Scripting Languages: Workshop 5

Pre-requisites:

- If you have not already done so, log into your Linux instance, start VS Code and navigate your way into the **ws5** folder.

- To complete these tasks you will need to place the following nine (9) files into your current working directory (from *Canvas -> Week 5 -> Workshop Tasks -> ws_files.zip*):
    - **sampledata.txt**
    - **salesdata.csv**
    - **datetimes.txt**
    - **checkstrings.sh**
    - **strlist.txt**
    - **puncount_errors.sh**
    - **text.txt**
    - **charcount.sh**
    - **stringset.txt**

Write the Code

Task 1

1. Write a script named **grep1.sh** that outputs to the terminal a count of the lines in **sampledata.txt** that *start* with the uppercase string *GET*

```
$ ./grep1.sh
60
```

2. If you encounter an error, read the error message printed to the terminal carefully and attempt to resolve the issue and run the **grep1.sh** script again or ask your tutor for assistance

Task 2

1. Write a script named **grep2.sh** that outputs to the terminal those lines in **sampledata.txt** that end with the code **573**

```
$ ./grep2.sh
GET /DVWA HTTP/1.1 301 573
GET /DVWA HTTP/1.1 301 573
GET /DVWA HTTP/1.1 301 573
GET /DVWA HTTP/1.1 301 573
GET /DVWA HTTP/1.1 301 573
GET /DVWA HTTP/1.1 301 573
GET /DVWA HTTP/1.1 301 573
GET /DVWA HTTP/1.1 301 573
GET /DVWA HTTP/1.1 301 573
GET /DVWA HTTP/1.1 301 573
GET /DVWA HTTP/1.1 301 573
GET /DVWA HTTP/1.1 301 573
GET /DVWA HTTP/1.1 301 573
GET /DVWA HTTP/1.1 301 573
GET /DVWA HTTP/1.1 301 573
GET /DVWA HTTP/1.1 301 573
```

2. If you get no results, the file may be in a DOS format, so run it through the **dos2unix** utility and then run your script again

```
$ dos2unix sampledata.txt
dos2unix: converting file sampledata.txt to Unix format...
```

3. If you encounter an error, read the error message printed to the terminal carefully and attempt to resolve the issue and run the **grep2.sh** script again or ask your tutor for assistance

Task 3

1. Write a script named **grep3.sh** that outputs to the terminal those lines in **sampledata.txt** that contain the file extension **jsp** or **py** or **asp** or **aspx**

```
$ ./grep3.sh
GET /4MlfjsG9.py HTTP/1.1 404 503
GET /ISizwTQw.asp HTTP/1.1 404 504
GET /WHxNxiyL.jsp HTTP/1.1 404 504
GET /jJ7szEYv.aspx HTTP/1.1 404 505
GET /iAio8STI.py HTTP/1.1 404 503
GET /qdWQOh2J.jsp HTTP/1.1 404 504
GET /YTl8rbT3.asp HTTP/1.1 404 504
GET /LlyDciYR.aspx HTTP/1.1 404 505
```

2. If you encounter an error, read the error message printed to the terminal carefully and attempt to resolve the issue and run the **grep3.sh** script again or ask your tutor for assistance

Task 4

1. Write a script named **grep4.sh** that outputs to the terminal a count of all the lines in **sampledata.txt** that contain HTTP error **404** and that also end in code **506**

```
$ ./grep4.sh
4
```

2. If you encounter an error, read the error message printed to the terminal carefully and attempt to resolve the issue and run the **grep4.sh** script again or ask your tutor for assistance

Task 5

1. Write a script named **grep5.sh** that retrieves all lines in **sampledata.txt** that end in the IP address **http://192.168.5.162/** and write these lines to a text file named **162attempts.txt**

```
$ ./grep5.sh
$ cat 162attempts.txt
POST /index.php HTTP/1.1 500 1924 http://192.168.5.162/
POST /DVWA/login.php HTTP/1.1 302 384 http://192.168.5.162/
POST /DVWA/login.php HTTP/1.1 302 384 http://192.168.5.162/
POST /index.php HTTP/1.1 200 237 http://192.168.5.162/
POST /index.php/component/users/ HTTP/1.1 200 237 http://192.168.5.162/
```

2. If you encounter an error, read the error message printed to the terminal carefully and attempt to resolve the issue and run the **grep5.sh** script again or ask your tutor for assistance

Task 6

1. Write a script named **grep6.sh** that retrieves all lines in **sampledata.txt** that contain HTTP error code 404 **and** also end in code 506 **and** write these lines to a text file named **404messages.txt**

```
$ ./grep6.sh
$ cat 404messages.txt
GET /phS4AQmy.xhtml HTTP/1.1 404 506
GET /xHEv5Bu9.htmls HTTP/1.1 404 506
GET /gpAe25Jl.xhtml HTTP/1.1 404 506
GET /j81R20RT.htmls HTTP/1.1 404 506
```

2. If you encounter an error, read the error message printed to the terminal carefully and attempt to resolve the issue and run the **grep6.sh** script again or ask your tutor for assistance

Task 7

1. Write a script named **grep7.sh** that outputs to the terminal a count of the lines in **sampledata.txt** that do **not** contain a *three digit code* that *starts* with **50**, i.e. *50x*

```
$ ./grep7.sh
66
```

2. If you encounter an error, read the error message printed to the terminal carefully and attempt to resolve the issue and run the **grep7.sh** script again or ask your tutor for assistance

Task 8

1. Write a script named **regex1.sh** that checks an argument passed in from the command line to ensure it meets four (4) rules, these being:
   I. Is eight (8) characters in length AND
   II. Contains at least three (3) numbers AND
   III. Contains at least one (1) uppercase letter AND
   IV. Contains at least one (1) lowercase letter
2. If the passed in argument complies with all four (4) rules, print to the screen a message that says **Valid! You may enter** and then exit the script with an appropriate exit code, else print **Invalid! You may not enter**, then exit the script with an appropriate exit code
3. Test your script by inputting a wide range of compliant and non-compliant arguments

```
$ ./regex1.sh Ja3s0n1
Invalid! You may not enter.
$ ./regex1.sh ja3s0n1e
Invalid! You may not enter.
$ ./regex1.sh Ja3s0nlE
Invalid! You may not enter.
$ ./regex1.sh JA3S0N1E
Invalid! You may not enter.
$ ./regex1.sh Ja3s0n1E
Valid! You may enter.
```

4. If you encounter an error, read the error message printed to the terminal carefully and attempt to resolve the issue and run the **regex1.sh** script again or ask your tutor for assistance

Task 9

1. Write a script named **`br1.sh`** that prompts the user for the name of a text file that contains a set of long datetime strings and then converts these into an alternate format

```
25/03/2024:16:44:32
26/03/2024:20:20:20
27/03/2024:10:30:02
28/03/2024:15:15:25
29/03/2024:09:10:08
```

datetimes.txt

```
$ ./br1.sh
Enter the name of the file to be processed: datetimes.txt
25-03-2024 16h 44m 32s
26-03-2024 20h 20m 20s
27-03-2024 10h 30m 02s
28-03-2024 15h 15m 25s
29-03-2024 09h 10m 08s
```

2. The file *datetimes.txt* has been provided to you for this purpose
3. Make sure your solution uses a *single* **sed** statement with the required **regex** code to get the outcome shown in the image immediately above

Comment the Code

Task 10

1. Download the files *checkstrings.sh* and *strlist.txt* to your Linux development environment into the ws5 directory you created in Week 1. The *strlist.txt* file is the data source the *checkstrings.sh* script will act upon.
2. Using only the lecture notes (Modules 1-5 inclusive) and what you have learned so far, fully comment the *checkstrings.sh* script to explain:
    a. The purpose of the script
    b. Its inputs
    c. Its main processing logic
    d. Its outputs

*Do **not** run the script before you comment it. Complete the commenting in full and then run the script to see how much of your commenting was accurate.*

*Do **not** ask any AI tool to comment the script for you, otherwise you will learn nothing!!!*

Fix/Debug the Code

Task 11

1. Download the files *puncount_errors.sh* and *text.txt* to your Linux development environment into the ws5 directory you created in Week 1. The *text.txt* file is the data source the *puncount_errors.sh* script acts upon.

2. SCENARIO: A junior team member has come to you for help with a shell script they are writing. It's purpose is stipulated in the comment at the top of the script. The *puncount_errors.sh* script they've written should be producing an output of **3** to the terminal when used on the *text.txt* input file, but rather, it's producing an incorrect count.
3. Examine the *puncount_errors.sh* script the junior team member has brought to you, and:
    a) Clearly identify the issue(s) in the script that are causing the incorrect output
    b) Explain what they have done wrong and how to fix the issue(s)
    c) Modify the script as required so that it produces the correct output; call this file *puncount_corrected.sh*

Use only the lecture notes (Modules 1-5 inclusive) and what you have learned so far to guide you in this process

Use comments to identify/document the issues within the script

*Do **not** ask any AI tool to tell you what the issues are or how to fix them, otherwise you will learn nothing!!!*

Critique the Code

Task 12

1. Download the files *charcount.sh* and *stringset.txt* to your Linux development environment into the ws5 directory you created in Week 1. The *stringset.txt* file is the data source the *charcount.sh* script acts upon.
2. SCENARIO: You asked a junior team member to write a shell script that counts strings from an input file that meet two (2) requirements:
    I. Start with a capital letter, and
    II. Contain no less than three (3) and no more than four (4) numbers.
3. The junior team member has now come to you with the script they've written (*charcount.sh*) and asked if you will approve it for production use. As the senior team member, would you approve this script for use in production? If not, record a short Panopto video explaining to the junior team member why you will not approve their script for production and outline what they need to do to make it acceptable for production use. Then send this video to your lecturer along with your version of the script (call it *charcount_better.sh*) to show the junior team member how you would have coded it as a learning opportunity for them.

*Do **not** ask any AI tool to critique the junior team member's script for you or to write a more efficient version, otherwise you will learn nothing!!!*

Task 13

1. **Copy** the ***.sh*** files you created in today's workshop to the *backups* directory using the same *_bu* name modification you used in last week's workshop
2. Navigate to the *backups* directory and make sure the copy procedure was successful

Conclude:

Close the *RDP connection* to your Azure VM (if you're using one) and then **power off** your VM in Azure.