SEARCH USING POWERSHELL

This document demonstrates using PowerShell to search for file names like 'find' in Linux. It also shows how to search within file contents like 'grep' in Linux. This can be done to some extent in Windows Explorer, but I've found using PowerShell is much faster and more powerful once you master the syntax.
In the examples below I use 'ls' because it's short and easy to type, but you could alternatively use 'dir', 'gci' or 'Get-ChildItem'.

NAME SEARCH

Search file and/or folder NAMES using '-Filter'.

```
# Search 'C:\logs' and subfolders for any file with 'local' in the file name
PS C:\> ls -Path 'C:\logs' -Recurse -File -Force -Filter '*local*' | Select FullName
```

-**Path**: Either use '-Path' like the example above, OR just cd to the folder first that you want to search.
-**Recurse**: Searches subfolders too (without it, subfolders are excluded).
-**File**: Searches for files only. Leaving this off gets both files and folders.
-**Directory**: Searches for directories only. Leaving this off gets both files and folders.
-**Force**: Includes system and hidden files/folders in search. Akin to 'ls -a' in Linux.

- **Select** is an alias for **Select-Object**.
- **ls, dir, gci**, are all aliases for **Get-ChildItem**. See them for yourself in PowerShell like this:

```
PS C:\> Alias -Definition Get-ChildItem
```

CONTENTS SEARCH

Search file CONTENTS by piping **Get-ChildItem** results to **Select-String** similar to 'grep' in Linux.

```
# Search for 'DPI' in the text of all batch files in the current directory
PS C:\> ls -Filter '*.bat' | Select-String 'DPI'

# Same as above, but only show each file once (summary instead of detailed output)
PS C:\> ls -Filter '*.bat' | Select-String 'DPI' | Select-Object -Unique Path
```

-**Pattern**: It's optional to spell this out (i.e., "Select-String -Pattern DPI" is the same as "Select-String DPI").
-**Context**: This parameter is very handy. It allows you to show lines before and after the string occurrence.
-**AllMatches**: A Select-String option. The default action of Select-String, is to only find the first occurrence on each line of text. Using this however, finds all matches in each line (multiple occurrences per line). It may make no noticeable difference in PowerShell 5.1, but in PowerShell 7.x, you'll notice a difference in the highlighting.

- **Select-String** has an alias of **sls**, whereas **Select-Object** has an alias of **Select**. Don't confuse the two.
- "**Select-Object -Unique Path**" is very handy when you just want to list file(s) that contain the string, but not detail out where each occurrence is within each file. For example, if a file contained 'DPI' ten times, you'd normally see ten results from it, but using this, you would only see one result per file.

EXAMPLES

Recursively search for any file (including hidden files) for any that contain 'fil' in the name:

```
PS C:\Scripts> ls -Recurse -Force -Filter '*fil*'

    Directory: C:\Scripts

Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----         4/22/2021  11:05 AM           3453 Get-DuplicateFiles.ps1
-a----         4/13/2020  10:17 PM           5899 New-IsoFileFunction.ps1
-a----        10/17/2013  11:01 AM            613 profile.ps1
-a----         2/15/2021   3:40 PM           1111 Rename-47Files.ps1
```

The difference between using 'Select-Object -Unique Path' or not:

```
PS C:\Scripts> ls -Filter '*.ps1' | Select-String address

Get-ListeningPorts.ps1:41:##  Changed to include all listening addresses instead of
Get-ListeningPorts.ps1:42:#   Where-Object { $_.LocalAddress -eq "0.0.0.0"}
Get-ListeningPorts.ps1:43:    Select-Object LocalAddress,LocalPort,
Get-ListeningPorts.ps1:55:##  Changed to include all listening addresses instead of
Get-ReservedIPs.ps1:6:    This script is used to get all the reserved IP addresses
of one DHPC server.
Show_Local_System_Info.ps1:90:        @{Name='IpAddress';Expression={$_.IpAddress
-join '; '}},
Show_Local_System_Info.ps1:94:        WinsPrimaryServer, WINSSecondaryServer|
ConvertTo-html -Body "<H2>IP Address </H2>" >>  "$filepath\$vComputerName.html"

                File          Line Number

PS C:\Scripts> # Same command as above, but only show summary
PS C:\Scripts> ls -Filter '*.ps1' | Select-String address | select -Unique Path

Path
----
C:\Scripts\Get-ListeningPorts.ps1
C:\Scripts\Get-ReservedIPs.ps1
C:\Scripts\Show_Local_System_Info.ps1
```

Using the '-Context' parameter to show the string and its adjacent lines:

```
PS C:\Scripts> ls .\Robo-BUp.ps1 | Select-String read

Robo-BUp.ps1:59:[void][System.Console]::ReadKey($FALSE)  # This will pause until a ke
Robo-BUp.ps1:81:    #$Src =     Read-Host "Enter the SOURCE path              (e.g.,
Robo-BUp.ps1:82:    #$Dest =    Read-Host "Enter the DESTINATION path (e.g., 'B:\Back
Robo-BUp.ps1:83:    #$LogPath = Read-Host "Enter a temp directory for the log file  (
Robo-BUp.ps1:101:    $Proceed = Read-Host "Continue with these settings (y/N)?"


PS C:\Scripts> # Same command as above, but show context (1 line before and after)
PS C:\Scripts> ls .\Robo-BUp.ps1 | Select-String read -Context 1

  Robo-BUp.ps1:58:Write-Host "Press any key to continue."
> Robo-BUp.ps1:59:[void][System.Console]::ReadKey($FALSE)  # This will pause until a
  Robo-BUp.ps1:60:Clear-Host
  Robo-BUp.ps1:80:    $LogPath = Get-Folder -Description "LOG FILE TEMP FOLDER" -Init
> Robo-BUp.ps1:81:    #$Src =     Read-Host "Enter the SOURCE path              (e.g.
> Robo-BUp.ps1:82:    #$Dest =    Read-Host "Enter the DESTINATION path (e.g., 'B:\Ba
> Robo-BUp.ps1:83:    #$LogPath = Read-Host "Enter a temp directory for the log file
  Robo-BUp.ps1:84:    #$Log =     $LogPath + $LogFile
  Robo-BUp.ps1:100:    Write-Host ""
> Robo-BUp.ps1:101:    $Proceed = Read-Host "Continue with these settings (y/N)?"
  Robo-BUp.ps1:102:    if($Proceed -ne 'y') {
```