get-Package, Get-PackageProvider, Get-Packa...}  
Script 3.4.0 Pester {Describe, Context, It, Should...}  
Script 1.0.0.1 PowerShellGet {Install-Module, Find-Module, Save-Module, Update-Module...}  
Script 2.0.0 PSReadline {Get-PSReadLineKeyHandler, Set-PSReadLineKeyHandler, Remov...
```

```

PS C:\Users\Администратор> Find-Module -Name VMware.PowerCLI
Для продолжения требуется поставщик NuGet
Для взаимодействия с репозиториями на основе NuGet модулю PowerShellGet требуется версия поставщика NuGet "2.8.5.201" и
и более новая. Поставщик NuGet должен быть доступен в "C:\Program
Files\PackageManagement\ProviderAssemblies" или "C:\Users\Администратор\AppData\Local\PackageManagement\ProviderAssemblies". Поставщик NuGet можно также установить, выполнив команду
"Install-PackageProvider -Name NuGet -MinimumVersion 2.8.5.201 -Force". Вы хотите, чтобы модуль PowerShellGet установил
и импортировал поставщик NuGet прямо сейчас?
[Y] Да - Y [N] Нет - N [S] Приостановить - S [?] Справка (значением по умолчанию является "Y"): у

Version      Name          Repository      Description
-----      ----          -----          -----
13.2.1....  VMware.PowerCLI    PSGallery      This Windows PowerShell module contains VMware.Power
CLI

PS C:\Users\Администратор> Install-Module -Name VMware.PowerCLI
Ненадежный репозиторий
Идет установка модулей из ненадежного репозитория. Если вы доверяете этому репозиторию, измените его значение
InstallationPolicy, запустив командлет Set-PSRepository. Вы действительно хотите установить модули из "PSGallery"?
[Y] Да - Y [A] Да для всех - A [N] Нет - N [L] Нет для всех - L [S] Приостановить - S [?] Справка
(значением по умолчанию является "N"):
```

```

(значением по умолчанию является "N")
PS C:\Users\Администратор> Get-ExecutionPolicy
RemoteSigned
PS C:\Users\Администратор> Import-Module VMware.PowerCLI
ПРЕДУПРЕЖДЕНИЕ: Please consider joining the VMware Customer Experience Improvement Program, so you can help us make
PowerCLI a better product. You can join using the following command:

Set-PowerCLIConfiguration -Scope User -ParticipateInCEIP $true

VMware's Customer Experience Improvement Program ("CEIP") provides VMware with information that enables VMware to
improve its products and services, to fix problems, and to advise you on how best to deploy and use our products. As
part of the CEIP, VMware collects technical information about your organization's use of VMware products and services
on a regular basis in association with your organization's VMware license key(s). This information does not
personally identify any individual.

For more details: type "help about_ceip" to see the related help article.

To disable this warning and set your preference use the following command and restart PowerShell:
Set-PowerCLIConfiguration -Scope User -ParticipateInCEIP $true or $false.
    Welcome to VMware PowerCLI!

Log in to a vCenter Server or ESX host:           Connect-VIServer
To find out what commands are available, type:   Get-VICmd
To show searchable help for all PowerCLI commands: Get-PowerCLIHelp
Once you've connected, display all virtual machines: Get-VM
If you need more help, visit the PowerCLI community: Get-PowerCLICommunity

    Copyright (C) VMware, Inc. All rights reserved.
```

Задание_3:

С помощью Powershell отфильтруйте все запущенные процессы

```

Get-Process | Where-Object {$_.WorkingSet -GT 0*1024} | select
processname,@{l="Used RAM(MB)"; e={$_.WorkingSet / 1mb}} | sort "Used
RAM(MB)" -Descending
```

```
Get-Process | Sort WS -Descending | Select Name,WS -First 50
```

```
PS C:\Users\Администратор.W2019SERV01> Get-Process | where-object {$_.WorkingSet -gt 0} | select processname,@{l="Used RAM(MB)"; e={$_.WorkingSet / 1mb}} | sort "Used RAM(MB)" -Descending
ProcessName      Used RAM(MB)
-----
powershell      524,19140625
MsMpEng        191,83984375
dns            123,2109375
svchost         93,9296875
explorer        90,9453125
Registry        80,94921875
svchost         69,53125
dwm             51,8671875
ShellExperienceHost 50,09765625
SearchUI         49,59375
LogonUI          45,51953125
dwm             34,3828125
svchost         32,2265625
svchost         30,11328125
svchost         28,390625
RuntimeBroker    27,59375
svchost         27,23828125
```

```
PS C:\Users\Администратор.W2019SERV01> Get-Process | Sort WS -Descending | Select Name,WS -First 50
Name           WS
---
powershell     549011456
MsMpEng       189202432
dns           129191936
svchost        98631680
explorer       95264768
Registry       84885504
svchost        68730880
dwm            54112256
ShellExperienceHost 52531200
SearchUI        52002816
LogonUI         47730688
dwm            36052992
```

Задание_4:

Запустите, потом остановите процесс notepad (блокнот)

```
Start-Process Notepad
# Start Notepad
Get-Process -Name Notepad
Stop-Process -Name Notepad
```

```
Administrator: Windows PowerShell
PS C:\Users\Администратор.W2019SERV01> start notepad
PS C:\Users\Администратор.W2019SERV01> Get-Process -Name Notepad
Handles  NPM(K)  PM(K)  WS(K)  CPU(s)  Id  SI ProcessName
-----  -----  -----  -----  -----  --  -- -----
  251     14    2992   15704    0,17  2044  2 notepad

PS C:\Users\Администратор.W2019SERV01> Start-Process Notepad
PS C:\Users\Администратор.W2019SERV01> Get-Process -Name Notepad
Handles  NPM(K)  PM(K)  WS(K)  CPU(s)  Id  SI ProcessName
-----  -----  -----  -----  -----  --  -- -----
  251     14    2992   15704    0,17  2044  2 notepad
  251     14    2992   15676    0,14  2260  2 notepad

PS C:\Users\Администратор.W2019SERV01> Stop-Process -Name Notepad
```

Задание_5:

Отфильтруйте все события с кодом 7036 за последний месяц журнала Система

```

Get-WinEvent -FilterHashtable @{"logname='System';id=7036;} | ft
ID,TimeCreated,Message

# 1 day
Get-WinEvent -FilterHashTable @{LogName='System'; StartTime=(get-
date).AddDays(-1); EndTime=(get-date).AddHours(-1); ID=7036}

```

```

PS C:\Users\Администратор.W2019SERV01> Get-WinEvent -FilterHashTable @{LogName='System'; StartTime=(get-date).AddDays(-1); EndTime=(get-date).AddHours(-1); ID=7036}

ProviderName: Service Control Manager
TimeCreated          Id LevelDisplayName Message
-----          -- -----
13.02.2024 2:25:37    7036 Сведения   Служба "Установщик модулей Windows" перешла в состояние Остановлена.
13.02.2024 2:24:46    7036 Сведения   Служба "Служба перечислителя переносных устройств" перешла в состояние Остановлена.
13.02.2024 2:24:32    7036 Сведения   Служба "Финансовая интеллектуальная служба передачи (BITS)" перешла в состояние Остановлена.
13.02.2024 2:24:15    7036 Сведения   Служба "Защита программного обеспечения" перешла в состояние Остановлена.
13.02.2024 2:24:03    7036 Сведения   Служба "Служба настройки сети" перешла в состояние Остановлена.

```

Задание_6:

Создайте задание в планировщике

```

#Задание для Планировщика заданий (Управление компьютера)
$Trigger = New-ScheduledTaskTrigger -At 10:00am -Daily
$user = 'NT AUTHORITY\SYSTEM'
$action = New-ScheduledTaskAction -Execute 'PowerShell.exe' -Argument
'C:\PS\StartUpaScript.ps1'
Register-ScheduledTask -TaskName 'StartupScript_PS' -Trigger $Trigger -
User $user -Action $action -RunLevel Highest -Force

```

Задание_7:

Посмотрите счетчики процессора при работе в пользовательском режиме

```

#Счетчик
(Get-Counter -ListSet процессор).counter
Get-Counter -Counter '\Процессор(*)\% работы в пользовательском режиме' -
SampleInterval 10 -MaxSamples 2

```

Задание_8:

Добавьте роль ADDS, установите Контроллер домена

_Serv01

```

Get-WindowsFeature -Name *ad*
Install-WindowsFeature -Name AD-Domain-Services -IncludeManagementTools
Get-Command -Module ADDS*
Install-ADDSForest -DomainName "domain.local01"

```

Password: Pass198

Display Name	Name	Install State
[X] Доменные службы Active Directory	AD-Domain-Services	Installed
[] Службы Active Directory облегченного доступа к к...	ADLDS	Available
[] Службы сертификатов Active Directory	AD-Certificate	Available
[] Центр сертификации	ADCS-Cert-Authority	Available
[] Веб-служба политик регистрации сертификатов	ADCS-Enroll-Web-Pol	Available
[] Веб-служба регистрации сертификатов	ADCS-Enroll-Web-Svc	Available
[] Сетевой ответчик	ADCS-Online-Cert	Available
[] Служба регистрации в центре сертификации чер...	ADCS-Web-Enrollment	Available
[] Служба регистрации на сетевых устройствах	ADCS-Device-Enrollment	Available
[] Службы управления правами Active Directory	ADRMS	Available
[] Сервер управления правами Active Directory	ADRMS-Server	Available
[] Поддержка федерации удостоверений	ADRMS-Identity	Available
[] Службы федерации Active Directory	ADFS-Federation	Available
[] Средства шифрования диска BitLocker	RSAT-Feature-Tools-B...	Available
[] Средство просмотра пароля восстановл...	RSAT-Feature-Tools-B...	Available
[X] Средства AD DS и AD LDS	RSAT-AD-Tools	Installed
[] AD LDS Snap-Ins and Command-Line Tools	RSAT-ADLDS	Available
[X] Модуль Active Directory для Windows ...	RSAT-AD-PowerShell	Installed
[X] Средства AD DS	RSAT-ADDS	Installed
[X] Оснастки и программы командной с...	RSAT-ADDS-Tools	Installed
[X] Центр администрирования Active D...	RSAT-AD-AdminCenter	Installed
[] Средства служб развертывания Windows	WDS-AdminPack	Available
[] Средства службы сертификации Active Direc...	RSAT-ADCS	Available
[] Средства управления центра сертификации	RSAT-ADCS-Mgmt	Available
[] Средства службы управления правами Activ...	RSAT-ADRMS	Available
[] Службы для средств управления NFS	RSAT-NFS-Admin	Available

PS C:\Users\Администратор.W2019SERV01> Install-WindowsFeature -Name AD-Domain-Services -IncludeManagementTools				
Success	Restart	Needed	Exit Code	Feature Result
True	No	NoChangeNeeded	{}	
PS C:\Users\Администратор.W2019SERV01> Get-Command -Module ADDS*				
CommandType	Name	Version	Source	
Cmdlet	Add-ADDSReadOnlyDomainControllerAccount	1.0.0.0	ADDSDeployment	
Cmdlet	Install-ADDSDomain	1.0.0.0	ADDSDeployment	
Cmdlet	Install-ADDSDomainController	1.0.0.0	ADDSDeployment	
Cmdlet	Install-ADDSForest	1.0.0.0	ADDSDeployment	
Cmdlet	Test-ADDSDomainControllerInstallation	1.0.0.0	ADDSDeployment	
Cmdlet	Test-ADDSDomainControllerUninstallation	1.0.0.0	ADDSDeployment	
Cmdlet	Test-ADDSDomainInstallation	1.0.0.0	ADDSDeployment	
Cmdlet	Test-ADDSForestInstallation	1.0.0.0	ADDSDeployment	
Cmdlet	Test-ADDSReadOnlyDomainControllerAccountCreation	1.0.0.0	ADDSDeployment	
Cmdlet	Uninstall-ADDSDomainController	1.0.0.0	ADDSDeployment	

- Install-ADDSForest -DomainName "domain.local01"

Message	Context	RebootRequired	Status
Операция выполнена успешно DCPromo.General.3		False	Success

Проверяем в настройках Администрирование - Active Directory - Domain Controllers (Serv1)

serv1_w2019serv01 x serv2_w2019serv02 non-gui x

Управление Администрирование

Файл Главная Поделиться Вид Средства работы с ярлыками

Имя

Active Directory — сайты и службы

Файл Действие Вид Справка

Имя Тип Описание

Active Directory — сайты и службы [v]
Sites
Subnets
Default-First-Site-Name
Servers
W2019SERV01
NTDS Settings
DNS Settings

PS C:\Users\Администратор.W2019SERV01> ipconfig

Настройка протокола IP для Windows

Адаптер Ethernet Ethernet:

DNS-суффикс подключения
Локальный IPv6-адрес канала : fe80::133f:2bdc:6754:e4ae%3
IPv4-адрес : 192.168.56.16
Маска подсети : 255.255.255.0
Основной шлюз. : 192.168.56.4

PS C:\Users\Администратор.W2019SERV01> nslookup domain.local01

Тип хоста: UnKnown
Address: ::1

Сервер: domain.local01
Addresses: 10.0.3.15
192.168.56.16

Serv02 (non-gui)

- DNS like Serv1 IP

- ping to Serv1 from Serv2

```
PS C:\Users\Администратор> ipconfig /all
```

Настройка протокола IP для Windows

```
Имя компьютера . . . . . : w2019core01
Основной DNS-суффикс . . . . . :
Тип узла . . . . . : Гибридный
IP-маршрутизация включена . . . . . : Нет
WINS-прокси включен . . . . . : Нет
Порядок просмотра суффиксов DNS . . . . . : fritz.box
```

Адаптер Ethernet Ethernet:

```
DNS-суффикс подключения . . . . . : 
Описание . . . . . : Intel(R) PRO/1000 MT Desktop Adapter
Физический адрес . . . . . : 08-00-27-D7-94-20
DHCP включен . . . . . : Нет
Автонастройка включена . . . . . : Да
Локальный IPv6-адрес канала . . . . . : fe80::b5d9:cf8c:f862:406e%7(Основной)
IPv4-адрес . . . . . : 192.168.56.17(Основной)
Маска подсети . . . . . : 255.255.255.0
Основной шлюз . . . . . : 192.168.56.4
IAID DHCPv6 . . . . . : 101187623
DUID клиента DHCPv6 . . . . . : 00-01-00-01-2D-5A-48-0B-08-00-27-D7-94-20
DNS-серверы . . . . . : 192.168.56.16
NetBIOS через TCP/IP . . . . . : Включен
```

ping 192.168.56.16

```
PS C:\Users\Администратор> ping 192.168.56.16
```

```
Обмен пакетами с 192.168.56.16 по с 32 байтами данных:  
Ответ от 192.168.56.16: число байт=32 время<1мс TTL=128  
Ответ от 192.168.56.16: число байт=32 время<1мс TTL=128  
Ответ от 192.168.56.16: число байт=32 время<1мс TTL=128
```

Статистика Ping для 192.168.56.16:

Пакетов: отправлено = 3, получено = 3, потеряно = 0
(0% потеря)

Serv02:

nslookup domain.local01

#Serv2

```
Add-Computer -DomainName "domain.local01" -restart
```

Администратор Pass198

```
#login as user domain\администратор
```

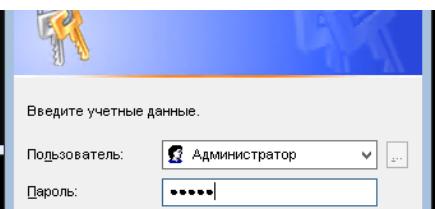
Powershell

```
Install-ADDSDomainController -InstallDns -DomainName "domain.local01"  
    а Для всех
```

Serv2

```
PS C:\Users\Администратор> Add-Computer -DomainName "domain.local01" -restart
```

Командлет Add-Computer в конвейере команд в позиции 1
Укажите значения для следующих параметров:
Credential



```
PS C:\Users\Администратор> nslookup domain.local01
```

Тип: Unknown
Address: 192.168.56.16

Name: domain.local01
Addresses: 192.168.56.16
 10.0.3.15

```
serv2_w2019serv02 non-gui x  
Выбрать Администратор: C:\Windows\system32\cmd.exe  
C:\Users\Администратор>whoami  
w2019core01\администратор
```

Задание 9-13

Serv01:

```
# Powershell  
Get-Command -module activedirectory  
  
New-ADOrganizationalUnit -name "1" -Path "DC=domain,DC=local01"  
...  
  
#add domain  
$fqdn = Get-ADDomain  
$fulldomain = $fqdn.DNSRoot  
$domain = $fulldomain.split(".")  
$Dom = $domain[0]  
$Ext = $domain[1]  
$Sites = ("SPB", "MSK", "Kaliningrad")  
$Services = ("Users", "Admins", "Computers", "Servers", "Contacts")  
$FirstOU = "Russia"  
New-ADOrganizationalUnit -Name $FirstOU -Description $FirstOU -Path  
"DC=$Dom,DC=$Ext" -ProtectedFromAccidentalDeletion $false  
foreach ($S in $Sites)  
{  
    New-ADOrganizationalUnit -Name $S -Description "$S" -Path  
    "OU=$FirstOU,DC=$Dom,DC=$Ext" -ProtectedFromAccidentalDeletion $false  
    foreach ($Serv in $Services)  
    {  
        New-ADOrganizationalUnit -Name $Serv -Description "$S $Serv" -Path
```

```

"OU=$S,OU=$FirstOU,DC=$Dom,DC=$EXT" -ProtectedFromAccidentalDeletion
\$false
}

}

#add user
New-ADUser -Name "Ivan Ivanov" -GivenName "Ivan" -Surname "Ivanov" -
SamAccountName "iivanov" -UserPrincipalName "iivanov@domain.local" -Path
"OU=1,DC=domain,DC=local01" -AccountPassword(Read-Host -AsSecureString
"Input Password") -Enabled $true
#Pa$$word1

#OU
$TargetOU = "OU=Russia,DC=domain, DC=local01"
Get-ADUser -Filter 'Name - like "a*' | Move-ADObject -TargetPath
$TargetOU

Get-ADComputer -SearchBase 'OU=Domain Controllers,DC=domain,DC=local01' -
Filter * -Properties * | FT Name, LastLogonDate -AutoSize

# Group MSK
New-ADGroup "MSK" -path 'OU=1, DC=domain,DC=local01' -GroupScope Global -
PassThru -Verbose

Add-AdGroupMember -Identity Msk -Members iivanov

# Password reset
Set-ADAccountPassword iivanov -Reset -NewPassword (ConvertTo-SecureString
-AsPlainText "P@ssw0rd1" -Force -Verbose) -PassThru

Disable-ADAccount iivanov
Enable-ADAccount iivanov

#DHCP
Install-WindowsFeature -Name 'DHCP' -IncludeManagementTools
Get-DhcpServerInDC
hostname

Add-DHCPServer4Scope -EndRange 192.168.56.254 -Name Office -StartRange
192.168.56.50 -SubnetMask 255.255.255.0 -State Active -ComputerName DC1

Set-DHCPServerv4OptionValue -ComputerName W2019SERV01 -DnsServer
192.168.56.16 -DnsDomain domain.local01 -Router 192.168.10.1

Get-DHCPServerv4OptionValue -ComputerName W2019SERV01 | Format-List
Get-DHCPServerv4OptionValue -ComputerName W2019SERV01 -ScopeId
192.168.10.0 | Format-List

Add-DhcpServer4ExclusionRange -ComputerName W2019SERV01 -ScopeId

```

```

192.168.10.0 -StartRange 192.168.10.70 -EndRange 192.168.10.75
Set-DhcpServer4Scope -ComputerName W2019SERV01 -ScopeId 192.168.10.0 -
State Active
Add-DhcpServerInDC -DnsName W2019SERV01 -IPAddress 192.168.10.15

#Export
Export-DHCPServer -ComputerName W2019SERV01 -File C:\dhcp\dhcp-export.xml

```

Задание_9:

На компьютере работающем в режиме Core поднимите резервный контроллер

```

PS C:\Users\Administrator> Get-NetAdapter

```

Name	InterfaceDescription	ifIndex	Status	MacAddress
Ethernet	Intel(R) PRO/1000 MT Desktop Adapter	7	Up	08-00-27-D7-94-20

Задание_10:

Создайте пользователя, группу, ОУ. Добавьте пользователя в группу.

The screenshot shows a dual-boot environment with two PowerShell windows and one Active Directory snap-in window.

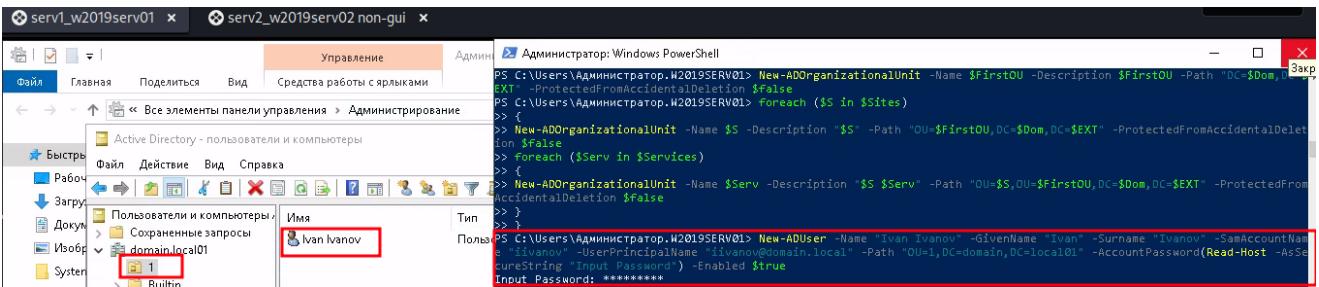
- Windows PowerShell (serv1_w2019serv01):** Shows the command `Get-Command -module activedirectory` being run, listing various cmdlets like `Add-ADCentralAccessPolicyMember`, `Set-ADReplicationSite`, etc.
- Windows PowerShell (serv2_w2019serv02 non-gui):** Shows the command `Get-Command -module activedirectory` being run, listing the same cmdlets.
- Active Directory Snap-in:** Shows the structure of the domain `domain.local01`. It includes objects for users (1, MSK, SPB), computers (Kalininograd, MSK, SPB), and domain controllers. A red circle labeled **1** points to the "Средства работы с ОУ" (OU Management Tools) button in the ribbon. A red circle labeled **2** points to the "Управление" (Management) tab in the ribbon. A red circle labeled **3** points to the "Пользователи и компьютеры" (Users and Computers) node in the navigation pane. A red circle labeled **4** points to the PowerShell window at the bottom right where the following PowerShell script is running:

```

PS C:\Users\Administrator.W2019SERV01> New-ADOrganizationalUnit -name "1" -Path "DC=domain,DC=local01"
PS C:\Users\Administrator.W2019SERV01> $fdn = Get-ADDomain
PS C:\Users\Administrator.W2019SERV01> $fulldomain = $fdn.DNSRoot
PS C:\Users\Administrator.W2019SERV01> $dom = $fulldomain.Split('.')[0]
PS C:\Users\Administrator.W2019SERV01> $ext = $dom[1]
PS C:\Users\Administrator.W2019SERV01> $sites = ("SPB", "MSK", "Kalininograd")
PS C:\Users\Administrator.W2019SERV01> $services = ("Users", "Admins", "Computers", "Servers", "Contacts")
PS C:\Users\Administrator.W2019SERV01> $firstOU = "Russia"
PS C:\Users\Administrator.W2019SERV01> New-ADOrganizationalUnit -Name $firstOU -Description $firstOU -Path "DC=$dom,DC=$ext" -ProtectedFromAccidentalDeletion $false
PS C:\Users\Administrator.W2019SERV01> Foreach ($s in $sites)
>> {
>> New-ADOrganizationalUnit -Name $s -Description "$$" -Path "OU=$firstOU,DC=$dom,DC=$ext" -ProtectedFromAccidentalDeletion $false
>> }
>> {
>> New-ADOrganizationalUnit -Name $serv -Description "$$ $serv" -Path "OU=$firstOU,DC=$dom,DC=$ext" -ProtectedFromAccidentalDeletion $false
>> }
>> }

 Чтобы активировать Windows, перейдите в раздел "Параметры".

```



Задание_11:

Установите роль DHCP сервера, создайте область, настройте её (Диапазон адресов, адреса DNS, gate). Посмотрите настройки.

```

PS C:\Users\Administrator.W2019SERV01> Install-WindowsFeature -Name 'DHCP' -IncludeManagementTools
Success Restart Needed Exit Code      Feature Result
----- ----- ----- ----- -----
True   Yes           SuccessRest... {DHCP-сервер, Средства DHCP-сервера}
ПРЕДУПРЕЖДЕНИЕ: Чтобы завершить установку, вам необходимо перезапустить этот сервер.

```

Задание_12:

Включите созданную область

Задание_13:

Экспортируйте настройки DHCP сервера

Команды:

Windows PowerShell

powershell.exe

Или в виде приложения:

_powershell_ise.exe_

Для проверки текущей политики выполнения выполним команду:
Get-ExecutionPolicy

- ****Restricted**** – Сценарии не могут быть запущены;
- ****AllSigned**** – Могут быть запущены только сценарии, подписанные доверенным издателем. Перед выполнением сценария доверенного издателя будет запрашиваться подтверждение;
- ****RemoteSigned**** – Разрешено выполнять созданные нами сценарии и скачанные сценарии, подписанные доверенным издателем;
- ****Unrestricted**** – Никаких ограничений, все скрипты могут быть запущены.

Для выполнения и тестирования понизим политику до ****RemoteSigned****

выполнив команду от имени Администратора:

```
Set-ExecutionPolicy RemoteSigned
```

Командлетами называются команды PowerShell, в которых заложена различная функциональность;

- Командлеты могут быть как системными, так и пользовательскими, созданные кем-либо;
- Командлеты именуются по правилу Глагол-Существительное, что упрощает их запоминание;
- Командлеты выводят результаты в виде объектов или их коллекций;
- Командлеты могут как получать данные для обработки, так и передавать данные по конвейеру;
- Командлеты не чувствительны к регистру (можно написать и get-process, и Get-Process, и GeT-pRoCeSs);
- После командлетов не обязательно ставить "++;", за исключением, когда выполняем несколько командлетов в одну строку (Get-Process; Get-Service).

Например, для получения текущих процессов, выполним команду:

```
Get-Process
```

```
Get-Service #для получения статуса служб, запущенных на компьютерах
```

```
Get-Content C:\Windows\System32\drivers\etc\hosts #для получения  
содержимого файла. В данном случае, файл hosts
```

Не обязательно знать наизусть все командлеты. **Get-Help** спасёт ситуацию.

```
Get-Help -Category cmdlet
```

Используя PowerShell ISE, облегчаем процесс разработки.

Достаточно ввести знак тире "----" после того, как ввели командлет, и получаем все возможные варианты параметров и их типы:

Выполним:

```
Get-Service -Name p*
```

Если забудем какие свойства есть у того или иного командлета, пропустим его через **Get-Member**:

```
Get-Process | Get-Member
```

#Знак "|" называется конвейером. О нём ниже.

Недостаточно информации? Обратимся к справке с параметром **-Examples**:

```
Get-Help Get-Process -Examples
```

- Командлеты могут иметь сокращённые названия – алиасы. Например, вместо **Get-Help** можно использовать просто **Help**. Для получения всех сокращений выполните **Get-Alias**.

Выполним:

```
Start-Process notepad
```

Что аналогично записи:

```
start notepad
```

Остановим процесс:

```
Stop-Process -Name notepad
```

Командлеты именуются по правилу Глагол-Существительное. Глагол не обязательно должен быть **Get**. Помимо того, можно задавать **Set** (**Set-ExecutionPolicy**), запускать **Start**, останавливать **Stop**, выводить **Out**, создавать **New** и многие другие. Название командлета ни чем не ограничивается, при создании собственного, можно назвать его угодно.

Выполним вывод в файл:

```
"Hello!" | Out-File C:\test.txt  
C:\test.txt
```

Аналогично можно записать так:

```
"Hello!">C:\test.txt  
C:\test.txt
```

****Комментарии****

Использовать комментарии является хорошим тоном.

Комментарии в PowerShell бывают строчные – **##** и блочные – **<#...#>**.
##:

****Конвейер****

Конвейер (**|**) – передаёт выходные данные одной команды во входные данные на обработку другой команде. Мы использовали конвейер ранее, получая все свойства объекта или, в предыдущем примере, выбирая из набора данных только поле *Caption*.

Чтобы понять принцип конвейера, выполним код:

```
Get-Service | Sort-Object -property Status
```

Что произойдёт: получаем все службы (**Get-Service**), передаём все полученные службы на сортировку в командлет **Sort-Object** и указываем, что хотим отсортировать их по параметру *Status*. На выводе мы получим сначала все службы со статусом *Stop*, а потом все службы со статусом *Running*.

В следующем примере сначала получим все запущенные службы. После первого конвейера проходимся по каждому элементу, выбираем только те службы, у которых статус *Running* и на втором конвейере выбираем, что хотим на выводе увидеть только *displayname* служб:

```
Get-Service | WHERE {$_ .status -eq "Running"} | SELECT displayname
```

В примере используем **\$_**. Данная запись означает текущий элемент в конвейере.

Переменные

– Переменная в PowerShell начинается со знака **\$** и в названии может

содержать любые буквы, цифры и символ подчёркивания.

Чтобы назначить переменной значение, достаточно его ей присвоить знаком "`**=**`". Чтобы вывести значение переменной, можно просто написать эту переменную. Вывод информации мы разберём по тексту ниже.

```
$var = 619  
$var
```

Над числами можно производить арифметические операции, строки можно складывать. Если к строке прибавить число, число автоматически преобразуется в строку.

```
$a = 1  
$b = 2  
$c = $a + $b  
$c #c = 3  
$str = "Hello"  
$str = $str + " World"  
$str #str = Hello World  
$str = $str + " 2021"  
$str #str = Hello World 2021
```

Если нам надо узнать, какой тип имеет та или иная переменная, можно воспользоваться методом `GetType()`

```
$s = "Это строка?"  
$s.GetType().FullName
```

Тип переменной PowerShell определяет автоматически или его можно назначить вручную.

```
$var = "one"  
[string]$var = "one"
```

PowerShell использует типы данных Microsoft .NET Framework. Рассмотрим основные:

Тип / .NET класс	Описание
[string] System.String	Строка \$var = "one"
[char] System.Char	Символ \$var = [char]0x263b
[bool] System.Boolean	Булево. Может иметь значение \$true или \$false . \$bvar = \$true
[int] System.Int32	32-разрядное целое число \$i = 123456789

[long] System.Int64	64-разрядное целое число \$long_var = 12345678910
[decimal] System.Decimal	128 битное десятичное число. Буква d на конце числа обязательна \$dec_var = 12345.6789d
[double] System.Double	8-байтное десятичное число с плавающей точкой [double]\$double_var = 12345.6789
[single] System.Single	32 битное число с плавающей точкой [single]\$var = 123456789.101112
[DateTime] System.DateTime	Переменная даты и времени. \$dt_var = Get-Date
[array] System.Object[]	<p>Массив. Индекс элементов массива начинается с 0 — чтобы обратиться к первому элементу массива \$mas, следует написать **\$mas[0]**.</p> <pre>\$mas = "one", "two", "three";</pre> <p>Для добавления элемента к массиву, можно записать</p> <pre>\$mas = \$mas + "four"</pre> <p>Каждый элемент массива может иметь свой тип</p> <pre>\$mas[4] = 1</pre>
[hashtable] System.Collections.Hashtable	<p>Хеш-таблицы. Различие между хеш-таблицами и массивами в том, что в массивах используются индексы, а в хеш-таблице именованные ключи. Хеш-таблицы строятся по принципу: **@{ ключ = «значение» }**</p> <pre>
\$ht = @{odin="one"; dva="two"; tri="three"}

Чтобы добавить элемент к хеш-таблице, можно или присвоить ей тот ключ, которого ещё нет, или воспользоваться методом **Add()**. Если присваивание делать к существующему ключу — значение ключа изменится на присваиваемое. Для удаления элемента из хеш-таблицы есть метод **Remove()**.</pre> <pre>
\$ht.Add("chetyre", "four")
</pre>

	<pre>
\$ht.pyat = "five"

\$ht.Remove("dva")</pre>	
--	--	--

В переменную можно записать любое значение, так же можно записать вывод любого командлета.

Задача: требуется узнать **IP-адрес** и **MAC-адрес** нашего компьютеров в сети. Имя компьютера определим командой HOSTNAME . Вариант решения :

```
$MAC = Get-WmiObject -ComputerName **<domain-comp1, domain-comp2>** -Class Win32_NetworkAdapterConfiguration -Filter IPEnabled=True | Select-Object -Property * | SELECT PSComputerName, @{Name="IPAddress";Expression={$_.IPAddress.get(0)}}, MACAddress, Description
$MAC
```

Что происходит:

Мы записываем в переменную **\$MAC** результат от выполнение командлета **Get-WmiObject**, с помощью которого передаём WMI запрос на компьютеры, указанные в параметре **-computername** и возвращаем нужную нам информацию из класса **Win32_NetworkAdapterConfiguration**.

Обратите внимание, если надо получить IP-адрес и MAC-адрес текущего компьютера, то вместо имён компьютеров, параметру **-computername** мы передадим точку.

Область действия переменных

Область действия переменной в PowerShell может быть либо **локальной**, либо **глобальной**. По умолчанию переменная имеет локальную область действия, например, доступна только в функции или только в текущем сценарии. Глобальная переменная действует во всём текущем сеансе PowerShell. Для того, чтобы обозначить глобальную переменную, достаточно написать следующую конструкцию:

```
**$Global: переменная = значение**
$Global:var = 12
```

Операторы сравнения и логические операторы

Операторы сравнения и логические операторы проверяют равенство или соответствие между двумя значениями.

- При выполнении условия, конструкция сравнения всегда возвращает логическое значение ****\$true**** или ****\$false****, если условие ложь.

В таблице ниже приведены операторы сравнения:

Оператор	Описание	Пример
-eq	Equal / Равно (=)	<code>var = "619"

var -eq 123 #\$false</code>

-ne	Not equal / Не равно (<>)	<code>var =//619//

var -ne 123 #\$true</code>
-gt	Greater than / Больше (>)	<code>var =//619//

var -gt 123 #\$true</code>
-ge	Greater than or equal / Больше или равно (>=)	<code>var =//619//

var -ge 123 #\$true</code>
-lt	Less than / Меньше (<)	<code>var =//619//

var -lt 123 #\$false</code>
-le	Less than or equal / Меньше или равно (<=)	<code>var =//619//

var -le 123 #\$false</code>
-like	Сравнение с учётом символа подстановки	<code>"Abc" -like "abc*" #true</code>
-notlike	Сравнение с учётом не соответствия символа подстановки	<code>"Abc" -notlike "abc*" _#false_</code>
-contains	Содержит ли значение слева значение справа	1, 2, 3, 4, 5 -contains 3 #\$true
-notcontains	Если значение слева не содержит значение справа, получим истину	1, 2, 3, 4, 5 -notcontains 3 #\$false
-match	Использование регулярных выражений для поиска соответствия образцу	<code>\$str = "http://mail.ru"
\$str - match "^http://(\S+)(\.\w+)\$" _#\$true_</code>
-notmatch	Использование регулярных выражений для поиска несоответствия образцу	<code>\$str = "http://mail.ru"

\$str -notmatch "^http://(\S+)+ (\.\w+)\$" _#true_</code>
-replace	Заменяет часть или все значение слева от оператора	<code>"Abcabc" -replace "Abc","abc"</code>

Можно дополнительно задать условие, если при сравнении следует учитывать регистр.
Для этого следует перед оператором подставить букву "**c**"

```
$str1 = "Abc"
$str2 = "abc"
$str1 -ceq $str2 #$false
```

- Можно использовать несколько операторов сравнения, применяя логические операторы. Логические операторы перечислены в таблице ниже:

Оператор	Описание	Пример
-and	Логическое и	("Строка" -eq "Строка") -and (619 -eq 619) #\$true
-or	Логическое или	("Строка" -eq "Строка") -or (619 -eq 123) #\$true

-not	Логическое не	-not (123 -gt 324) #\$trueили!(123 -gt 324) #\$true
------	----------------------	---

Пара слов про **if... elseif** и **else**

if... elseif и **else**

```
if( условие1-верно ) { выполняем-код } #если условие 1 верно, выполняем код, если нет - идём дальше
elseif( условие2-верно ) { выполняем-код } #необязательное условие: иначе, если условие 2 верно, выполняем
код
else { выполняем-код } #необязательное условие: если прошлое условие не верно, выполняем
```

Если условие верно, выполняется следующий за условием код в фигурных скобках.

Если условие не верно, то пропускаем выполнение и смотрим, продолжается ли условие дальше.

Если находим elseif — проверяем новое условие и так же, в случае успеха, выполняем код, иначе, снова смотрим, есть ли продолжение в виде elseif или else.

Если условие выполняется, то код выполняется и ветка elseif или else пропускается.

```
$str = ""
if ( 1 -gt 2 ) { $str = "Апельсин" }
elseif ( $str -eq "Апельсин" ) { $str }
else { $str = "Яблоко"; $str }
```

Циклы

Циклы в PowerShell мало чем отличаются от циклов в других языках. Почти всё как и везде:

Do While

Делать до тех пор, пока условие верно

Do Until

Цикл, который повторяет набор команд, пока выполняется условие

For

Цикл, который повторяет одинаковые шаги определённое количество раз

ForEach (Цикл **foreach** позволяет перебирать значения, которые находятся в массиве без условий.

Единственное условие: выполнить действие для каждого элемента массива)

Функции

Функции в PowerShell имеют следующую структуру:

```
function имя-функции ([параметр1, ...параметрN])
{
    тело-функции
}
```

Функции не обязательно должны возвращать результат.

Командлеты модуля AD для PowerShell

В модуле Active Directory для Windows PowerShell имеется большое количество командлетов для взаимодействия с AD.

Командлеты класса **Get**- используются для получения различной информации из AD (Get-ADUser — свойства пользователей, Get-ADComputer — параметры компьютеров, Get-ADGroupMember — состав групп и т.д.).

Командлеты класса **Set**- служат для изменения параметров объектов в AD, например, вы можете изменить свойства пользователя (Set-ADUser), компьютера (Set-ADComputer), добавить пользователя в группу и т.д.

Команды, начинающиеся с **New**- позволяют создать объекты AD (создать пользователя — New-ADUser, группу — New-ADGroup).

Командлеты **Remove**- служат для удаления объектов AD.

Дополнительно:

Глоссарий

Дополнительные материалы

Используемые источники

Выполнил: AndreiM