**Data Management Coursework 2**  
**(Linux)**  
   
 Members:

Name: Goh, Yin Xu  
 Email: [yxg1a22@soton.ac.uk](mailto:yxg1a22@soton.ac.uk)  Student ID: 34330119

Name: Abdullah Hashmat Azeez  
 Email: [aha1a22@soton.ac.uk](mailto:aha1a22@soton.ac.uk)   
Student ID: 33400679

1. Unix script for Data collection

My first step here was to learn how to pull data from a website. We learnt about grep and curl here and used learning websites such as [tutorialsPoint](https://www.tutorialspoint.com/unix_commands/grep.htm) and [geeksforgeeks](https://www.geeksforgeeks.org/grep-command-in-unixlinux/) to further learn how to implement them and how to use their extensions. We then assembled the tools with [crontab](https://www.geeksforgeeks.org/crontab-in-linux-with-examples/) as per the coursework guidelines.

The website used was the suggested one for cryptocurrency which was [coinmarketcap](https://coinmarketcap.com/currencies/bitcoin/). The website was available even through an offline html saved page as suggested in the guidelines and had no pesky protection. I then proceeded to start making my scraper and planning my database with MySQL. We had to learn how to install phpMyAdmin on a Debian based system with this [link](https://randomnerdtutorials.com/raspberry-pi-apache-mysql-php-lamp-server/). Below is the final iteration of my first tracker that was going to track the main bitcoin price from the website.

* We had to use variables to setup the grep link with some additional parameters and tools like ‘cut’ and ‘tr’ to format the data in a more favorable way to be stored and scraped. There are also error validation checks scattered through the bash file as annotated with comments.
* There was also an echo to test the output and another echo to store the output with a time stamp in a csv file

We also had to scrape other values off the website for bitcoin, but since these values shared a html tag separate from the main price html tag, there was a need to group this scrape on a different bash file for this reason.

* This one is similar to the previous one except for a few things.
* Still used a master link to curl from and since the grep shares the same link amongst multiple values, we had to figure out how to separate the values.
  + This was done using the sed -n ‘1p’ with the numbers representing the values and the lines
  + There was also regex and whitespace filtering along with error handling in the bash script also.
* The values were assigned and outputted together as echoes and saved to a csv.
* So thus, the data for the bitcoin was scrapped for a period of time, there were some issues with the pi and some data was slightly disjointed.
* We were still some data plots short of the next goal, so we did some more scraping of similar data on another coin.
  + This was done very similar to the previous batch script with some minor adjustments and the biggest being the master curl link changing to the Ethereum link.

Below are some excerpt examples of the code:

Tracker for extra coin details:

#!/bin/bash

# Redirect output to a log file

log\_file="/home/pi4r/Desktop/linuxCoursework/cron2.log"

exec >> "$log\_file" 2>&1

# Define the URL of the cryptocurrency website to scrape

URL="https://coinmarketcap.com/currencies/bitcoin/?timestamp=$(date +%s)"

# Fetch the webpage content using curl

HTML=$(curl -s "$URL")

# Check if curl command was successful

if [ $? -ne 0 ]; then

echo "Error: Failed to fetch webpage content." >> "$log\_file"

exit 1

fi

# Extract the main HTML tag containing all five values

MAIN\_TAG=$(echo "$HTML" | grep -Eo '<dd class="sc-f70bb44c-0 bCgkcs base-text">([^<]+)' )

# Check if extraction was successful

if [ -z "$MAIN\_TAG" ]; then

echo "Error: Failed to extract main HTML tag." >> "$log\_file"

exit 1

fi

# Extracting and formatting each value separately

# Ratio

RATIO=$(echo "$MAIN\_TAG" | sed -n '1p' | sed 's/<[^>]\*>//g' | tr -d '%')

# Check if ratio extraction was successful

if [ -z "$RATIO" ]; then

echo "Error: Failed to extract ratio value." >> "$log\_file"

exit 1

fi

# Circulating Supply

CIRCULATING\_SUPPLY=$(echo "$MAIN\_TAG" | sed -n '2p' | sed 's/<[^>]\*>//g' | tr -d 'BTC,')

# Check if circulating supply extraction was successful

if [ -z "$CIRCULATING\_SUPPLY" ]; then

echo "Error: Failed to extract circulating supply value." >> "$log\_file"

exit 1

fi

# Total Supply

TOTAL\_SUPPLY=$(echo "$MAIN\_TAG" | sed -n '3p' | sed 's/<[^>]\*>//g' | tr -d 'BTC,')

# Check if total supply extraction was successful

if [ -z "$TOTAL\_SUPPLY" ]; then

echo "Error: Failed to extract total supply value." >> "$log\_file"

exit 1

fi

# Max Supply

MAX\_SUPPLY=$(echo "$MAIN\_TAG" | sed -n '4p' | sed 's/<[^>]\*>//g' | tr -d 'BTC,')

# Check if max supply extraction was successful

if [ -z "$MAX\_SUPPLY" ]; then

echo "Error: Failed to extract max supply value." >> "$log\_file"

exit 1

fi

# Fully Diluted Market Cap

FULLY\_DILUTED\_MARKET\_CAP=$(echo "$MAIN\_TAG" | sed -n '5p' | sed 's/<[^>]\*>//g' | tr -d 'BTC,$')

# Check if fully diluted market cap extraction was successful

if [ -z "$FULLY\_DILUTED\_MARKET\_CAP" ]; then

echo "Error: Failed to extract fully diluted market cap value." >> "$log\_file"

exit 1

fi

# Print the extracted data

echo "Ratio: $RATIO"

echo "Circulating Supply: $CIRCULATING\_SUPPLY"

echo "Total Supply: $TOTAL\_SUPPLY"

echo "Max Supply: $MAX\_SUPPLY"

echo "Fully Diluted Market Cap: $FULLY\_DILUTED\_MARKET\_CAP"

# Store the extracted data in a file

echo "$(date +"%Y-%m-%d %H:%M:%S"),$RATIO,$CIRCULATING\_SUPPLY,$TOTAL\_SUPPLY,$MAX\_SUPPLY,$FULLY\_DILUTED\_MARKET\_CAP" >> /home/pi4r/Desktop/linuxCoursework/cryptocurrency\_data3.csv

# Check if data was successfully stored

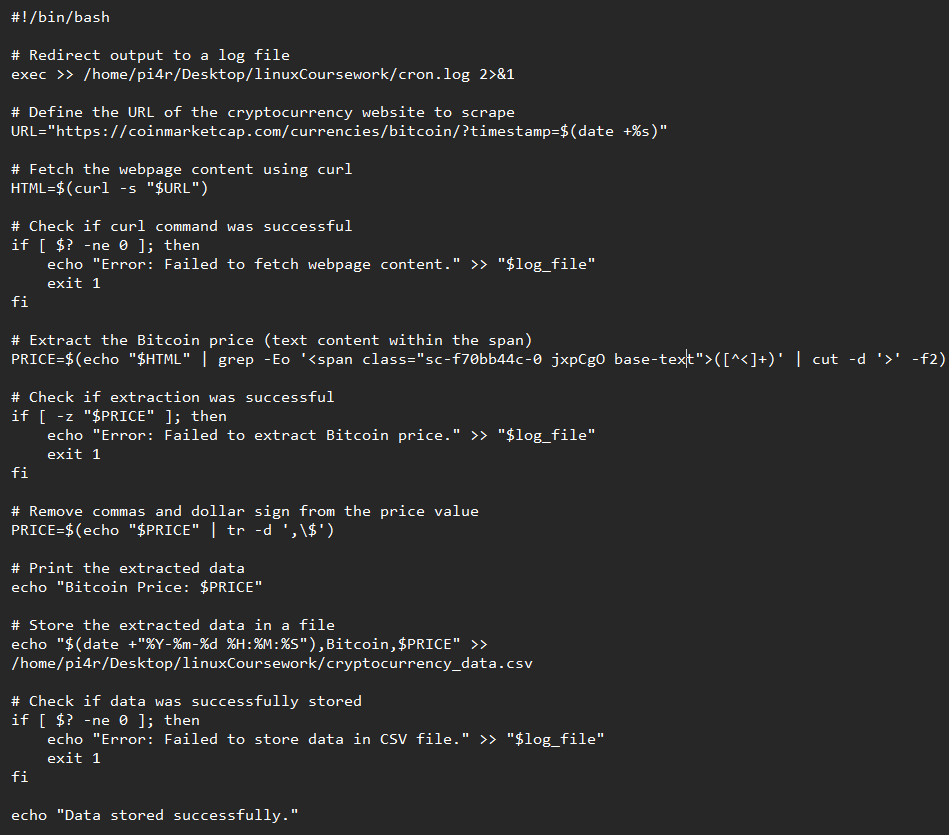
if [ $? -ne 0 ]; then

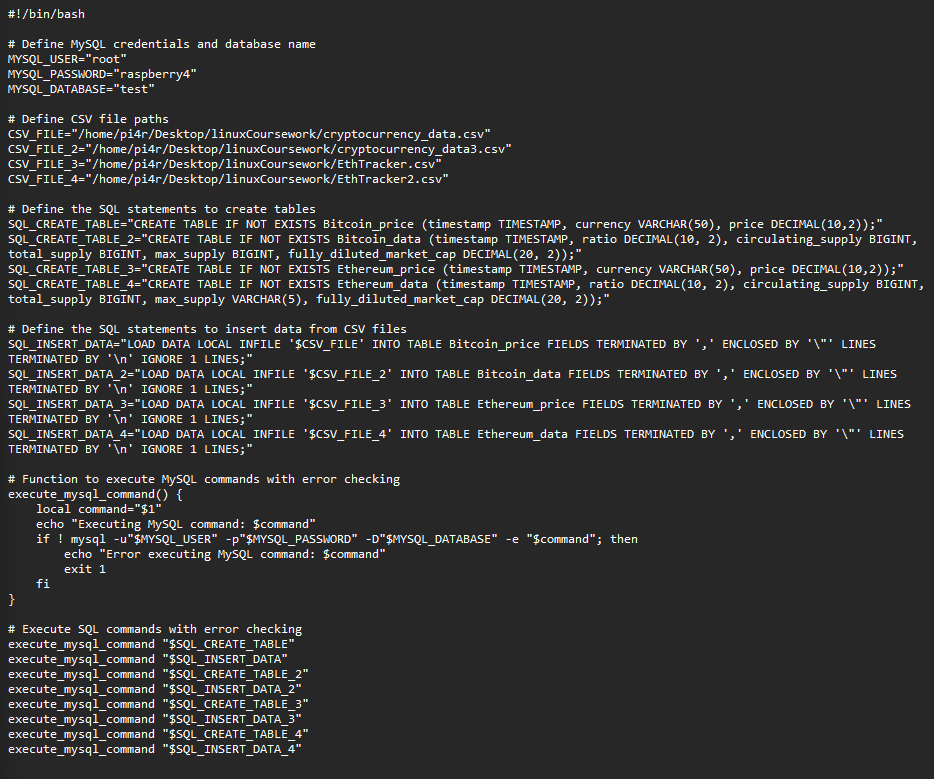
echo "Error: Failed to store data in CSV file." >> "$log\_file"

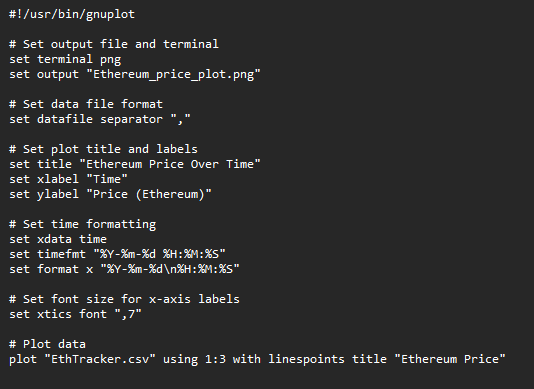
exit 1

fi

echo "Data stored successfully."

Tracker for coin price:  


Tracker for sql insertion:  


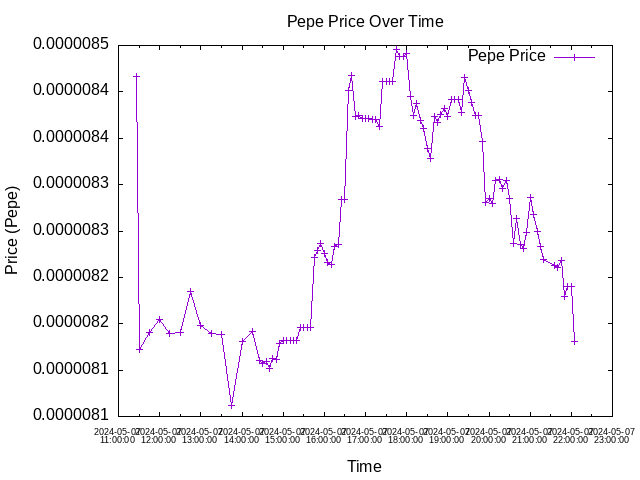
Tracker for Ethereum plot script (similar to bitcoin):  


1. Unix script for plotting

After having plotted a decent amount of data over a length of days over 30 minute or less intervals with crontab, it was time to start plotting the data. A simple search on [gnuplot](http://www.gnuplot.info/) led to an easy sudo apt-get install gnuplot and some learning on the tool itself to use it to generate our plotted graphs.

The generation scripts were drafted and were put as bash scripts to be run in the terminal as instructed by the [coursework](https://www.youtube.com/watch?v=fLKEb-2cAzQ.). Some additional features needed to be picked up such as the ability to change the font size of the x-axis.

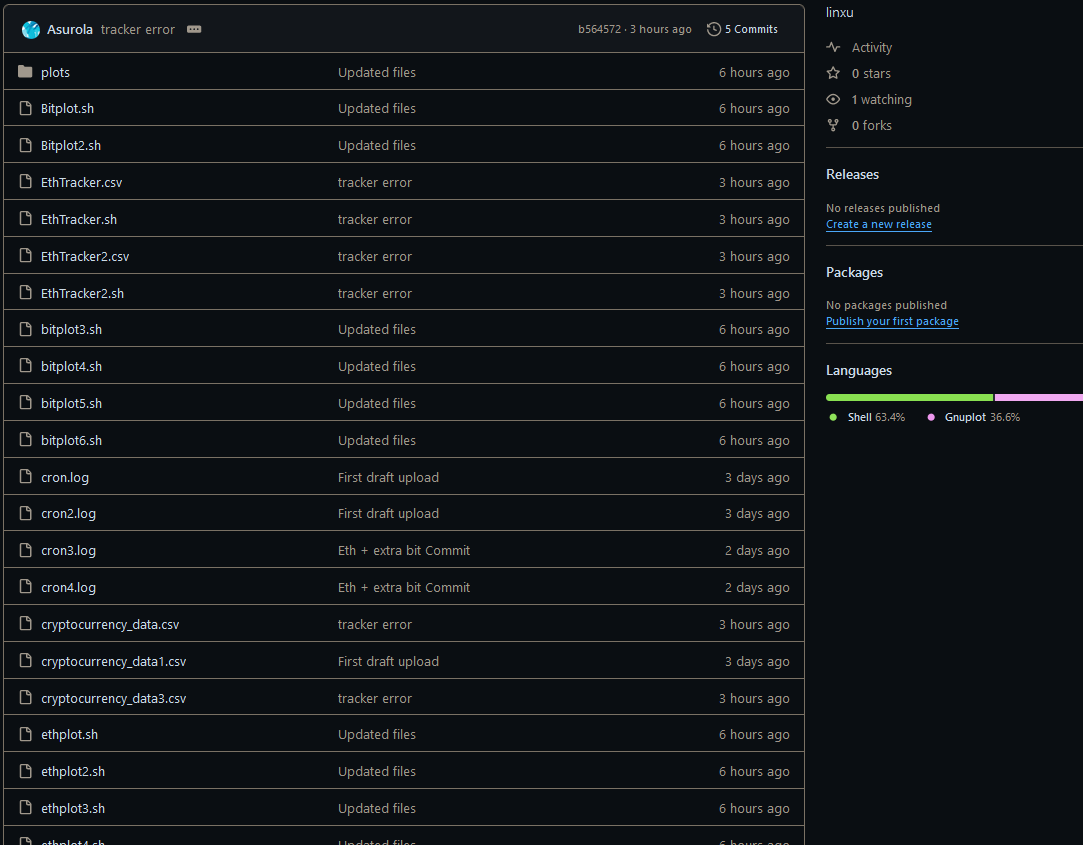
The data was structured to fit the needs of a graph and it was plotted by gnuplot quite hassle-free. Below is an example of a plotted graph over a period with a visible change in the value over time:



1. Use of Git for Version Control

From the moment the coursework started we were committing the files to a github repo to practice professionalism.

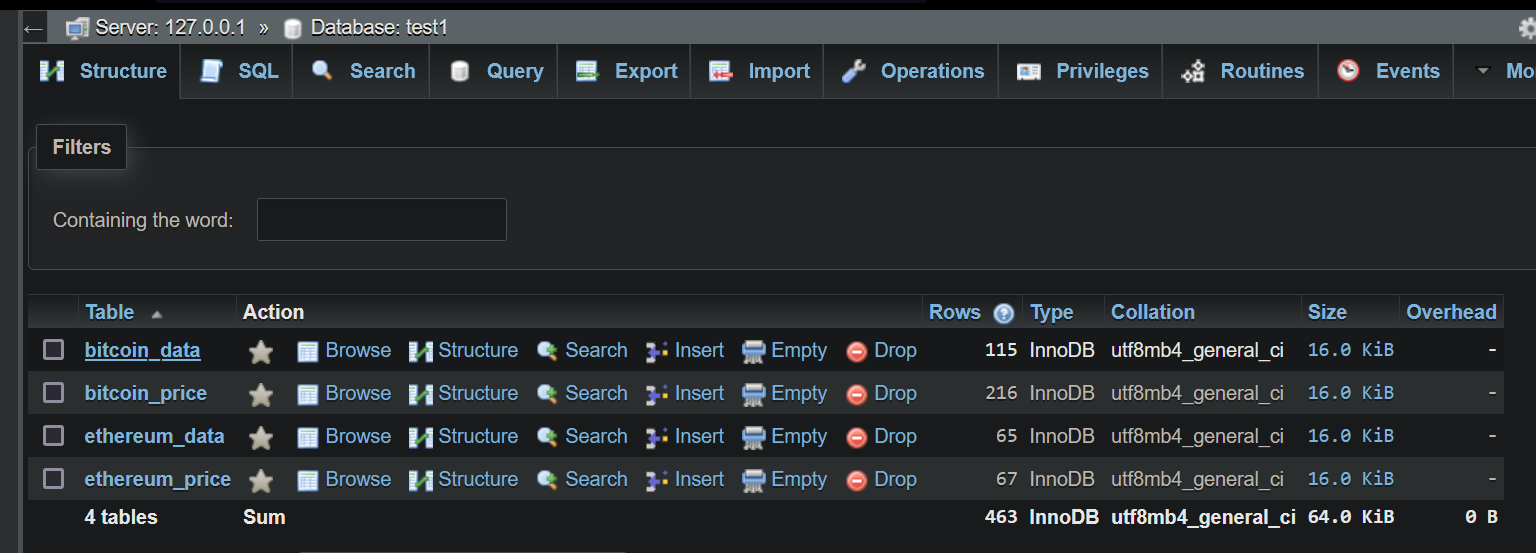
Link: <https://github.com/Asurola/linuxCoursework>



We committed files almost every day to make sure the commits were spread out throughout the week. Updated files overwrote the older version of the files with updated content. Everything was done in bash scripts and sql through the linux terminal after learning the commands. Most commits were done at the end of a day’s work. A link to the github was included in the references as well as above.

1. MySQL for Data Storage

We created a bash script to upload the csv files and the scraped data to a database also created within the bash script on defined database details within the bash script as well. The sql database with its tables will be exported and included in the final zip file.



**References:**

1. [**https://www.tutorialspoint.com/unix\_commands/grep.htm**](https://www.tutorialspoint.com/unix_commands/grep.htm)
2. [**https://www.geeksforgeeks.org/crontab-in-linux-with-examples/?v=fLKEb-2cAzQ**](https://www.geeksforgeeks.org/crontab-in-linux-with-examples/?v=fLKEb-2cAzQ)
3. [**https://randomnerdtutorials.com/raspberry-pi-apache-mysql-php-lamp-server/**](https://randomnerdtutorials.com/raspberry-pi-apache-mysql-php-lamp-server/)
4. [**https://www.geeksforgeeks.org/grep-command-in-unixlinux/**](https://www.geeksforgeeks.org/grep-command-in-unixlinux/)
5. <https://github.com/Asurola/linuxCoursework>
6. [**https://www.youtube.com/watch?v=fLKEb-2cAzQ**](https://www.youtube.com/watch?v=fLKEb-2cAzQ)
7. [**http://www.gnuplot.info/**](http://www.gnuplot.info/)