Advanced Scripting   
Install Toolsets

Last Updated: 3/13/2024 10:30 AM Version 1  
Document Prepared for: CYBER360 Student

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

# Instructions

Save a copy of this document. Answer all questions directly in this document. You will save and upload this completed document as your homework submission.

# Overview

Install most of the software that will be necessary for upcoming Advanced Scripting study and work.

# Setup

You should have plenty of free disk space (persistent storage) on your laptop computer.

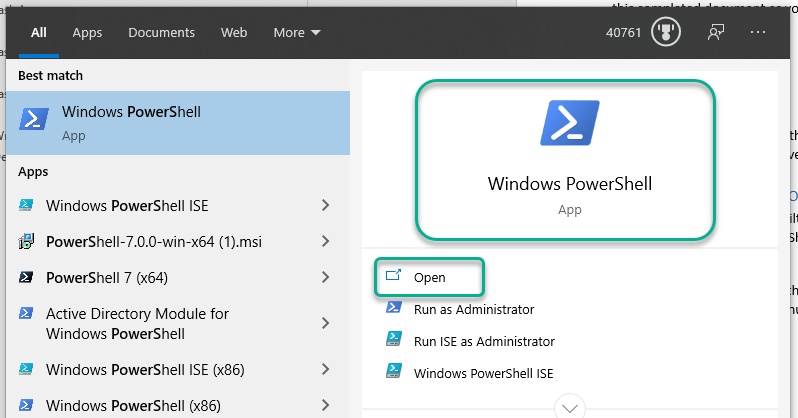
# Task 1—Microsoft Windows

If your laptop already runs Windows 11, you’re ready to go. Windows PowerShell (Version 5, Desktop Edition) is already installed. Skip ahead to the “Steps” at the end of this task.

If your laptop runs Linux or macOS, be prepared to do some research using search engines as you complete the following:

* If you don’t already have one, download and install hypervisor software capable of running a virtual machine. A couple of popular and recommended options are:
  + VirtualBox. Visit <https://www.virtualbox.org/> to find your download links.
  + VMware Player. Use a search engine with query “Download VMware Player” to find it.
    - Their no-cost Linux hypervisor is called “VMware Workstation Player.”
    - Their macOS product is called “VMware Fusion Player.” It is also free, but you’ll need to register for an account with VMware to download it.
* Your computer’s firmware must be configured to enable virtualization. The procedure is dependent on the make and model of your computer, so if you need to figure this out, try including the make and/or model in your careful search engine queries.
* Download virtual-DVD installation media (probably a .ISO file) for Microsoft Windows 11. Students should be able to find this at <https://azureforeducation.microsoft.com/devtools> as follows: sign in with your BYU-I credentials, then select [Software] from the left menu and find “Windows 11 Education.”
  + BYU-I Students should be authorized for Windows 11 downloads, but sometimes a student isn’t. Try Windows Server 2022 instead if you can’t get Windows 11. Be sure you install Windows Server with its “Desktop Experience” option.
  + macOS users: if your laptop has the new Apple Silicon (ARM architecture) processor, you will instead need to obtain Windows 11 ARM installation media. Students should be able to sign up for an appropriate Microsoft account that lets them download this at no cost. Query a search engine to learn how to proceed.
* Use your hypervisor (VMware, Virtualbox, or possibly some other product) to install a new Windows VM from your downloaded installation media file. When your installation is complete, follow the steps below.

## Steps

1. Launch Windows PowerShell using the Start menu.
   1. Click or tap the Start menu button, or press the Windows (“Flag”) key on your keyboard.
   2. Rather than navigate the start menu, instead just start typing PowerShell. When you see Windows PowerShell under “Best match,” click it:
   3. You should then see a Terminal console with a PowerShell command line prompt.
   4. Type $PSVersionTable at that prompt, then hit the [Enter] key.
      1. What is the PSVersion? Click or tap here to enter text.
      2. What is the PSEdition? Click or tap here to enter text.
      3. Note that this version and edition is only for Microsoft Windows. (Later in this exercise you will install and start using the newer cross-platform version called PowerShell Core.)
2. Next, launch an administrative shell. You will need administrative privilege on your system to do so. Again, type PowerShell at the Start menu, but this time, when you see the PowerShell best match, click the Run as Administrator link: A screenshot of a computer

   Description automatically generated
   1. After you accept the User Access Control (UAC) authorization notification, you should see another Terminal with a PowerShell prompt, but this time it will show “Administrator” in the title bar of the new window. There are some commands that require administrator privilege. Try the following:
      1. Enter Get-ExecutionPolicy.  
         What is the resulting output? Click or tap here to enter text.
      2. Enter Set-ExecutionPolicy -ExecutionPolicy Unrestricted
      3. Again, Get-ExecutionPolicy.   
         Now what is the resulting output? Click or tap here to enter text.
3. The temporary folder C:\TEMP will be useful in upcoming work. Enter the following “pipeline-chain,” which will activate the temporary folder, creating it if it doesn’t already exist:   
   cd C:\TEMP || New-Item -Type Directory C:\TEMP && cd C:\TEMP

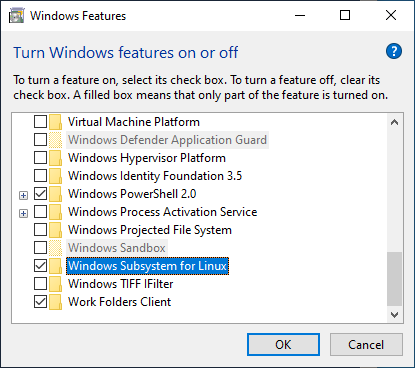
# Task 2—Linux

If your laptop already runs Linux or macOS, you’re ready to go. Skip ahead to the “Steps” at the end of this task.

If your laptop runs Windows, your most convenient option is Windows Subsystem for Linux. If you haven’t already installed this on your system, you may add it as follows, or you may use a search engine to find other tutorials for installing WSL.

* Note: if you already have a hypervisor installed, such as VirtualBox or VMware Workstation Player for Windows, you may need to tweak various settings to successfully install and use WSL. Use careful search engine queries if you need to troubleshoot. As a last resort, you may install and use a Linux VM instead, but give WSL a try first.

Enable WSL:

* From the PowerShell prompt, enter: OptionalFeatures.exe
* This will open the Windows Features configuration utility. 
* **Check** the **Windows Subsystem for Linux** checkbox and click **OK**. Be patient; after some time the installation will complete. You may need to restart.
* After Windows restarts open the **Microsoft Store** and search for **Linux.**
* Select a distribution (Linux “flavor”) you want to install. (If you’re not sure, try **Ubuntu**.)
* After it installs, **launch** it. It will take some time to finish installing. (You might also be prompted to install a WSL update, and to enable the Virtual Machine Platform feature.)
* Enter a username and password when prompted.
* Yippee! Linux is now installed.

## Steps

1. For most Linux distributions, your shell will probably be Bash. (For macOS, probably zsh.)
2. At your Linux shell prompt, enter the command: cat /proc/version   
   (For macOS, try “About This Mac” from the top left system menu)  
   What version of Linux (or macOS) is reported? Click or tap here to enter text.

# Task 3—PowerShell Core

The current version of PowerShell enables cross-platform scripting on Windows, macOS, and Linux.

## Steps

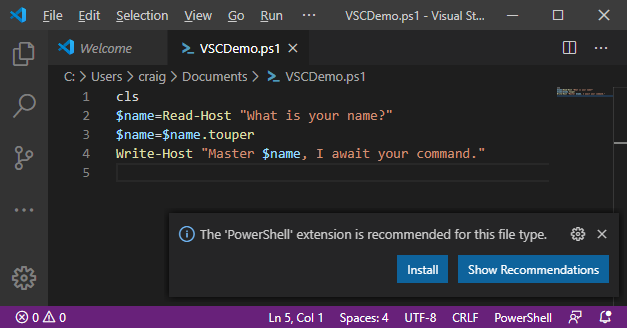
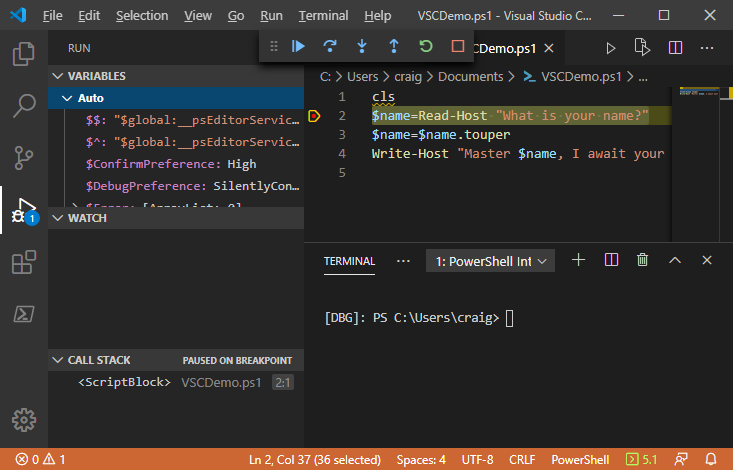
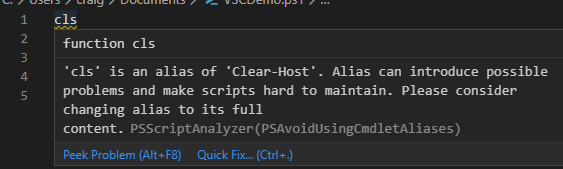
1. First, install PowerShell Core on Windows. Visit <https://github.com/PowerShell/PowerShell>
2. For our class exercises, install the stable release of PowerShell Core. You will probably want the 64-bit version.
   1. Find the link for the version you would like to install and download the installer.
   2. Run the installer
   3. When you get to the Optional actions check all the options.
   4. Done
3. To launch PowerShell Core, you can find it in the Start menu under the name **PowerShell 7**.
   1. If you already have a command prompt open, you could also enter the command: pwsh
4. From your PowerShell Core prompt, enter the command: $PSVersionTable
   1. What is the PSVersion? Click or tap here to enter text.
   2. What is the PSEdition? Click or tap here to enter text.
   3. What is the OS? Click or tap here to enter text.
   4. What is the Platform? Click or tap here to enter text.
5. Now install PowerShell Core on Linux. This task is just a bit more involved. Each distribution is handled a bit differently. Use the instructions on the download page for details: <https://learn.microsoft.com/en-us/powershell/scripting/install/installing-powershell-on-linux>
   1. For example, if you selected Ubuntu as your Linux distribution, click on the link to <https://learn.microsoft.com/en-us/powershell/scripting/install/install-ubuntu>, scroll down to the section “Installation via direct download,” and follow the instructions you find there, which look similar to this:  
        
      # Prerequisites  
      # Update the list of packages  
      sudo apt update  
      # Install pre-requisite packages.  
      sudo apt install -y wget  
      # Download the PowerShell package file  
      wget \  
      https://github.com/PowerShell/PowerShell/releases/download/v7.4.1/powershell\_7.4.1-1.deb\_amd64.deb  
      # Install the PowerShell package  
      sudo dpkg -i powershell\_7.4.0-1.deb\_amd64.deb  
      # Resolve missing dependencies and finish the install  
      # (if necessary)  
      sudo apt install -f  
      # Delete the downloaded package file  
      rm powershell\_7.4.0-1.deb\_amd64.deb  
      # Start PowerShell  
      pwsh  
        
      (If you chose some other distribution, find the appropriate instructions for your selection.)
6. From your PowerShell Core prompt in Linux, enter the command: $PSVersionTable
   1. What is the PSVersion? Click or tap here to enter text.
   2. What is the PSEdition? Click or tap here to enter text.
   3. What is the OS? Click or tap here to enter text.
   4. What is the Platform? Click or tap here to enter text.
7. Congratulations. You now have PowerShell running on Linux.

# Task 4—Visual Studio Code

Install and configure VSCode for PowerShell script development.

## Steps

1. If you don’t have it already, download and run the appropriate VSCode installer from <https://code.visualstudio.com/Download>
2. After installing, if it’s not already running, launch VSCode. (Note: you can also launch it from a command line by entering **code** at a shell prompt.)
3. Create a short script:
   1. Open a new document: **File🡪New Text File**
   2. In the editor pane, enter the following code:  
        
      cls  
      $name=Read-Host "What is your name?"  
      $name=$name.touper  
      Write-Host "Master $name, I await your command."   
        
      (As you type, VS Code may recognize this as PowerShell code, and prompt you to install the recommended PowerShell extension; go ahead and install.)
   3. You should end up seeing something like this: A screenshot of a computer

      Description automatically generated  
      Notice in the screencap shown here that there is no coloring and no PowerShell console. This is for two reasons. First, since we have not saved the file, VSCode does not know what kind of code file it is. Second, even if VSCode knows what kind of file it is, VSCode does not have a PowerShell Extension installed. Let’s fix these problem/s. First, save the file:
      1. Press Ctrl+S to save the file (or save it from the **File🡪Save** menu). Name the file:  
         VSCDemo.ps1 (Make sure you include the .ps1 extension)
      2. Shortly after saving, you should see colored code a message at the bottom of your screen 
      3. VSCode realized that this is a PowerShell file and suggested a recommended extension to work with the file. Click **Install** to install the extension. (You might have already done this previously in step 3.2.)
      4. Now notice that you now have color coding in the editor pane, and a PowerShell console in the Terminal pane.
   4. To run the script, Press F5 now.
      1. You should see the prompt in the console window. Type your name, then press Enter.
   5. In the editor pane, place your cursor somewhere on line 2, then use F9 to toggle a break point on line 2, then F5 to run the script again. It should stop at and highlight line 2.
   6. You haven’t entered your name yet because execution stopped at the breakpoint. Hover over the $name variable. This is the variable that will contain whatever you will type as input. Observe what happens when you hover over the variable. Does it show a value? Does it just show **$name**? Does nothing happen? Click or tap here to enter text.
   7. Compare your screen to this screen capture: 
4. There is a lot going on here! Let’s dig into it.
   1. The **Variables** window shows all the variables PowerShell is using, both the ones defined by PowerShell, and also the ones the user defined.
   2. The red dot by line 2 shows there is a breakpoint on that line. Think of it as a tiny “stop sign.”
   3. The highlighted line is where the code is currently prepared to execute.
   4. The Debug toolbar is displayed above the editor pane. Hover over the buttons in the toolbar at the top of the screen and record what each icon means.
      1. What does  Do? Click or tap here to enter text.
      2. What does  Do? Click or tap here to enter text.
      3. What does  Do? Click or tap here to enter text.
      4. What does  Do? Click or tap here to enter text.
      5. What does  Do? Click or tap here to enter text.
      6. What does  Do? Click or tap here to enter text.
5. Tap the “continue” button, then in the Terminal pane, enter your name and finish running the script.
6. Fix the code so it works correctly: change the third line to $name=$name.ToUpper()
7. Remove the break point by clicking on the red dot. Run it again to make sure it works.
8. Clean up the code:
   1. Notice there is a yellow squiggly under the cls on line 1. This indicates there is a potential issue with the code. Hover your mouse over the squiggly line. You should see 
   2. As you can see, the *linter* is telling you that cls in an alias, and best practice is to use the full name. If you click or press in the **QuickFix** link, VS Code’s *IntelliSense* will offer suggested fixes. A black rectangle with white text

      Description automatically generated
   3. Click on **Replace cls with Clear-Host**. Then [Ctrl]+S to save your code.

# Deliverable

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