PowerShell Tips, Tricks, and Snippets

Snippets

- Restart Wifi Adapter : `Get-NetAdapter "Wi-Fi" | Restart-NetAdapter`
- Cmdlet documentation : `help Get-Process`
- List process by name, id and WS: `Get-Process | Select-Object -Property ProcessName, Id, WS`
- Select process using the most memory: `Get-Process | Sort-Object -Property WS | Select-Object -Last 5`
- Enable DHCP on PC :

```
$id = Get-NetIPInterface -InterfaceAlias 'wi-fi' -AddressFamily 'IPv4' | Select-Object -ExpandProperty ifIndex

Set-NetIPInterface -ifIndex $id -AddressFamily 'IPv4' -Dhcp Enabled
```

- `Get-Process explorer | Select-Object -Property processname -ExpandProperty modules | Format-List`
- Select unique character from array: `"a","b","c","a","a","a" | Select-Object -Unique`
- Select newest and oldest event in eventlog for powershell :

```
$a = Get-EventLog -LogName "Windows PowerShell"
$a | Select-Object -Index 0, ($A.count - 1)
```

- List of the content of "sever.txt" but skip the first one, and create a new PSsession from it : `New-PSSession -ComputerName (Get-Content Servers.txt | Select-Object -Skip 1)`
- List only readonly file, then rename them, them display the 5 first file :

```
Get-ChildItem *.txt -ReadOnly |
    Rename-Item -NewName {$_.BaseName + "-ro.txt"} -PassThru |
    Select-Object -First 5 -Wait
```

- `Get-Process | Select-Object -Property ProcessName,{$_.StartTime.DayOfWeek}`
- `$days = @{l="Days";e={((Get-Date) - $_.LastAccessTime).Days}}`
- `@{ name = 'a' ; weight = 7 } | Select-Object -Property name, weight`
- Where-Object Tutorial :

```
# 2 ways to use Where-object: 'Script Block' and 'Comaprison Statement'
# Script Block
Get-Process | Where-Object {$_.PriorityClass -eq "Normal"}
```

```
# Comparison Statement
Get-Process | Where-Object -Property PriorityClass -eq -Value "Normal"
Get-Process | Where-Object PriorityClass -eq "Normal"

('hi', '', 'there') | Where-Object Length -GT 0
('hi', '', 'there') | Where-Object {$_.Length -gt 0}

# Get all Stopped services :
Get-Service | Where-Object {$_.Status -eq "Stopped"}
Get-Service | where Status -eq "Stopped"

# List processes with workingSet greater than 250MB
Get-Process | Where-Object {$_.WorkingSet -GT 250MB}
Get-Process | Where-Object WorkingSet -GT (250MB)

# Get process based on Process name
Get-Process | Where-Object {$_.ProcessName -Match "^p.*"}
Get-Process | Where-Object ProcessName -Match "^p.*"

# Use Comparison statement format
Get-Process | Where-Object -Property Handles -GE -Value 1000
Get-Process | where Handles -GE 1000

# Get command based on Property
# Use Where-Object to get commands that have any value for the OutputType property
of the command.
# This omits commands that do not have an OutputType property and those that have
an OutputType
# property, but no property value.
Get-Command | where OutputType
Get-Command | where {$_.OutputType}

# Use Where-Object to get objects that are containers.
# This gets objects that have the **PSIsContainer** property with a value of $True
and excludes all
# others.
Get-ChildItem | where PSIsContainer
Get-ChildItem | where {$_.PSIsContainer}

# Finally, use the Not operator (!) to get objects that are not containers.
# This gets objects that do have the **PSIsContainer** property and those that
have a value of
# $False for the **PSIsContainer** property.
Get-ChildItem | where {!$_.PSIsContainer}
# You cannot use the Not operator (!) in the comparison statement format of the
command.
Get-ChildItem | where PSIsContainer -eq $False

# Use multiple condition
Get-Module -ListAvailable | where {
    ($_.Name -notlike "Microsoft*" -and $_.Name -notlike "PS*") -and
$_.HelpInfoUri
}
```

- Character to start line comment: `\#`
- `$env:PATH`
- `$env:java_custom = "A custom java value"`
- `$env:java_custom += " is special"`
- Create new File in Current directory : `New-Item -Path . -Name "testfile1.txt" -ItemType "file" -Value "This is a text string."`
- Create new Folder : `New-Item -Path "c:\" -Name "logfiles" -ItemType "directory"`
- Create directory "scripts" in "c:\ps-test\ : `New-Item -ItemType "directory" -Path "c:\ps-test\scripts"`
- Create multiple files : `New-Item -ItemType "file" -Path "c:\ps-test\test.txt", "c:\ps-test\Logs\test.log"`
- Similar to `ls C:\Temp`: `Get-ChildItem -Path C:\Temp\`
- Create 'temp.txt' in all directories matched by 'Path' argument: `New-Item -Path C:\Temp\* -Name temp.txt -ItemType File | Select-Object FullName`
- Create folder 'TestFolder' : `New-Item -Path .\TestFolder -ItemType Directory`
- Create file 'TestFile.txt' : `New-Item -Path .\TestFolder\TestFile.txt -ItemType File`
- Force recreation folder 'TestFolder' : `New-Item -Path .\TestFolder -ItemType Directory -Force`
- A curl and wget replacement : `$Response = Invoke-WebRequest -URI https://www.bing.com/search?q=how+many+feet+in+a+mile`
- List content of a directory : `Get-ChildItem -Path C:\Test\`
- List content of a directory recursively: `Get-ChildItem -Path C:\Test\*.txt -Recurse -Force`
- `Get-ChildItem -Path C:\Test\* -Include *.txt`
- `Get-ChildItem -Path C:\Test\Logs\* -Exclude A*`
- `Expand-Archive -Path Draftv2.zip -DestinationPath C:\Reference`
- `Expand-Archive -LiteralPath 'C:\Archives\Draft[v1].zip' -DestinationPath C:\Reference`

Don't know what this one does

```
$username = 'windows-user'
$password = 'myPassword'
$pass = ConvertTo-SecureString -AsPlainText $Password -Force
```

```
$SecureString = $pass

$secureCred = New-Object -TypeName System.Management.Automation.PSCredential -
ArgumentList $Username,$SecureString

# Start-Process powershell "-File .\script_enable_dhcp.ps1" -Credential
$secureCred

# $id = Get-NetIPInterface -InterfaceAlias 'wi-fi' -AddressFamily 'IPv4' | Select-
Object -ExpandProperty ifIndex

# Set-NetIPInterface -ifIndex $id -AddressFamily 'IPv4' -Dhcp Enabled

Read-Host -Prompt "press to continue ..."


# Start-Process powershell "Set-NetIPInterface -ifIndex $id -AddressFamily 'IPv4'
-Dhcp Enabled" -Credential $secureCred
```

Powershell script to install font family

```
if ('S-1-5-32-544' -notin
[System.Security.Principal.WindowsIdentity]::GetCurrent().Groups) {
    throw 'Script must run as admin!'
}

$source = 'Montserrat.zip'
$fontsFolder = 'FontMontserrat'

Expand-Archive -Path $source -DestinationPath $fontsFolder

foreach ($font in Get-ChildItem -Path $fontsFolder -File) {
    $dest = "C:\Windows\Fonts\$font"
    if (Test-Path -Path $dest) {
        "Font $font already installed."
    }
    else {
        $font | Copy-Item -Destination $dest
    }
}
```

Quick References

- Learn X in Y minutes - Powershell