Scripting Languages: Workshop 4

Pre-requisites:

- If you have not already done so, log into your Linux instance, start VS Code and navigate your way into the **ws4** folder.

- To complete the tasks below, you will need a copy of the following files to your current working directory, all of which can be found in the zip file on Canvas named **ws_files.zip**:
    - **foldernames.txt**
    - **nums.txt**
    - **maketable.sh**
    - **reclist.csv**
    - **wordcounter_errors.sh**
    - **sentences.csv**
    - **charcounter.sh**
    - **strlist.txt**

Write the Code

Task 1

1. Write a script named **`validint.sh`** that prompts the user to enter a *three-digit number code* that is greater than or equal to *110* and less than or equal to 150. Ensure that:
    a. Only a *valid integer* that meets the above-defined rules is accepted
    b. Inputs that do not meet the above-defined rules; *strings*; *nulls*, \n are to be rejected
2. If the validation fails, the user is to be told that the input is invalid and returned to the prompt
3. This process is to continue until a valid input is made, at which point the user is to be given a message that they have provided a valid input

```
$ ./validint.sh
Please enter an integer between 110 and 150 inclusive: 109
The value provided is invalid. Please try again
Please enter an integer between 110 and 150 inclusive: 151
The value provided is invalid. Please try again
Please enter an integer between 110 and 150 inclusive: 140
The value provided is valid
```

```
vbrown@LAPTOP-4EJP6J7N:~/scrlang/lec/week4$ ./validint.sh
Please enter an integer between 110 and 150 inclusive: hello
The value provided is invalid. Please try again
Please enter an integer between 110 and 150 inclusive: ENTER
The value provided is invalid. Please try again
Please enter an integer between 110 and 150 inclusive: 150
The value provided is valid
```

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

Task 2

1. Write a script named **`autofolder.sh`** that populates an array named **`newfolders`** with the folder names listed in the file named **`foldernames.txt`** and then only creates those folders for which the folder names contained within the **`newfolders`** array do **not** exceed *14* characters in length
2. You do **not** need to actually create the folders on your system, just print a message to this effect

foldernames.txt

```
1  humanresources
2  businesssolutions
3  accounting
4  maintenance
5  groundservices
6  motorpool
7  catering
8  managementteam
9  security
10 publicrelations
11 marketingteam
12 foreignaffairs
13 socialclub
14 procurement
15 localgovernment
```

autofolder.sh output

```
$ ./autofolder.sh
Folder humanresources has been created
Folder accounting has been created
Folder maintenance has been created
Folder groundservices has been created
Folder motorpool has been created
Folder catering has been created
Folder managementteam has been created
Folder security has been created
Folder marketingteam has been created
Folder foreignaffairs has been created
Folder socialclub has been created
Folder procurement has been created
The source file contained 15 names, out of which 12 folders were created
```

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

Task 3

1. Write a script named **`addnums.sh`** that *adds* all of the digits in each number contained within a file named *nums.txt* and print this sum to screen as shown in the screenshot below

nums.txt

```
1  245
2  3265
3  458
4  951
5  1598
6  5645
7  887
8  653
```

output

```
vbrown@LAPTOP-4EJP6J7N:~/scrlang/lec/week4$ ./addnums.sh
Sum of digits in 245    - 11
Sum of digits in 3265   - 16
Sum of digits in 458    - 17
Sum of digits in 951    - 15
Sum of digits in 1598   - 23
Sum of digits in 5645   - 20
Sum of digits in 887    - 23
Sum of digits in 653    - 14
```

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

Comment the Code

Task 4

1. Download the files *maketable.sh* and *reclist.csv* to your Linux development environment into the ws4 directory you created in Week 1. The *reclist.csv* file is the data source the *maketable.sh* script will act upon.
2. Using only the lecture notes (Modules 1-4 inclusive) and what you have learned so far, fully comment the maketable.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.*

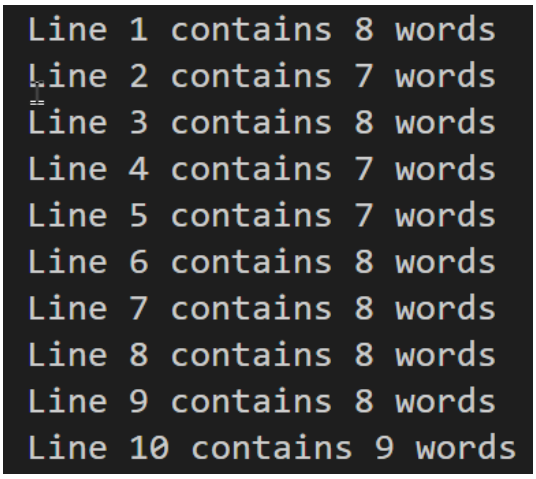
*Do **not** ask any AI tool to comment the script for you, otherwise you will learn nothing!!!*

Fix/Debug the Code

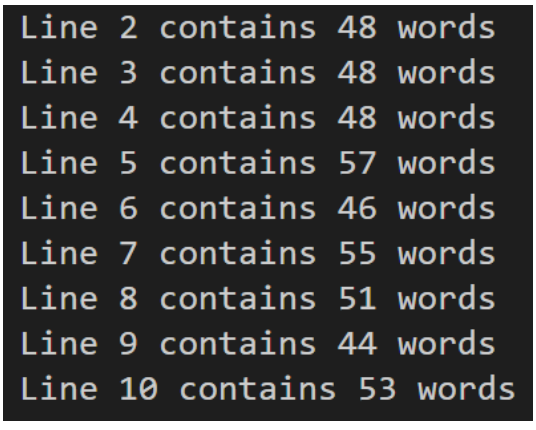
Task 5

1. Download the files *wordcounter_errors.sh* and *sentences.csv* to your Linux development environment into the ws4 directory you created in Week 1. The *sentences.csv* file is the data source the *wordcounter_errors.sh* script acts upon.
2. SCENARIO: A junior team member has come to you for help with a shell script they are writing. The *wordcounter_errors.sh* script they written should be producing the results shown in the image on the *left* below, but rather, is producing the results in the image on the *right* below.

Results should be this…



However, results are like this…



3. Examine the *wordcounter_errors.sh* script the junior team member has brought to you, and:
    a) Clearly identify the issues in the script that are causing the incorrect output
    b) Explain what they have done wrong and how to fix these issues
    c) Modify the script as required so that it produces the correct outputs as shown in the image on the left above; call this file *wordcounter_corrected.sh*

Use only the lecture notes (Modules 1-4 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 6

1. Download the files *charcounter.sh* and *strlist.txt* to your Linux development environment into the ws4 directory you created in Week 1. The *strlist.txt* file is the data source the *charcounter.sh* script acts upon.
2. SCENARIO: You asked a junior team member to write a shell script that counts the number of characters in each of the strings contained within a text file. The junior team member has now come to you with the script they've written (*charcounter.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 *charcounter_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 members script for you or write a more efficient version, otherwise you will learn nothing!!!*

Task 6

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.