# Scripting Languages: Workshop 6

**Pre-requisites:**

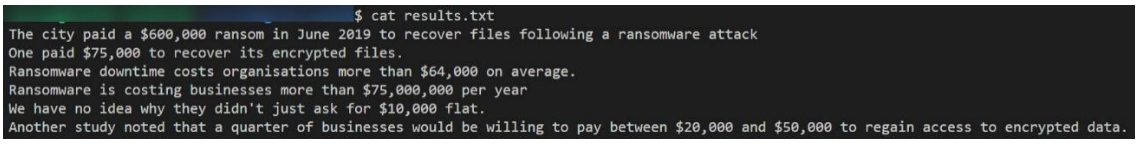
- If you have not already done so, log into your Linux instance, start VS Code and navigate your way into the **`ws6`** folder.

- To complete these tasks you will need to place the following six (6) files into your current working directory (from *Canvas -> Week 6 -> Workshop Tasks -> ws_files.zip*):
    - **`attdata.txt`**
    - **`attdata2.txt`**
    - **`charextract.sh`**
    - **`specharcnt_errors.sh`**
    - **`strlist.txt`**
    - **`strmaths.sh`**

## Write the Code

### Task 1

1. Write a script named **`bigcash.sh`** that retrieves all lines in a given file that contain monetary sums of *$10,000* or greater and writes these to a file named *results.txt*
2. The input file has been provided and is named *attdata.txt*
3. All of the logic required to achieve this outcome is to be contained within a function named **`getsumlines()`**
4. The only code to reside outside of the function is the function call itself, i.e.
   **`getsumlines attdata.txt`**
5. Once the script has executed, run the command **`cat results.txt`**, and you should get the following output to terminal:



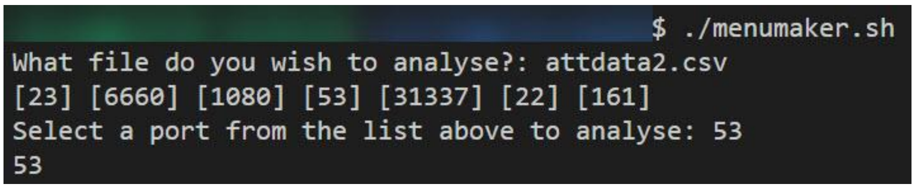
```
[redacted]$ cat results.txt
The city paid a $600,000 ransom in June 2019 to recover files following a ransomware attack
One paid $75,000 to recover its encrypted files.
Ransomware downtime costs organisations more than $64,000 on average.
Ransomware is costing businesses more than $75,000,000 per year
We have no idea why they didn't just ask for $10,000 flat.
Another study noted that a quarter of businesses would be willing to pay between $20,000 and $50,000 to regain access to encrypted data.
```

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

### Task 2

1. Write a script named **`menumaker.sh`** that will create a menu from which the user can choose a *port number* upon which a log file can be analysed
2. A sample log file has been provided with which to develop the script named *attdata2.csv* (the port numbers are the first column of this file)
3. All of the logic required to achieve this outcome is to be contained within a function named **`makemenu()`**
4. The only code to reside outside of the function is the function call itself, i.e.
   **`makemenu() $selfile`**
   …where *$selfile* is the variable that holds the attdata2.csv file name after the user provides it, either at the command line or in response to a prompt

5. The final output of the script to the terminal will be the port number selected by the user
6. Your function code is working as required when you get the following output to terminal:



```
$ ./menumaker.sh
What file do you wish to analyse?: attdata2.csv
[23] [6660] [1080] [53] [31337] [22] [161]
Select a port from the list above to analyse: 53
53
```

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

## Comment the Code

### Task 3

1. Download the files *charextract.sh* to your Linux development environment into the ws6 directory you created in Week 1.
2. Using only the lecture notes (Modules 1-6 inclusive) and what you have learned so far, fully comment the *charextract.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!!!*

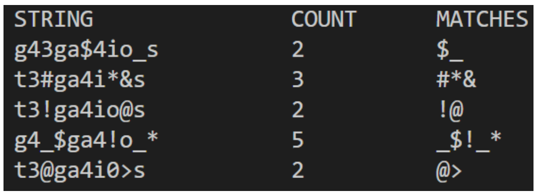
## Fix/Debug the Code

### Task 4

1. Download the files *specharcnt_errors.sh* and *strlist.txt* to your Linux development environment into the ws6 directory you created in Week 1. The *strlist.txt* file is the data source the *specharcnt_errors.sh* script acts upon.

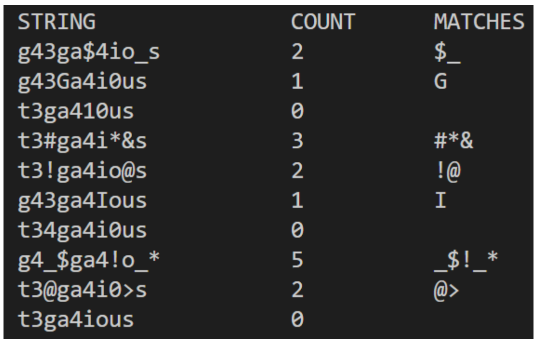
2. SCENARIO: A junior team member has come to you for help with a shell script they are writing. The *specharcnt_errors.sh* script they've written should be producing the results shown in the image on the *left* below, but rather, is producing the results in the image on the *right* below.

**Results should be this…**



```
STRING             COUNT     MATCHES
g43ga$4io_s        2         $_
t3#ga4i*&s         3         #*&
t3!ga4io@s         2         !@
g4_$ga4!o_*        5         _$!_*
t3@ga4i0>s         2         @>
```

**However, results are like this…**



```
STRING             COUNT     MATCHES
g43ga$4io_s        2         $_
g43Ga4i0us         1         G
t3ga410us          0
t3#ga4i*&s         3         #*&
t3!ga4io@s         2         !@
g43ga4Ious         1         I
t34ga4i0us         0
g4_$ga4!o_*        5         _$!_*
t3@ga4i0>s         2         @>
t3ga4ious          0
```

3. Examine the *specharcnt_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 *specharcnt_corrected.sh*

*Use only the lecture notes (Modules 1-6 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 5

1. Download the files *strmaths.sh* and *strlist.txt* (which you already have from the previous task) to your Linux development environment into the ws6 directory you created in Week 1. The *strlist.txt* file is the data source the *.sh* script acts upon.
2. SCENARIO: You asked a junior team member to write a shell script that goes through all the strings contained within an external file, extracts the numbers from each one, and then either "adds" or "multiplies" these numbers depending on whether the script user provides a compulsory **-a** flag (for addition) or an **-m** flag (for multiplication), which would be run as follows:

   **`./strmaths.sh strlist.txt -a`**
   or
   **`./strmaths.sh strlist.txt -m`**

3. The junior team member has now come to you with the script they've written (*strmaths.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 *strmaths_better.sh*) to show the junior team member how you would have coded it as a learning opportunity for them.

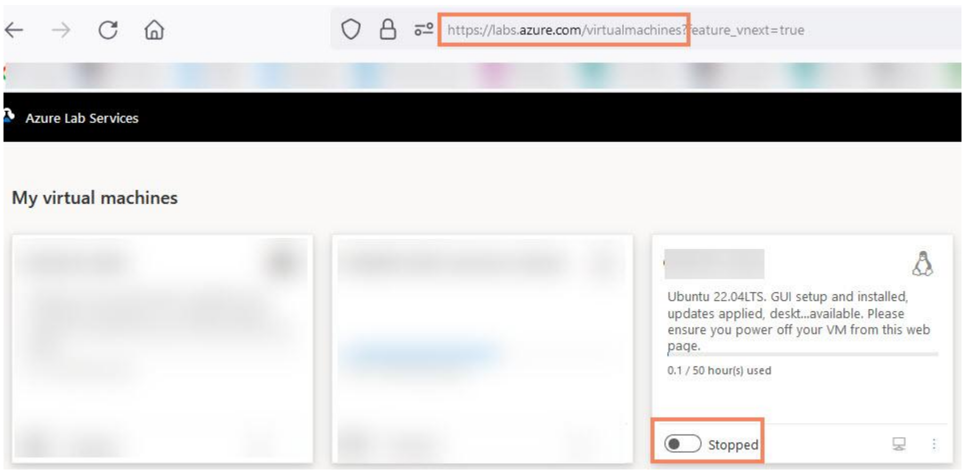
*Do **not** ask any AI tool to critique the junior team members script for you or write a more efficient version, otherwise you will learn nothing!!!*

## Task 6

7. **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
8. 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.



Ubuntu 22.04LTS. GUI setup and installed, updates applied, deskt...available. Please ensure you power off your VM from this web page.

0.1 / 50 hour(s) used