Advanced Scripting   
Lab: Manipulating text-based data

Last Updated: 3/16/2024 6:50 PM Version 1  
Document Prepared for: CYBER360 Student

# Instructions

Work together with your partner to complete these lab activities. Prepare your own script file to define the functions outlined below. At the bottom of your script, include a few example commands that call and successfully execute your functions. *(DO NOT submit one jointly-authored file; each team member should submit their own copy of their team’s script.)*

# Overview

A common task in PowerShell is to manipulate textual data: either

* *mapping* (transforming the data),
* *filtering* (selecting relevant data and discarding irrelevant data), or
* *reducing* (summarizing or reorganizing the data).

For this lab, you and your buddy will write two PowerShell functions:

* The first function will get the Ethernet address of every interface on a machine, then look up and return the name of the vendor associated with each Ethernet address.
* The second function will load a *discography* (a list of song recordings) from a file, then return a formatted report that organizes the songs by album.

The data you’ll need for these functions are in text files found in the course **psfiles.zip** archive. To help you accomplish the lab activities, please leverage Internet resources at your disposal, including advanced search engines and generative-AI LLM chatbots. Collaborate with your partner and share successful searches and chatbot prompts.

# Requirements

* Visual Studio Code
* MACDatabase.txt and RushSongs.txt from the **psfiles.zip** archive.

# Setup

*Recommended:* In VS Code, select **File** > **Open Folder**. In the file selector dialog, find the **psfiles** folder where you extracted the contents of psfiles.zip, and open that folder. This will focus VS Code on that folder: any new scripts will be placed in **psfiles**, and every terminal pane will start with **psfiles** as its current working directory.

# Tasks:

1. Create a script named **manipulate.ps1**.
2. Place the following block comment template at the top of your script:   
     
   **<#  
   Script name: manipulate.ps1  
   Course: CIT 361  
   Date:** *<date created>* **Authors:** *<you and your buddy’s names>* **Affidavit: We affirm this script is our original work.  
   Citations: We leveraged ideas and help from the following sources:  
    --** *<ChatGPT, https://chat.openai.com/>* **--** *<etc.>* **#>**   
     
   (replace each bracketed item in the template with you and your buddy’s information.)
3. First function:
   1. Name your function **Get-MACVendor**.
   2. The function returns either:  
       one string containing a vendor’s full name, or  
       an array of strings containing vendors’ full names.
   3. The function takes two arguments:  
       Use the named parameter **MACAddress** for the first argument.  
       Use the named parameter **DatabasePath** for the *mandatory* second argument.
      1. The **DatabasePath** specifies the file name of the database to use. (Don’t assume a directory location for the database.)
      2. Throw an error if the specified database file can’t be found. (You can use a **throw** statement in your script to raise an exception.)
   4. If a MAC address is specified, return the full vendor’s name of that MAC address.
   5. If a MAC address is not supplied, discover the MAC addresses of the computer’s network adapters, look up the vendor of each address, and return an array containing each vendor’s name.
      1. On a Windows host, you can get each adapter with the **Get-NetAdapter** cmdlet. On Linux, list the adapters with the **ip address** command. On macOS, use **ifconfig**.
      2. Your script should work correctly on both Windows and Linux (or both Windows and macOS).
   6. To test your scripted function, use **psfiles/data/MACDatabase.txt** as the vendor database.
4. Second function:
   1. Name your function **Format-Songs**.
   2. The function returns a formatted text report, based on data retrieved from the file **psfiles/data/RushSongs.txt**.
   3. The function takes two arguments:   
       Use the named parameter DatabasePath for the *mandatory* first argument.   
       Use the named parameter OutPath for the (optional) second argument.
      1. DatabasePath specifies the file name of the song database.
      2. OutPath specifies the file where the formatted report should be saved. If missing, just return the report to the output stream.
      3. Throw an error if the database can’t be found.
      4. Throw an error if the report file can’t be written.
   4. In the report, group songs by album. Under each album, sort song tracks alphabetically by title.
      1. Example output:

R40 Live (2015)

The Story so Far

Clockwork Angels Tour (2013)

Drumbastica

Here It Is!

Peke's Repose

The Percussor

Clockwork Angels (2012)

BU2B  
…

* + 1. (Optional stretch goal: In the report, sort the albums by year, newest to oldest.)

# Hints:

* Use PowerShell interactively (VS Code terminal pane).
* Use the terminal to learn how to access and manipulate the data, until you have figured out what commands and code work best, then put successful code in your script (VS Code editor pane).
* Don’t code like you would in Java or C or Python. Use the strengths of PowerShell.
* Test your functions on your own computer, and also upload and test them on your BO-BOBO and DONPATCH virtual machines.

## Get-MacVendor

* The vendor portion of a MAC address is the first three bytes of the physical Ethernet address. The first three bytes are the first six hexadecimal digits. For example, if the MAC address is **1a:2b:3c:4d:5e:6f**, then the vendor portion is **1a:2b:3c**.
* **Import-Csv** might not be the easiest way to work with the database. (But if you preprocess the file to filter out the comment section at the front of the file, you might get **ConvertFrom-Csv** to work.)
* Windows usually formats MAC addresses using hyphens ( **-** ), while Linux and macOS (and the database) formats MAC addresses using colons ( **:** ).

## Format-Songs

* To add the year to sort by, consider adding a property to the object returned by **Group-Object**.

# Deliverable

Upload your completed script to I-Learn Canvas.

# Scoring Standard (“rubric”)

|  |  |
| --- | --- |
| Completed comment block | 1 point (or -50 points if it’s missing) |
| Error Handling | 4 points |
| MAC vendor lookup | 20 |
| Discover MAC addresses on Windows | 2 |
| Discover MAC addresses on Linux (or macOS) | 3 |
| Song report | 20 |
| TOTAL | 50 points |
| Extra credit: albums sorted by year | 5 |
|  |  |
|  |  |
|  |  |