PowerShell Providers

Working with Provider cmdlets -1

In this lecture we will take a look at several different **cmdlets** that are designed for use with **Powershell Providers**. If you want to follow along you will need to download the lesson that came with this lecture. We will be copying and pasting commands into PowerShell.

If you would like a complete list of provider cmdlets and links for further information, download the document called PSProvider cmdlet links.pdf.

Location Cmdlets – These cmdlets are used for directory navigation. The cd (change directory) command can be used to navigate between directories. But, as the number of directories that we need to track grows, this approach becomes more and more inefficient, as most of these paths are usually too long to type. And that's why location cmdlets can be **extremely useful.**

Get-Location – (alias GL) This cmdlet gets an object that represents the **current directory**. Sets the working location to a specified location

Set-Location – (alias is SL) Sets the working location to a specified location. That location could be a directory, a subdirectory, a registry location, or any provider path.

Push-Location – (alias is **pushd**) Adds ("pushes") the current location onto a location **stack**.

Pop-Location – (alias is **popd**) Changes the current location to the location most recently pushed onto the **stack** by using the Push-Location cmdlet.

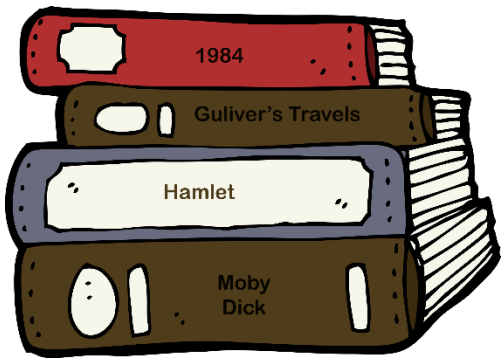
What is a stack? Think of a stack like a stack of books.

1984

Guliver's Travels

Hamlet

Moby Dick

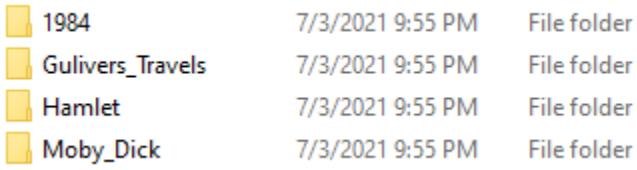


I add books by adding them to the top of the stack. If I want to remove a book, let's say Guliver's Travels, normally I would remove the first book, then remove the second book. In

computer terms this is referred to “Last in, first out” or (LIFO)
So my book stack has two methods, **add** and **remove**. So, in computer terms a stack has two methods push and pop. An item is **pushed** to the stack (added) or **popped** off (removed) from the stack.

I can demonstrate this by creating 4 folders.

Let’s say that we need to work with these four folders.



From your C: drive create a folder called books, then create the four sub-folders. Stop the video while you do this.

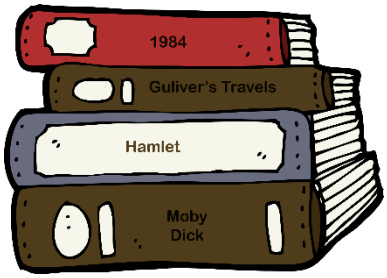
Now lets **add** all four folders to the stack by using our **pushd** alias cmdlet

Type pushd c:\books\1984 (pressh return)

Type pushd c:\books\gulivers_travels

Type pushd c:\books\hamlet

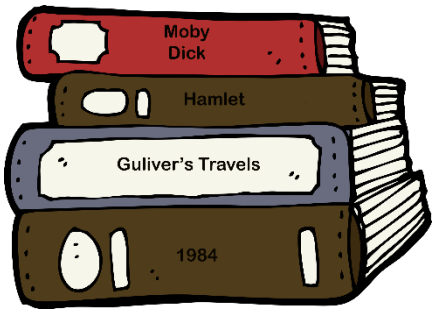
Type pushd c:\books\moby_dick



Now let’s view the stack by using the get-location -stack (parameter) remember (LIFO) **last in first out**

Type **get-location -stack** (press return)

Because get-location displays our **current location**, moby_dick is in the stack but **not shown** in the stack.



Now from PS C:\books**moby_dick** location type **popd**

Type **get-location -stack** (moby_dick) has been removed from the stack)

From PS C:\books**Hamlet** location type **popd**

Type **get-location -stack** now hamlet has been removed from the stack

From PS C:\books**Gulivers_Travels l**ocation type **popd**

Type **get-location -stack** (gulivers travels has been removed from the stack)

From PS C:\books**1984** location type **popd**

Type **get-location -stack** (1984 has been removed from the stack)

Type get-location -stack and now the stack is **empty**.

As you can see we can move between these four folders and easily remove the folders from the stack that are no longer needed.

Set-Location

You can use this cmdlet to access PowerShell drives. This cmdlet changes the current location to a new location. The location could be a **registry location**, folder or subfolder. To understand how this works type:

Type **Set-Location c:\windows**

Type **Set-Location –Path Env:**

Type **Set-Location -Path HKCU:**

Command #1
set-location software\microsoft\windows\currentversion\uninstall

press **return**

To drill down into the **uninstall location** you can use the **Get-childitem** cmdlet.
Type **get-childitem**

Get-childItem - Gets the items in one or more specified locations.

If you are installing software and you need the software version or the uninstall string, you can use this command to find that string.

Additional examples for using get-childitem
Use get-childitem to **Search for all the DOCX files in the Documents and settings folder and subfolders.**

Command #2
get-childitem -path 'C:\Documents and Settings' -include "*docx" -Recurse -file

Parameters:

-Path This parameter specifies the path of one or more locations.
-include This is a string parameter and when this parameter is used, it displays specific files and folders.

-recurse This parameter instructs powerShell commands such as Get-ChildItem to repeat in sub directories.
-file The file attribute gives an output of only files under that container

Now press **return** The errors are from the search not finding the correct path to the files.

Now let's take a look at **one of the Item cmdlets**

New-Item - Creates a new item and sets its value. This can be a folder, file or multiple files inside a directory. Use **get-help New-item** to view all the parameters.

let's use the New-Item cmdlet to Create a folder on your C: drive called PSTest

Command #3

New-Item -Path C:\ -Name "PSTest" -Itemtype "directory"

For Parameters
-Itemtype Can be a file or a folder. Now press **return**
We can check for the folder in windows explorer.

Let's Create a file called test1.txt inside the PStest directory, and add txt to the file.

Command #4

New-Item -Path C:\PSTest -Name "test1.txt" -Itemtype "file" -Value "Hey boss, I want a 50% raise."

Parameters
-Value allows you to add txt as a value. And press **return**

And if you open windows explorer again, you can click on test1.txt and checkout our value Hey boss I want a 50% raise.

Now let's try to overwrite an existing file using the **-confirm** parameter.

Command: #5

New-Item -Path C:\PSTest -Name "test1.txt" -Itemtype "file" -Confirm

- The result is that the -confirm parameter displays a warning, and asks you to confirm this.
- If you want to overwrite the existing file just add the **-force** parameter.

Let's go ahead and create multiple files in a designated folder.

Command #6

New-Item -ItemType "file" -Path "C:\PSTest\child1", "C:\PSTEST\child2"
and press **return.**

Now from Windows Explorer if we open the PSTest folder there are our two files that we just created.

Content Cmdlets

Now let's take a look at **Content cmdlets. These cmdlets** Append, clear, get and replace the content of a file.

Add-Content – Appends content to a file.

- From the PSTest folder on your C: drive create a file called **test.txt, use the command New-Item to create the file.**

Command #7

New-Item -path "C:\PSTest" -name "test.txt" -itemtype "file"

Use the **add-content** cmdlet to append the following text to the file. **Let's go to lunch by Noon.**

Command #8

Add-content "C:\PSTest\test.txt" -Value "Let's go to lunch by noon"
and press **return.**

And open windows explorer and click on **test.txt**, and you will see the added txt that we appended to the file.

Now delete the contents of the **test.txt** file using the **clear-content** cmdlet.

Command #9

Clear-content "C:\PSTest\test.txt"
and press **return**

Now from Windows Explorer, If we take a look at test.txt we can see that the contents have been cleared.

A this point we are done with this lecture, good job and we will see you in the next lecture.