Чернівецький національний університет імені Юрія Федьковича

Інститут фізико-технічних та комп’ютерних наук

Відділ комп’ютерних технологій

Кафедра математичних проблем управління і кібернетики

**Звіт**

про виконання лабораторної роботи №4

*«Керування процесами та службами, отримання та аналіз системної інформації»*

з дисципліни

**“Адміністрування операційних систем”**

Виконав: студент 441 групи

Бужак Андрій

Перевірив: …………асист. Коцур М.П.

Оцінка:

Дата захисту:

Чернівці

2019

*Завдання*: Ознайомитись із засобами PowerShell для керування процесами та службами. Отримати навички застосування засобів PowerShell для отримання системної інформації. Розробити сценарій для виконання перерахованих дій, описати використані засоби, провести тестування та оформити звіт:

• вивести список процесів;

• запустити (за виборм викладача) процес і засобами PowerShell зупинити його;

• вивести список запущених служб;

• запустити службу (за виборм викладача);

• вивести список команд що виконуються при завантаженні системи;

• вивести основні властивості ОС (назва, версія, розрядність);

• вивести список встановлених програмних продуктів;

• вивести інформацію про розмір фізичної пам'яті;

• вивести інформацію про процесор(-и);

• вивести інформацію про мережевий адаптер;

• вивести IP-адресу комп’ютера.

Інформацію слід виводити у структурованому вигляді.

Кількість балів – 8.

Додаткове завдання:

• розробити сценарій для вимкнення комп’ютера в зазначений час.

Кількість балів – 2.

Роздрукований звіт та працездатність сценарію перевіряє та оцінює викладач.

Максимальна кількість балів за роботу – 10.

Термін здачі роботи – до 22.11.19.

**Хід виконання основного завдання**

Мною було розроблено сценарій для розв’язування основної задачі.

'Process list:';

Get-Process;

'';

'============================================================';

'';

$pathToStartProcess = Read-Host -Prompt 'Input filePath to start process';

$startedProcess = Start-Process -FilePath $pathToStartProcess -PassThru;

Start-Sleep -Seconds 2;

Stop-Process -Id $startedProcess.Id;

'';

'============================================================';

'';

'Running services:';

Get-Service | Where-Object -Property Status -eq Running | Format-Table Name, DisplayName;

'';

'============================================================;

'';

$serviceNameToStart = Read-Host -Prompt 'Input name of service to start';

Start-Service -Name $serviceNameToStart;

'';

'============================================================';

'';

'Commands that runs on system loading:';

Get-WmiObject Win32\_StartupCommand | Format-Table Caption, Command, Location;

'';

'============================================================';

'';

'OS info.';

$osInfo = Get-WmiObject Win32\_OperatingSystem;

'Name: ' + $osInfo.Name.Split('|')[0];

'Version: ' + $osInfo.Version;

'Architecture: ' + $osInfo.OSArchitecture;

'';

'============================================================';

'';

'Installed applications:';

Get-ItemProperty HKLM:\Software\Wow6432Node\Microsoft\Windows\CurrentVersion\Uninstall\\* `

| Select-Object DisplayName, DisplayVersion, Publisher, InstallDate `

| Sort-Object DisplayName `

| Where-Object DisplayName -ne $null `

| Format-Table;

'';

'============================================================';

'';

'Physical memory size: ' + (Get-CimInstance Win32\_PhysicalMemory | Measure-Object -Property Capacity -Sum).Sum / 1gb + ' GB';

'';

'============================================================';

'';

'Processors info:';

$property = 'Name', 'MaxClockSpeed', 'AddressWidth', 'NumberOfCores', 'NumberOfLogicalProcessors';

Get-WmiObject Win32\_Processor -Property $property | Format-Table -Property $property;

'';

'============================================================';

'';

'Network adapters:';

Get-NetAdapter;

'';

'============================================================';

'';

'IP-address: ' + (Test-Connection -ComputerName (hostname) -Count 1).IPV4Address;

'';

'============================================================';

pause;

**Пояснення до сценарію основної задачі**

1. Вказуючи властивість PassThru нам повернеться об’єкт, що містить дані про запущений процес. Запам’ятавши його в змінну можемо пізніше викликати його властивість Id що завершити процес по Id попередньо зачекавши дві секунди після старту.

$startedProcess = Start-Process -FilePath $pathToStartProcess -PassThru;

Start-Sleep -Seconds 2;

Stop-Process -Id $startedProcess.Id;

1. Wmi об’єкт Win32\_OperatingSystem у властивості Name містить окрім ім’я надлишкові дані, розділені символом ‘|’, то зробивши Split беремо перший елемент.

$osInfo = Get-WmiObject Win32\_OperatingSystem;

'Name: ' + $osInfo.Name.Split('|')[0];

1. Символ ‘`’ необхідний для розбиття виразу на декілька рядків, щоб полегшити читання коду.

'Installed applications:';

Get-ItemProperty HKLM:\Software\Wow6432Node\Microsoft\Windows\CurrentVersion\Uninstall\\* `

| Select-Object DisplayName, DisplayVersion, Publisher, InstallDate `

| Sort-Object DisplayName `

| Where-Object DisplayName -ne $null `

| Format-Table;

1. Так як планок пам’яті в системі може бути декілька, то повертається масив. Просумувавши ємність всіх елементів, маємо загальний розмір.

'Physical memory size: ' + (Get-CimInstance Win32\_PhysicalMemory | Measure-Object -Property Capacity -Sum).Sum / 1gb + ' GB';

**Хід виконання додаткового завдання**

Мною було розроблено сценарій для розв’язування додаткової задачі.

$answer = Read-Host -Prompt "Do you want enable scheduled shutdown? ['y' = enable; otherwise = disable]";

'';

if (!($answer.ToLower()[0] -eq 'y'))

{

shutdown -a;

'Scheduled shutdown disabled.';

'';

pause;

exit;

}

function IsTimeValid ([string]$time)

{

if ([regex]::IsMatch($time, '\d\d:\d\d:\d\d'))

{

$timeItems = $time.Split(':');

if ([int]$timeItems[0] -le 23 -and [int]$timeItems[1] -le 59 -and [int]$timeItems[2] -le 59)

{

$true;

}

else

{

$false;

}

}

else

{

$false;

}

}

do

{

$targetTime = Read-Host -Prompt "Input target shutdown time in next format - 'hh:mm:ss', for example - 18:30:00 ";

}

while (!(IsTimeValid $targetTime));

$currentDateTime = Get-Date;

$targetTimeItems = $targetTime.Split(':');

$targetDateTime = Get-Date -Date $currentDateTime -Hour $targetTimeItems[0] -Minute $targetTimeItems[1] -Second $targetTimeItems[2];

$timespan = (New-TimeSpan -Start $currentDateTime -End $targetDateTime).TotalSeconds;

if ($timespan -lt 0)

{

$timespan = 86400 + $timespan;

}

shutdown -f -s -t $timespan;

'';

'Scheduled shutdown enabled.';

'System will shutdown after ' + $timespan + ' seconds.';

'';

pause;

**Пояснення до сценарію додаткової задачі**

1. Користувачу надається вибір або задати заплановане вимкнення або скасувати його.

$answer = Read-Host -Prompt "Do you want enable scheduled shutdown? ['y' = enable; otherwise = disable]";

'';

if (!($answer.ToLower()[0] -eq 'y'))

{

shutdown -a;

'Scheduled shutdown disabled.';

'';

pause;

exit;

}

1. Функція перевіряє регулярним виразом чи відповідає введений рядок формату часу і чи числові значення компонентів часу в допустимих межах.

function IsTimeValid ([string]$time)

{

if ([regex]::IsMatch($time, '\d\d:\d\d:\d\d'))

{

$timeItems = $time.Split(':');

if ([int]$timeItems[0] -le 23 -and [int]$timeItems[1] -le 59 -and [int]$timeItems[2] -le 59)

{

$true;

}

else

{

$false;

}

}

else

{

$false;

}

}

1. Читання необхідного часу для вимкнення системи допоки він не буде введений у припустимому форматі.

do

{

$targetTime = Read-Host -Prompt "Input target shutdown time in next format - 'hh:mm:ss', for example - 18:30:00 ";

}

while (!(IsTimeValid $targetTime));

1. Запам’ятовуємо поточний час, та будуємо цільовий час поки що припускаючи, що він буде в поточній добі.

$currentDateTime = Get-Date;

$targetTimeItems = $targetTime.Split(':');

$targetDateTime = Get-Date -Date $currentDateTime -Hour $targetTimeItems[0] -Minute $targetTimeItems[1] -Second $targetTimeItems[2];

1. Якщо цільовий час виявиться в наступній добі, то результат (New-TimeSpan).TotalSeconds поверне від’ємне значення. Кількість секунд у добі 86400, додаємо до цього від’ємне обчислене значення $timespan і отримаємо необхідниу кількість секунд до вимкнення наступної доби.

$timespan = (New-TimeSpan -Start $currentDateTime -End $targetDateTime).TotalSeconds;

if ($timespan -lt 0)

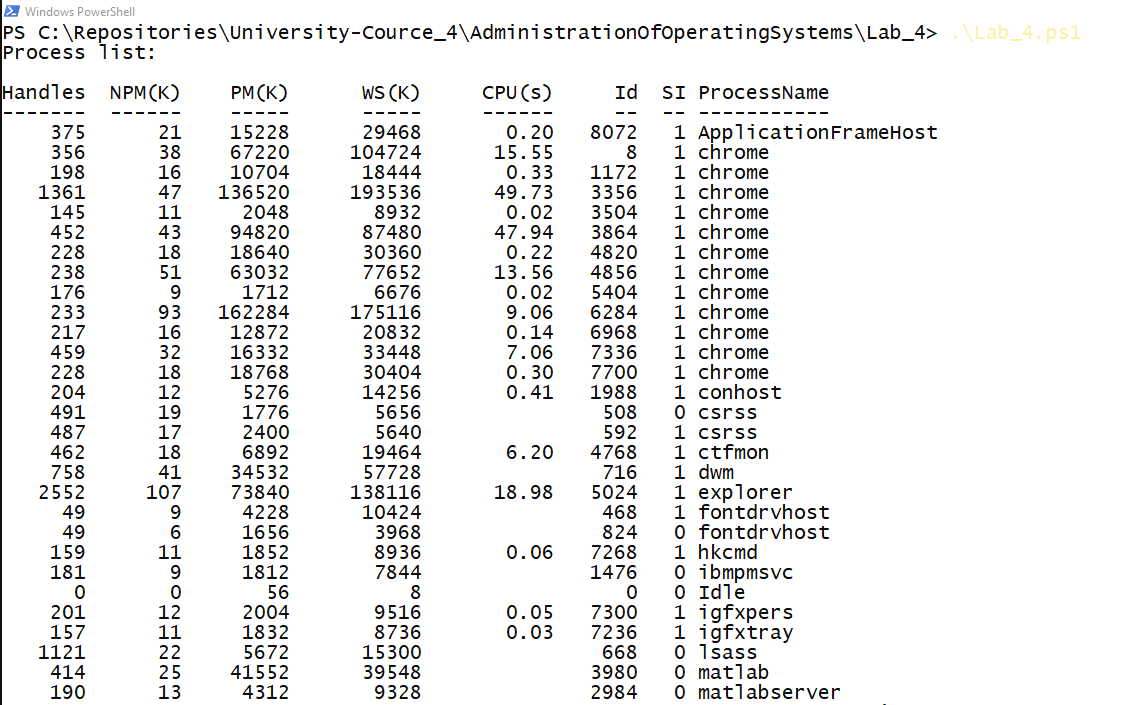
{

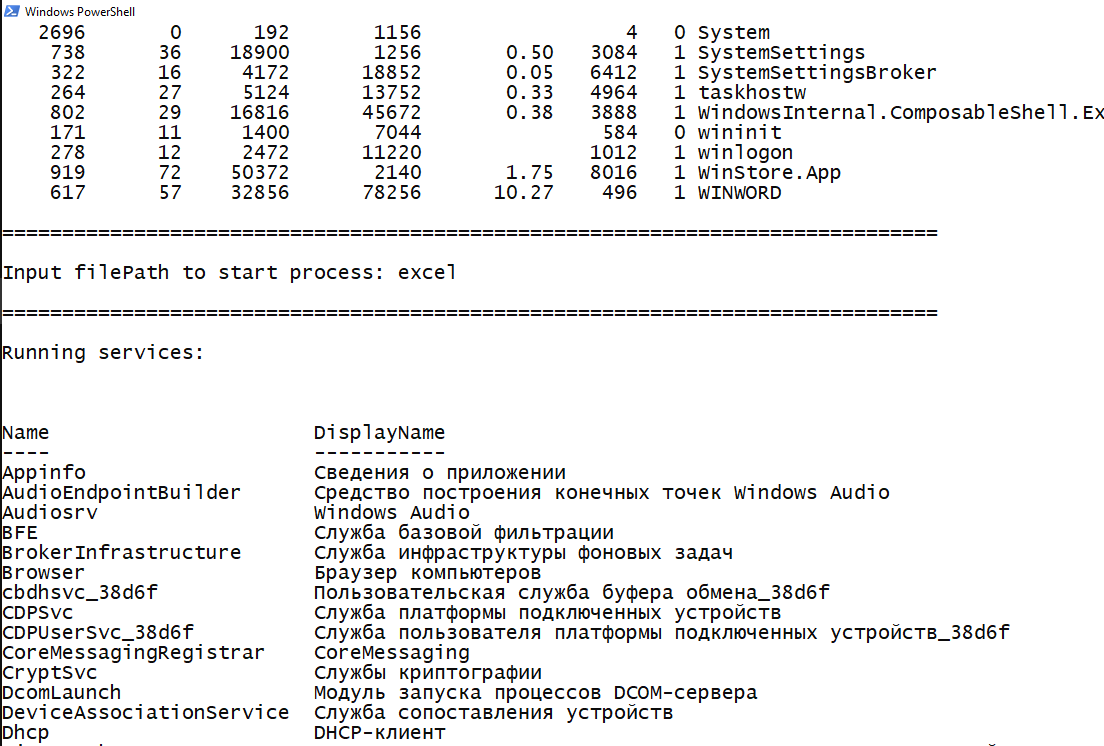
$timespan = 86400 + $timespan;

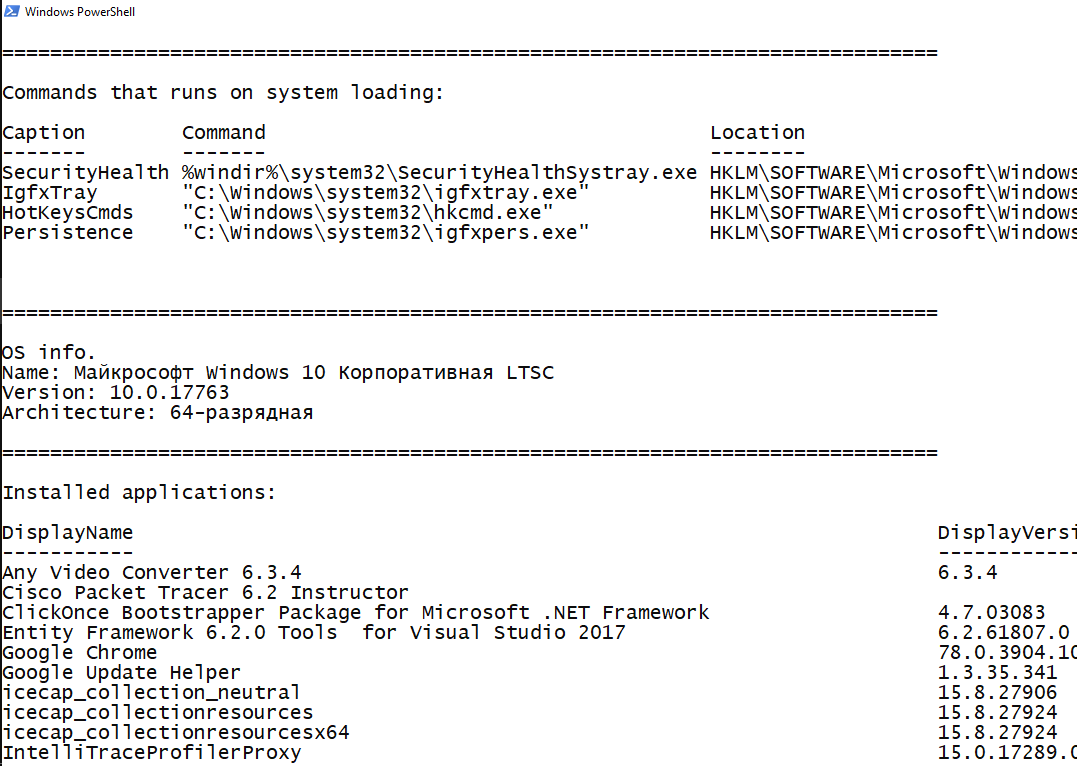
}

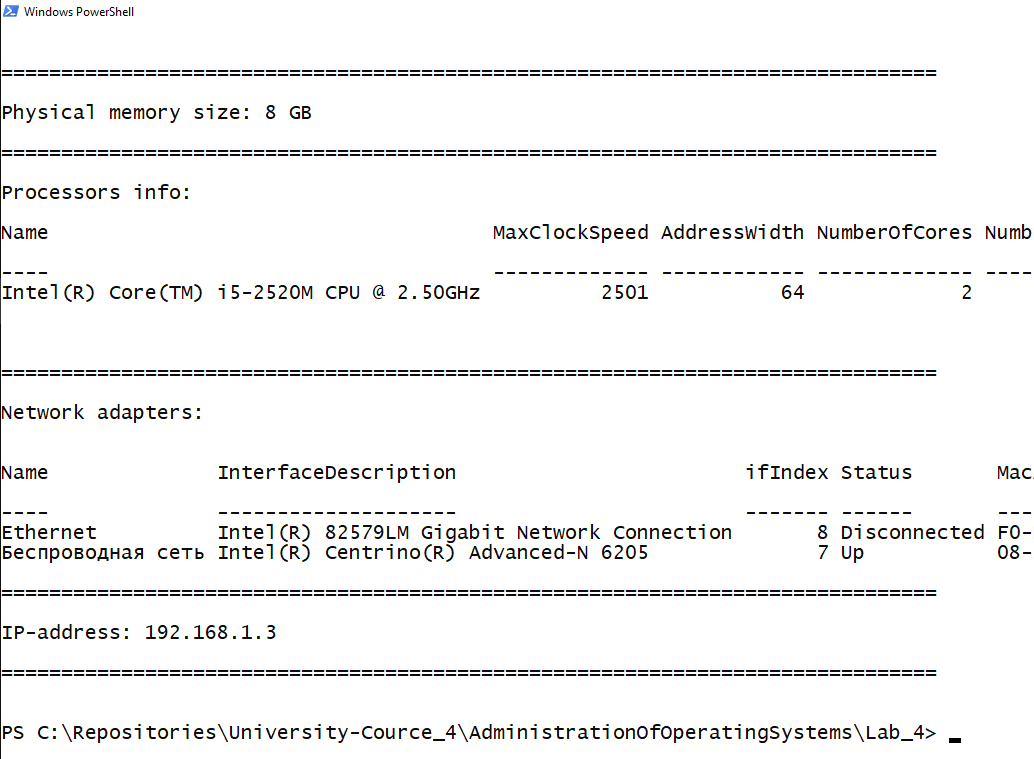
***Результат виконання***

[Основне завдання]

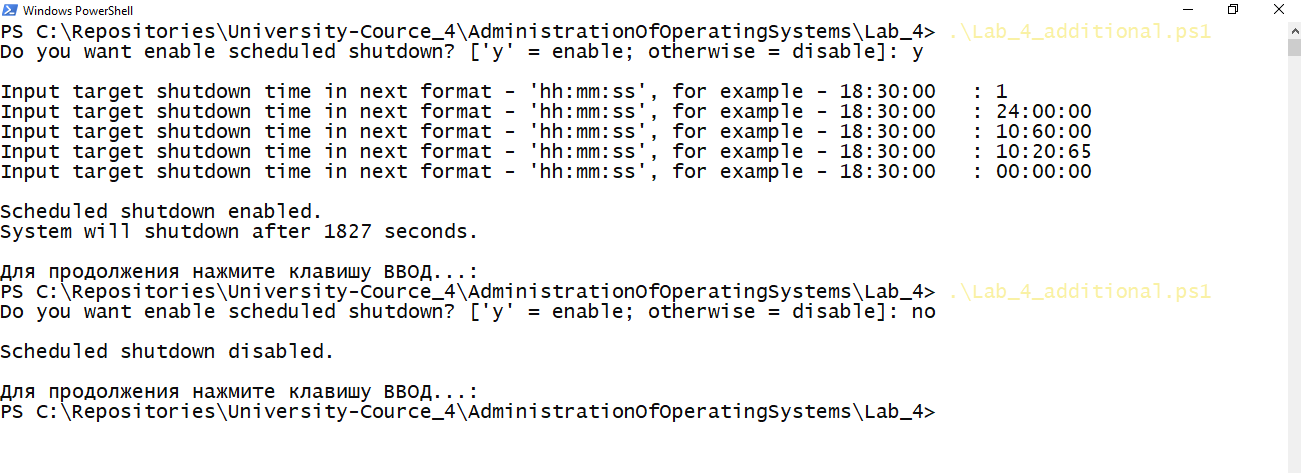








[Додаткове завдання]



***Висновки***

Отже, аналіз результатів виконання роботи дозволяє стверджувати, що завдання виконане в повному обсязі – розроблений код працює згідно поставлених вимог.