

CSIS 2260 : Winter 2021 - Due March 18 @ 4 PM
Lab 8 : Windows Server 2016 – Windows PowerShell

Name: Catherine Methven **Student No.:** 300361000

Objectives: To learn basics about using Windows **PowerShell** in Windows Server 2016.

A. Introduction to Windows PowerShell

[____]/6]

The **Windows PowerShell** is a powerful command line tool that is available in current versions of Microsoft Windows. It uses a special set of commands (called cmdlets) that are not available in the Windows command prompt (cmd). It can be used for a wide range of administrative tasks. In this assignment #5, we will look at some basic features of the PowerShell.

- 1) Starting Windows Power Shell
 - a) Click on *Start* and start typing “PowerShell”. Note that two different options show up – PowerShell and PowerShell ISE (which stands for integrated scripting environment). We will be using PowerShell ISE later – for now, just start the “regular” PowerShell.
 - b) What does the command prompt look like?
Like Windows Command Line
- 2) Updating and Using the Built-in Help Function
 - a) To install and update the local help files for PowerShell commands, type the command **Update-Help** and press Enter. Note that the PowerShell is general not case sensitive, i.e. it doesn’t matter whether you use uppercase or lowercase.
 - b) PowerShell commands (cmdlets) typically have names that consist of a verb, a hyphen, and a noun. For example, the cmdlet to list the content of the current location (e.g. directory or folder) is Get-ChildItem.
 - c) Type the command **Get-ChildItem** and press Enter. What is the output?
The folders with their last write time and file permissions are displayed.
 - d) For many commands, a shorter, alternative name called an alias is defined. To see a list of all available aliases, enter the command **Get-Alias** and press Enter. From the list shown, write down two aliases that sound familiar to you (e.g. from the commands we learned in the Linux labs, or from prior experience with the Windows Command Prompt). Also write down what the aliases stand for.
ps – get-process
ls – get-childitem

- e) To get help for a command or alias, use the command **Help**, followed by the command or alias you want help for. For example, type the command **Help ghy** and press Enter. What command does this alias stand for, and what does it do?

Get-history = displays commands that have been used in the past with their id numbers

- f) Try out the command **ghy**. What is its output?

**1 update-help
2 get-childitem
3 get-childitem
4 get-alias
5 help ghy**

B. Basic Command-Line Techniques

[____/8]

The PowerShell behaves very similarly to the Linux command shell. In particular, you can use the up and down keys to cycle through previous commands, and you can use the tab key to have the PowerShell autocomplete commands, file names etc.

- 1) Use of Wildcards
- a) Just like in the Linux command prompt, we can use wildcards in PowerShell. For example, the asterix wildcard (*) stands for any sequence of characters. This is for instance useful in connection with the command **Get-Command**, which lets you search for commands based on keywords in case you don't remember or know the exact name of a command.
- b) Type the command **Get-Command *path*** and press Enter. Write down three of the commands that are listed:
- Add-PartitionAccessPath
Get-RDVirtualDesktopTemplateExportPath
Remove-PartitionAccessPath**
- 2) Input and Output Redirection and Basic File and Directory Commands
- a) Just like in Linux, the input for and output of commands can be redirected to and from files. The greater sign (>) redirects output, the less than sign (<) redirects input, and the pipe symbol (|) can be used to send the output from one command to the next command.
- b) Type the command alias **ls** and press Enter. Just like in a linux shell, this command will list the content of the current directory. Now type **ls > files.txt** and press Enter. Use **ls** to verify that a new file files.txt is now in the current directly.
- c) Using the commands you remember from Linux, or using the help functions described above, perform and write down the commands to do the following: i) Create a subdirectory named backups:
- Mkdir backups**

c.ii) Create a copy of the file files.txt called files.bak:

Cp files.txt files.bak

c.iii) Move the file files.bak into the subdirectory backups:

Mv files.bak backups

c.iv) Delete the file files.txt in the current directory:

Rm files.txt

d) To show a list of all currently running processes, type the command **Get-Process** and hit Enter. Note that the result is a long list of processes. One of the columns displayed has the heading WS(K). This stands for working set (in Kilobytes), i.e. the amount of memory currently allocated to each process.

e) Assume we only want to see processes that currently use more than 50 Mbytes. For this purpose, type **Get-Process | Where-Object WorkingSet -gt 50mb** and hit Enter. Remember how the pipe mechanism works – the output of the Get-Process command is not displayed directly, but handed to the Where-Object command, which filters by the size of the working set and then displays the result.

f) Using another pipe, add the command **sort-object -Descending ws** to the previous command. Describe what this addition does: **Orders process of more than 50mb in order of largest to smallest WS(K)**

3) Variables

a) We can also use variables to store the results of commands temporarily, to be re-used in later commands. For example, the command **Get-Date** displays the current date. (Try it out!). To

store the result of the command (i.e., the current date) in a variable called \$today, type the following command and hit Enter: **\$today = Get-Date**

b) To display the content of the variable, simply type its name followed by enter (i.e. **\$today**).

c) To access and display different parts of the date, use the commands **\$today.Year**, **\$today.Month**, **\$today.Day**, **\$today.Hour**, **\$today.Minute** and **\$today.Second**.

d) What is the output of **\$today.DayOfYear**? **317**

One of the most powerful applications of command line interfaces like the PowerShell is that you can use command scripts, i.e. files containing a number of commands, to automate (possibly repetitious) tasks.

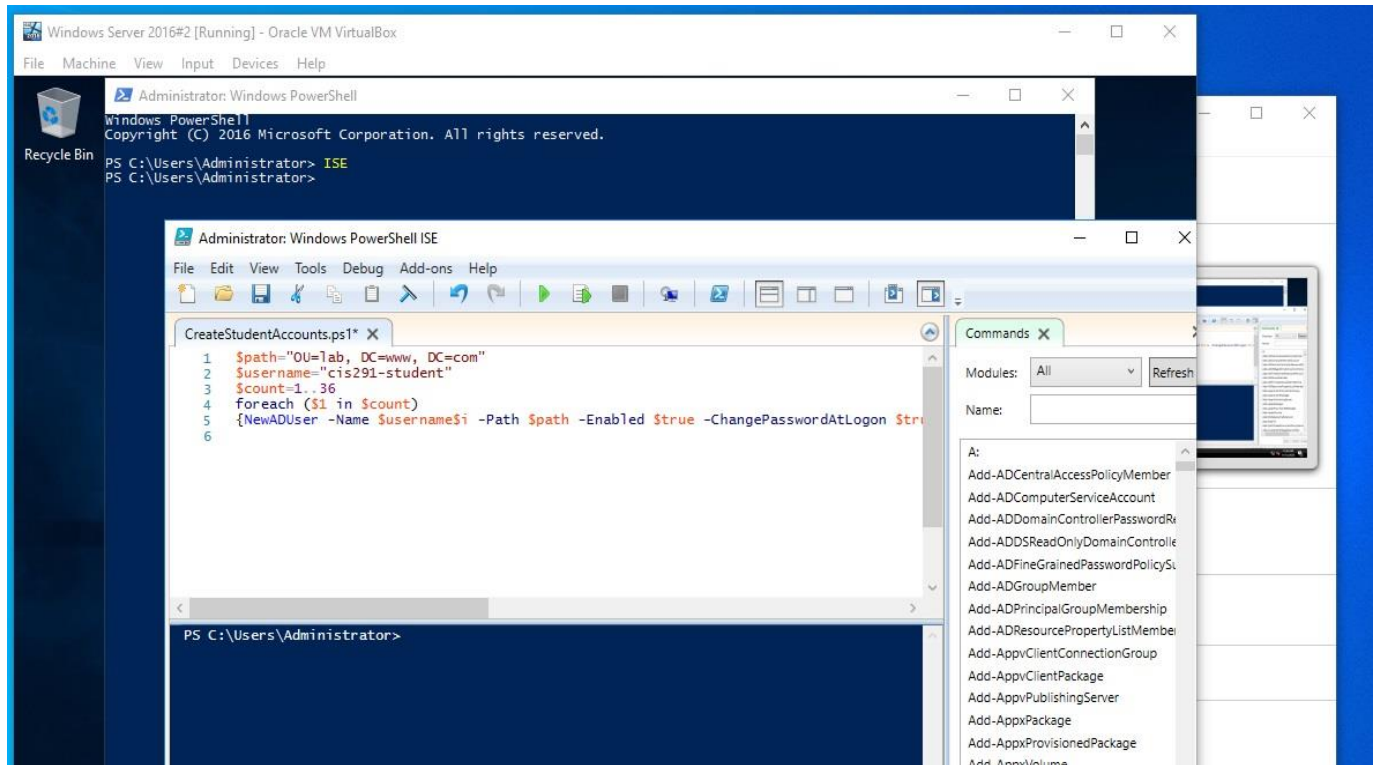
1) Enabling PowerShell Scripts

- a) Since PowerShell scripts are a very powerful mechanism to execute a number of (possibly harmful) commands very quickly, the execution of PowerShell scripts is turned off by default.
- b) Type the command **Get-ExecutionPolicy** and hit Enter. What is the output?
RemoteSigned
- c) Type the command **Help About_Execution_Policies** to get information about the different possible execution policies. What are the five Execution Policies? **Unable to see five execution policies due to an error with the help command.**
- d) To allow the execution of locally created (or downloaded and signed) scripts, enter the command **Set-ExecutionPolicy RemoteSigned** and hit Enter. When prompted, type
Y to confirm and press Enter.

2) Using a Script to Create User Accounts

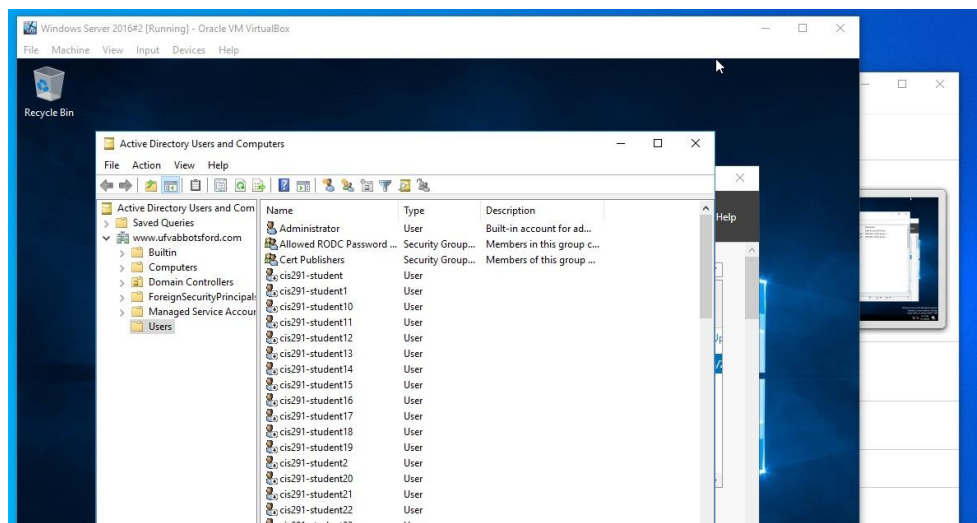
- a) PowerShell scripts are simply text files that contain PowerShell commands. While they can be created with any text editor, using the integrated scripting environment of PowerShell has advantages such as syntax highlighting, debugging support etc. To open the integrated scripting environment from the already open PowerShell Window, type **ISE** and press Enter.
- b) In the editor that opens, type the following script. Note that the last three lines are just a single line (broken here for space reasons):

```
$path="OU=lab,DC=csis,DC=com" where csis is your domain name
from Assign#1-www
$username="CIS291-student"
$count=1..36
foreach ($i in $count)
{ New-ADUser -Name $username$i -Path $path -Enabled $True
-ChangePasswordAtLogon $true -AccountPassword (ConvertTo-
SecureString "P@ssw0rd" -AsPlainText -force) -passThru }
```
- c) Save the script under the name **CreateStudentAccounts** and run it by clicking at the green arrow or by pressing F5.
- d) If you get any error messages, go back and check that you typed the script correctly.



e) Once the script works, what is its output? **It creates 36 users in the Organizational Unit under the Domain Controller**

f) In the Server Manager, under Tools, select Active Directory Users and Computers.

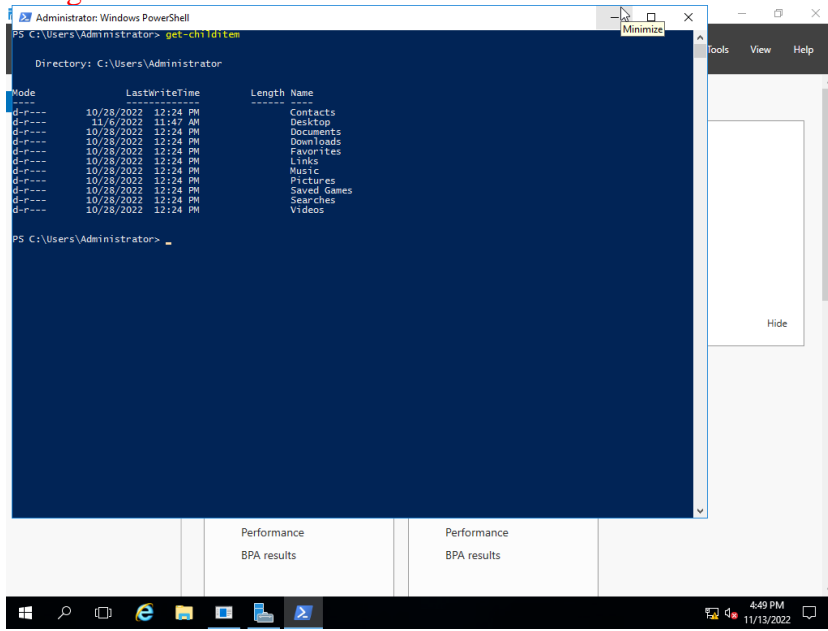


g) Navigate to OU lab in the domain csis.com. Can you see the 36 newly created users?

Yes

- h) **Take a Screen Print of the successful creation of the 36 users and paste here – submit as Firstname student# Lab8 in Blackboard**

Using cmdlets



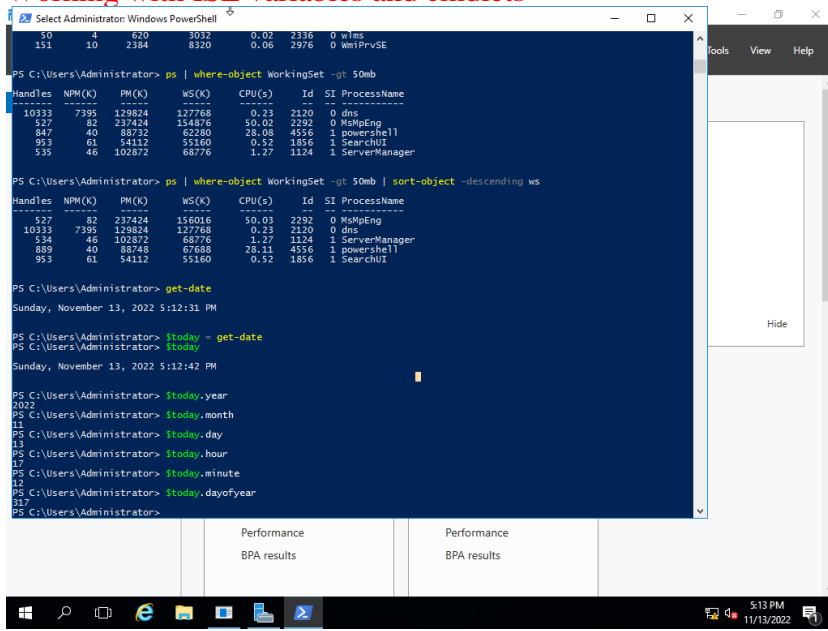
```
PS C:\Users\Administrator> get-childitem
```

Directory: C:\Users\Administrator

Mode	LastWriteTime	Length	Name
d-r----	10/28/2022 12:24 PM		Contacts
d-r----	11/8/2022 11:47 AM		Desktop
d-r----	10/28/2022 12:24 PM		Documents
d-r----	10/28/2022 12:24 PM		Downloads
d-r----	10/28/2022 12:24 PM		Favorites
d-r----	10/28/2022 12:24 PM		Links
d-r----	10/28/2022 12:24 PM		Music
d-r----	10/28/2022 12:24 PM		Pictures
d-r----	10/28/2022 12:24 PM		Saved Games
d-r----	10/28/2022 12:24 PM		Searches
d-r----	10/28/2022 12:24 PM		Videos

PS C:\Users\Administrator>

Working with ISE variables and cmdlets



```
PS C:\Users\Administrator> ps | where-object WorkingSet -gt 50mb
```

Handles	NPM(K)	PM(K)	WS(K)	CPU(s)	Id	SI	ProcessName
10333	7395	129824	127768	0.23	2120	0	dns
527	82	237424	154876	50.02	2292	0	Winlogon
847	40	88732	62280	28.08	4556	1	powershell
953	61	54112	55160	0.52	1856	1	SearchUI
535	46	102872	68776	1.27	1124	1	ServerManager

```
PS C:\Users\Administrator> ps | where-object WorkingSet -gt 50mb | sort-object -descending ws
```

Handles	NPM(K)	PM(K)	WS(K)	CPU(s)	Id	SI	ProcessName
527	82	237424	156016	50.03	2292	0	Winlogon
10333	7395	129824	127768	0.23	2120	0	dns
534	46	102872	68776	1.27	1124	1	ServerManager
809	40	88748	67688	28.11	4556	1	powershell
953	61	54112	55160	0.52	1856	1	SearchUI

```
PS C:\Users\Administrator> get-date
```

Sunday, November 13, 2022 5:12:31 PM

```
PS C:\Users\Administrator> $today = get-date
```

```
PS C:\Users\Administrator> $today
```

Sunday, November 13, 2022 5:12:42 PM

```
PS C:\Users\Administrator> $today.year
```

2022

```
PS C:\Users\Administrator> $today.month
```

11

```
PS C:\Users\Administrator> $today.day
```

13

```
PS C:\Users\Administrator> $today.hour
```

12

```
PS C:\Users\Administrator> $today.minute
```

12

```
PS C:\Users\Administrator> $today.dayofyear
```

317

```
PS C:\Users\Administrator>
```

Creating 36 users with PowerShell ISE scripting

