Powershell

Introduction
COMP2101 Fall 2018

Powershell Versions

- Windows Powershell version 5.1 is the target version for this course
- The get-host command can be used to see your Windows Powershell version (e.g. (get-host).version.tostring())
- If you do not have version 5.1 of Windows Powershell, upgrade your system

- Windows Powershell 5.1 is the current release of the Windows-specific version of Powershell
- Powershell Core 6.0 for Linux, MacOSX, and Windows is the newest release from Microsoft and has serious changes for Windows Powershell users
- 6.0 is so different they came up with a new command to run it (pwsh) and renamed the old Powershell to Windows Powershell - we will just use the name Powershell to save slide real estate

Powershell Github Setup (optional)

- You can create a folder in your github COMP2101 repository to hold Powershell scripts, or create a new repository if you prefer

- Clone that repository to your PC - use your student network drive or a folder on a USB stick if you are using a lab PC

- Create your scripts during the semester in the Powershell folder that was part of your github clone and keep it synchronized with github using git add, commit, push or the windows git tools from github

Powershell Privileged User

- Your Windows login provides a privilege level
- Windows administrator login does not grant Powershell administrator privilege
- Use Run As to get administrator privilege level in Powershell, regardless of what login you used for Windows

Privilege Exercise

- Start Powershell in console mode
- Run the command

```
get-acl c:/windows/*
```

- Note the error
- Run Powershell using Run As to gain administrator privilege and rerun the command
- Note the difference in the window frame title

Console vs. ISE

- Console mode is available even without the gui, and is also useful when you have a low resolution display
- ISE (Integrated Scripting Environment) is a development environment and provides convenient access to supplemental tools
- Privilege restrictions apply to both
- They have separate profiles, most commands work in both
- Only console mode has a future and is cross-platform as of 6.0
- ISE is deprecated now, and only works with Windows Powershell 5.1 and below

Interface Exercises

- Start Powershell in console mode and in ISE mode
- Run the command **ise** from the Powershell console
- Try entering these commands in both modes

get-process

get-host

gwmi -class win32_processor

get-history

Cmdlets

- Cmdlets are what the light-weight commands in Powershell are called, similar in concept to bash built-in commands
- Thousands are built into Powershell and you can create your own by writing functions
- Cmdlets and their parameters are case insensitive

Cmdlet Name Composition

- Powershell cmdlets are tied to the .NET libraries and are therefore dependent on the .NET version you have installed, as well as what operating system you have
- The general form for cmdlets is verb-noun with a list of well-known verbs which you can display using get-verb
- Nouns are defined by the installed modules from .NET along with any modules you load

Cmdlet Naming

- Use well-known verbs when creating your own cmdlets whenever possible
- Nouns from loaded modules for use with a verb can be listed using get-help *verb*-
- Using get-help to show available cmdlets and topics is limited to the ones which belong to modules already loaded

Getting Help

- Powershell provides online help with the get-help command
- help is a function invoking get-help that automatically paginates the output by piping get-help to the more command
- Running help or get-help without any parameters displays how to use the get-help command

Help Types

- You can run get-help on cmdlets or on topics
- Topic help pages are named about_*topic*, cmdlet help pages are named *cmdlet*

e.g. help about_

e.g. help get-date

Default Help

- By default, help only displays basic help including DESCRIPTION, SYNTAX, and SEE ALSO sections
- Like most cmdlets, get-help accepts several parameters which modify how it works and what it displays
- Powershell only includes the basic help in the default installation
- More help content is available for most cmdlets by using the -Detailed, -Examples, and -Full parameters

Help Exercises

- Use get-help about_ to view the available topics list
- Try viewing the topic help for command syntax, pipelines, and parameters
- Use get-help with the start, stop, clear, get, and set verbs only to see what nouns are available for those verbs
- Use get-help to get some descriptions for the following sample cmdlets:
- get-process, get-date, get-host, clear-host, stop-job, start-service

Updating Help

- Use the update-help cmdlet to install complete help pages and keep them up to date
- update-help will only update your pages once a day unless you use the -Force parameter
- update-help requires administrator privilege
- update-help should be added to scheduled tasks if you keep local help pages

Online Help

- The -Online parameter can be used to view the latest help for cmdlets and topics on the web

- e.g. help -Online get-help

- The online help allows selection of Powershell version because that defines which cmdlets are available to you and how they work

- Powershell online help does not provide Powershell 1.0 or 2.0 help

ISE Help

- The show-command cmdlet will display a popup window which allows click-based command construction

- You can access help from the show-command popup

- The show-command popup captures input

- The show-command popup is implemented as a pane in ISE, which does not capture the input

Extending Help Exercises

- Run the update-help cmdlet to install full help pages on your computer
- Compare the output for the help get-date cmdlet when using the help cmdlet with no parameters versus using the -detailed, -examples, and -full parameters
- Compare the help -full get-date output to the online version from help -online get-date
- Use show-command to try the help popup and compare it to the command help pane in ISE

Tab Completion

- Parameters in Powershell are words starting with a - character
- Both commands and their parameters can be completed using the tab key
- Repeatedly pressing tab cycles alphabetically through matching choices
- Shift-tab moves backwards through the list of choices
- The list wraps around at both ends
- Pressing Control-space shows a list of possible completions

Parameters in Scripts

- Parameters only require you to type enough characters to uniquely identify a specific parameter
- Cmdlets which require parameters to run will complain when you try to run them without the required parameters
- Parameters can be organized in named sets to avoid conflicts between mutually exclusive parameters
- Always use complete parameter names in scripts
- See about_Parameters for more info

Command Completion Exercises

- In a Powershell console window, try using tab to view all possible parameters for the get-date cmdlet

- In ISE, observe what happens as you type commands and their parameters, use get-random as your sample command for this

- In ISE, use the command list pane to create and run a get-date command that displays the date with day set to 1, hour set to 2, minute set to 3, month set to 4, and year set to 5

- Use control-left click on the cmdlet name in the command list pane to dismiss the cmdlet entry subpane

Execution of Scripts

- On Windows, execution policy determines whether scripts can be run as commands
- Execution policy has scope, there are separate process, user, and system scopes available
- The file extension is used to determine if a file contains a Powershell script
- The extension ps1 means a Powershell script

Execution Policy

- Execution Policy is retrieved using get-executionpolicy
- Execution Policy is set using set-executionpolicy policy (using Run As Administrator) where policy is one of several choices: restricted, allsigned, remotesigned, unrestricted, bypass
- The default policy is restricted, up to 5.1 and prior to Windows Server 2012R2, remotesigned after that
- See about_Execution_Policies for more info
- Execution Policy only exists in Windows

Execution Policy Exercises

- Use get-executionpolicy to see what your policy is currently set to
- Try the -list parameter to see what it is set to for different scopes
- Create a file named helloworld.ps1 with one line it that looks like this:

"Hello World!"

- Try to run your helloworld.ps1 script as a command
- Use set-executionpolicy to set your policy to a mode that allows you to run scripts
- Rerun your script as a command

Command Path

- Like bash, Powershell has a path variable that defines where the shell looks for commands using a semicolon-delimited list of folder names, $env:path
- To see what is in the variable, type the variable name on the command line
- To change it, type $env:path = "$env:path;drive:/new/path/name/to/add"
- Powershell provides a default command path
- You can create a profile file to run startup commands, which is how you might choose to set a different path

Profiles

- Powershell has several recognized profile files
- To see the name of the profile file that applies to your current session, look in the $profile variable
- To see if you have a profile file, run test-path $profile
- To create such a file, try notepad $profile
- You can also create a profile file using

```
new-item -itemtype file -force $profile
```

- See about_Profiles for more info

Profile Exercises

- Create a folder on your PC named scripts in your home directory (or network drive if you prefer) and move your helloworld.ps1 script to that directory

- Create a file named myprofileexample.ps1 in that directory to hold your profile file and put a line in it adding the folder you created in the previous step to your $env:path

- Add that same line to your $profile file on your PC and start a new powershell

- Verify you can run helloworld.ps1 without entering a path to the command