PowerShell Fundamentals Practice Lab

Objective

This lab provides hands-on practice with fundamental PowerShell concepts, including file system navigation, accessing help resources, displaying system information, and automating tasks using scripts.

Lab Exercises

Exercise 1: Navigating the File System

1. Open PowerShell and execute the following commands:

- Display the current directory:
- Get-Location
- List all files and folders in the current directory:
- Get-ChildItem
- Create a new directory named TestFolder:
- New-Item -ItemType Directory -Name 'TestFolder'
- Navigate into TestFolder:
- Set-Location .\TestFolder
- Create a text file named SampleFile.txt:
- New-Item -ItemType File -Name 'SampleFile.txt'
- Delete the file SampleFile.txt:
- Remove-Item .\SampleFile.txt
- Navigate back to the parent directory:
- Set-Location ..

Exercise 2: Accessing PowerShell's Help System

1. Get an overview of the Get-ChildItem command:

   Get-Help Get-ChildItem

2. View examples of how to use Get-ChildItem:

   Get-Help Get-ChildItem -Examples

3. Update the help system (requires internet access):

   Update-Help

4. Use help to explore the New-Item command:

   Get-Help New-Item -Full

Exercise 3: Display System Information

- Retrieve the hostname of the system:
- hostname
- Display detailed system information:
- Get-ComputerInfo
- View the current user's username:
- $env:USERNAME
- Get information about the operating system:
- Get-WmiObject -Class Win32_OperatingSystem
- Display a list of running processes:
- Get-Process

Exercise 4: Beginner Automating Tasks with Scripts

1. Task: Write a script that displays system information in a formatted way:

```
# SystemInfo.ps1
$os = Get-WmiObject -Class Win32_OperatingSystem
$cpu = Get-WmiObject -Class Win32_Processor
$memory = Get-WmiObject -Class Win32_PhysicalMemory | Measure-Object -Property Capacity -Sum

Write-Output "System Information:"
Write-Output "--------------------"
Write-Output "OS: $($os.Caption)"
Write-Output "CPU: $($cpu.Name)"
Write-Output "Total RAM: $([math]::Round($memory.Sum / 1GB, 2)) GB"
Write-Output "Username: $env:USERNAME"
Write-Output "Hostname: $env:COMPUTERNAME"
```

Challenge

Create a script that:
1. Retrieves the top 5 processes consuming the most memory.
2. Writes the process name, ID, and memory usage to a text file named TopProcesses.txt.

Hint: Use Get-Process and sort the processes by their WorkingSet property.