Advanced Scripting   
Legacy Shells and PowerShell

Last Updated: 4/18/2024 8:14 AM Version 2  
Document Prepared for: CYBER360 Student

# Name Click here to enter name ID Click here to enter id

# Instructions

Save a copy of this document. Use Microsoft Word to edit and answer all questions directly in this document. You will save and upload this completed document as your homework submission.

# Overview

These exercises are intended as a review of other command shells you have used previously.

# Setup

Launch a Windows legacy Command Prompt.  
-- To find it: press the [Flag] key (or [Ctrl]+[Esc]), then start typing **cmd.exe**.

Also, launch a Linux bash shell.  
-- If you installed WSL: [Flag], type **bash**, then under best matches, look for it under “Command.”  
-- If you use macOS, Linux, or a Linux VM: launch its Terminal app.

Also, launch PowerShell Core.  
-- To find it in Windows: [Flag], then start typing pwsh.exe.  
-- In Linux or macOS: launch another Terminal app, then enter **pwsh**.

# Task 1—Navigate the file system using legacy shells.

## Steps

1. *Print Working Directory*: show the current working directory (CWD) location as a “path” sequence of directories and subdirectories (folders and subfolders). *Metaphor: think of this as the shell’s “cursor” location within the filesystem’s tree structure.*
   1. In CMD.EXE: type   
      **cd**   
      and press [Enter]. You should see output similar to: **C:\Users\Alice**. What output was produced in your Command Prompt console? Click or tap here to enter text.
   2. In bash: type   
      **pwd**   
      and press [Enter]. You should see output such as: /home/alice. What output do you see in your bash terminal? Click or tap here to enter text.
   3. In pwsh: Type   
      **Get-Location**   
      and press [Enter]. Your output: Click or tap here to enter text.   
      The PowerShell command aliases **gl** and **pwd** also work. Try them!
2. *Make Directory*: create a new subdirectory (subfolder). *Metaphor: graft a new “branch” into the filesystem’s directory tree structure.*
   1. CMD: Enter   
      **md stash1**
   2. bash: Enter   
      **mkdir stash2** # (this command also works in CMD and PowerShell)
   3. pwsh: Enter   
      **New-Item -Type Directory stash3**
3. *Change Directory*: set a new CWD location. *Metaphor: think of this as moving the shell’s cursor to a different folder path*.
   1. CMD:
      1. Enter these two commands:  
         **cd stash1**   
         **cd**   
         Record your output here: Click or tap here to enter text.
      2. **cd ..**   
          **cd**   
         Your output: Click or tap here to enter text.   
         (You should be out of the stash1 subdirectory and back in its parent directory.)
      3. Enter   
          **cd \**   
          **cd**   
         Your output: Click or tap here to enter text.   
         (You should be at the top level directory, the “root” folder of the filesystem tree).
   2. bash:
      1. **cd stash2**   
          **pwd**   
         Your output: Click or tap here to enter text.
      2. **cd ..**   
          **pwd**   
         Your output: Click or tap here to enter text.
      3. **cd /**   
          **pwd**   
         Your output: Click or tap here to enter text.
   3. pwsh:
      1. **Set-Location stash3**   
          **Get-Location**   
         Your output: Click or tap here to enter text.
      2. The aliases **sl** and **cd** also work in PowerShell:   
          sl ..   
          gl   
         Your output: Click or tap here to enter text.
      3. **cd /**   
          pwd   
         Your output: Click or tap here to enter text.
4. *List directory contents*: show a list of the files and subdirectories contained in the CWD.
   1. CMD:
      1. **dir** # (You should see the contents of the root-level folder of your Windows filesystem.)
      2. **cd %userprofile%**   
          **dir** # (You should see the contents of your account’s home user folder.)
   2. bash:
      1. **ls** # (You should see the contents of the root-level of your Linux/macOS filesystem.)
      2. **cd ~**   
          **ls** # (You should see the contents of your Linux/macOS home user folder.)
   3. pwsh:
      1. **Get-ChildItem** # (You should see the contents of the root level folder.)
      2. **Set-Location ~**   
          **Get-ChildItem** # (You should see your home folder’s contents.)   
         *Note:* **Set-Location** *by itself (without* **~** *) also sets the CWD to your home folder*.
5. *External command*: use the external “tree” console program to show the tree structure of the entire filesystem.
   1. CMD: **tree \**   
      (*Note:* this will probably tie up your shell with a lot of output, because the Windows tree structure has thousands of folders and subfolders in various paths. It’s not worth your time to wait for it to finish outputting everything! Instead, impatiently press **[Ctrl]**+**C** to interrupt the command and get back to the prompt.)
   2. bash: If you don’t already have the tree utility installed (likely), you’ll need to install it first:
      1. **sudo apt install -y tree** (This command installs tree in Ubuntu or Debian; use your favorite search engine to research equivalent commands in macOS, Fedora, etc.)
      2. **tree /** (Again, [Ctrl]+C to interrupt.)
   3. pwsh: entering the name of any external command launches that program as if it were executed in a legacy shell. In this example, PowerShell Core in Windows executes the same **tree** as CMD, PowerShell Core in Linux executes the same **tree** as bash.
6. There are many more commands for other filesystem tasks like moving files and folders, renaming them, deleting them, changing their attributes and permissions, etc. If you don’t remember, or haven’t yet learned, you can always consult a search engine or an LLM generative AI chatbot to find out how to do those tasks in a legacy shell. Later, we’ll practice PowerShell filesystem tasks in more depth. For now, let’s move on to other tasks.

# Task 2—Operate input and output streams

## Steps

1. Use **>** to redirect output to a file instead of to the scrolling display.
   1. CMD: **echo payment:100 > myinfo1.txt**
   2. bash: **echo payment:200 > myinfo2.txt**
   3. pwsh: **Write-Output payment:300 > myinfo3.txt**   
      (The **echo** and **write** aliases also work.)
2. Use **>>** to append additional output to an existing file.
   1. CMD: **echo expense:91 >> myinfo1.txt**
   2. bash: **echo expense:92 >> myinfo2.txt**
   3. pwsh: **write expense:93 >> myinfo3.txt**
3. Get content from a file.
   1. CMD: **type myinfo1.txt**
   2. bash: **cat myinfo2.txt**
   3. pwsh: **Get-Content myinfo3.txt**
      1. Enter:   
         **Get-Alias -Definition Get-Content**   
         What are your shorter aliases for **Get-Content** ?   
         List them all here on just one line: Click or tap here to enter text.
4. Use **|** to “pipe” the output of one command into the input of another command.
   1. CMD:
      1. **type myinfo1.txt | find "pay"**   
         Your output: Click or tap here to enter text.
      2. **tree \ | more**   
         (Press **space** to see more output, **Q** to quit the more command, and **[Ctrl]**+**C** to interrupt the tree command)
   2. bash:
      1. cat myinfo2.txt | grep pay   
         Your output: Click or tap here to enter text.
      2. **tree / | more**   
         (Press **space** to see more output, **Q** to quit more.)
   3. pwsh: **Get-Content myinfo3.txt | Select-String pay**   
      Your output: Click or tap here to enter text.

# Task 3—Managing processes

## Steps

1. List all running processes.
   1. CMD: **tasklist**
   2. bash: **ps ax**
   3. pwsh: **Get-Process**
2. Use a process identifier (PID) to terminate a running process.
   1. CMD:
      1. Enter   
          **calc**   
         to launch a Calculator app window.
      2. **tasklist | find "Calculator"**   
         What is the PID of your Calculator app’s process? Click or tap here to enter text.
      3. **taskkill /F /PID** *<PID number>*   
         Example: my Calculator’s PID was 8124, so I entered: **taskkill /F /PID 8124**
   2. bash:
      1. Launch another bash terminal console. In that second console, start the interactive password changer utility: **passwd** (You need not change your password; just ignore the “current password” prompt and leave it idle.)
      2. Back in your original bash terminal:   
         **ps ax | grep passwd**   
         What is the PID of your password-changer’s process? Click or tap here to enter text.
      3. Again in your original bash terminal:   
         **kill -KILL** *<PID number>*   
         Example: my passwd PID was 84, so I entered: **kill -KILL 84**
   3. pwsh:
      1. Launch one of the above programs (**calc** if you’re using PowerShell Core in Windows, **passwd** if you’re using PowerShell Core in Linux).
      2. **Get-Process | more**   
         Find the PID number of your process.   
         (Or, if you prefer, pipe into a suitable **Select-String** command instead of **more**.)
      3. **Stop-Process -Id** *<PID number>*
      4. **Get-Alias -Definition Stop-Process**   
         What are shorter aliases for Stop-Process? Click or tap here to enter text.

# Task 4—Access built-in help documentation

## Steps

1. Show help for the command that lists directory contents.
   1. CMD: **dir /?**
      1. Notice that the command option **/B** changes the output to “bare” format.
      2. Try it:   
         **dir \ /B**
      3. What command option adds file ownership information to the output? Click or tap here to enter text.
   2. bash: **man ls**   
      (Press Q to quit reading the “manpage” and return to the shell prompt.)
      1. Notice that the command option **-1** (that’s the numeral “one”) changes the output to “single-column” format. This is like the “bare” format of CMD’s **dir**.
      2. Try it:   
          **ls -1 /**
      3. What option produces a “long listing,” adding file permission codes, ownership, size, and modification date information to the output? Click or tap here to enter text.
   3. pwsh: **Get-Help -Online Get-ChildItem**
      1. This opens a new browser tab to show the online help documentation.
      2. In every new installation of PowerShell Core, the built-in help system is kept sparse, so that it doesn’t use too much local disk storage space. In other words, by default, most of the help manuals are online rather than built-in.
      3. You can download the help manuals. On your everyday laptop or workstation, you *should* do so. Here’s how: launch a new PowerShell Core prompt with administrator privileges:  
           
         -- In Windows, find it in the Start menu and tap “Run as Administrator.”  
         -- In Linux or macOS, open a new Terminal and enter **sudo pwsh**, and If necessary, authenticate to escalate to root privilege.)  
           
         At that new administrator PowerShell Core prompt, enter **Update-Help**. This command will download the online documentation to PowerShell Core’s local built-in help system, which will take a while. Some help packages may not be available on every system, so don’t freak out if you see some red-colored warning or error messages. You might see hints in those errors. You can follow those hints to get some of the content that Update-Help couldn’t get by itself.
      4. Okay, now that the built-in help system is updated, you can get help locally at the console prompt instead of in a browser: **Get-Help Get-ChildItem**
      5. Notice that the command-line switch option **-Name** changes the output to names-only “bare” format. Try it:   
         **Get-ChildItem -Name**
      6. What command-line switch option causes **Get-ChildItem** to list not just the contents of a folder, but also the contents of all its child locations (subfolders), and their child locations (sub-subfolders), and so forth? Click or tap here to enter text.
2. Explore built-in documentation for other commands, such as the process listers:
   1. CMD: **tasklist /?**
   2. bash: **man ps**
   3. pwsh: **Get-Help Get-Process**
3. That’s enough review for now. Use the **exit** command to close each of your command prompt, PowerShell, and bash terminal windows.

# Deliverable

Upload this document with completed answers to I-Learn Canvas.