BASH SCRIPTING

Cryptocurrency Trend

LINUX

Jia Xin Eu

Contents

[Unix Scripting for File Processing 2](#_Toc102602135)

[Crontab Setup 14](#_Toc102602136)

[MySQL for Data Storage 15](#_Toc102602137)

[Database Structure 15](#_Toc102602138)

[Creating database and tables 15](#_Toc102602139)

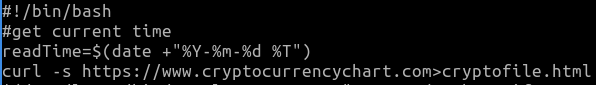
[Inserting data into database 16](#_Toc102602140)

[Script for Plotting 17](#_Toc102602141)

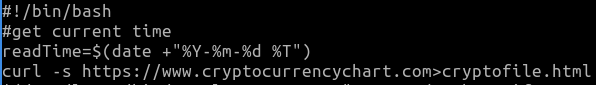
[Use of Git for Version Control 34](#_Toc102602142)

# Unix Scripting for File Processing

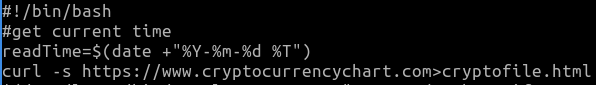
Specify path of bash interpreter



Store current date and time in specified format into variable



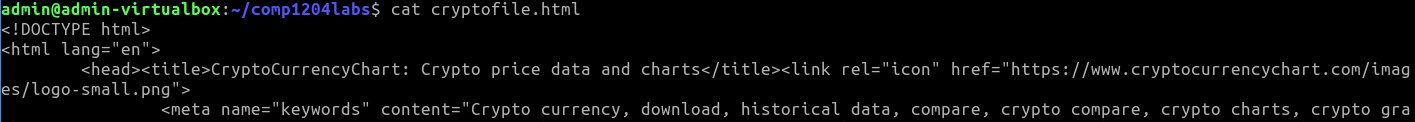
Download source file from website and write it to html file



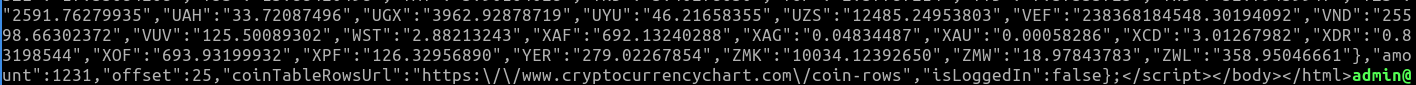
Escape character in html file and assign to variable code for inputting into database



* Read html file



:



* Pipe output to search for expression ‘ and replace with expression \’ where backslash is used to escape character

Eg. Before sed



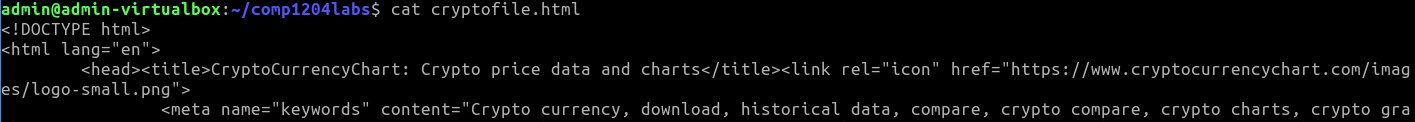
Eg. After sed



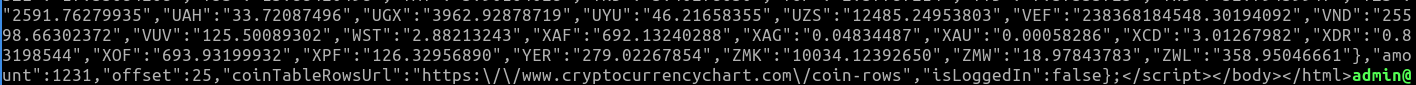
Clean up html file to get crypto coin code and assign to variable code



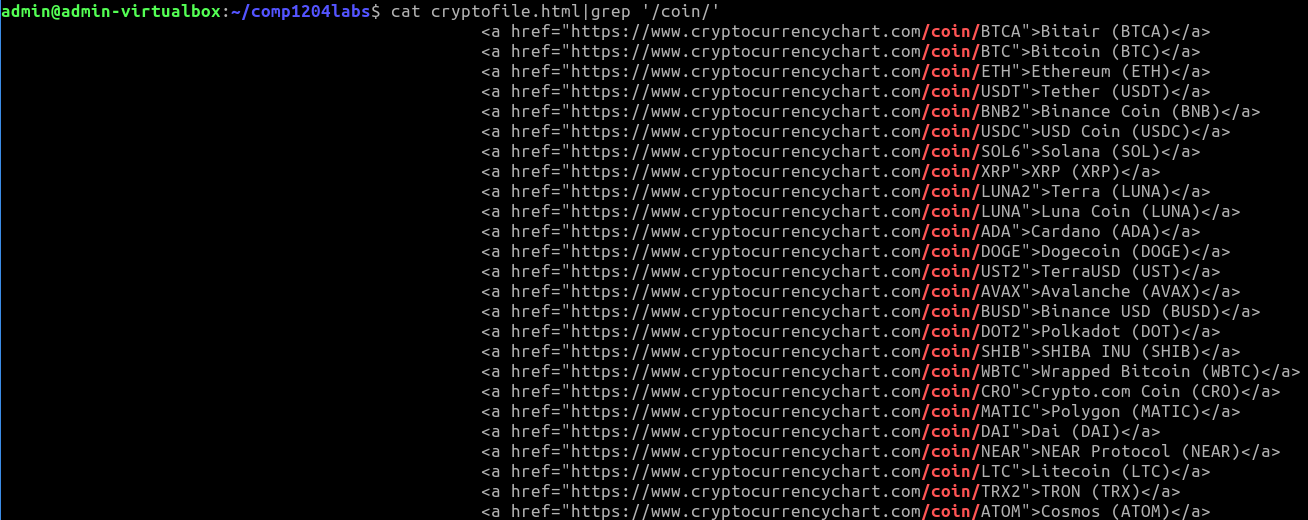
* Read html file



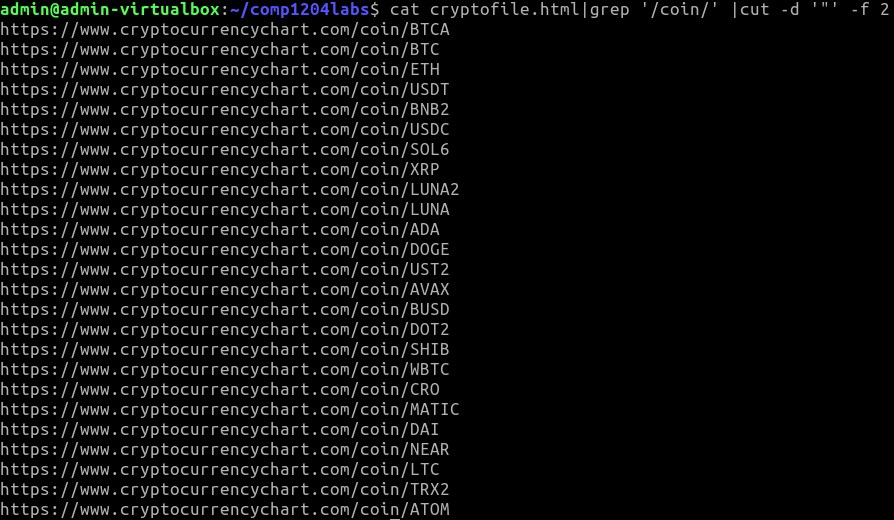
:



* Pipe output to grep line with expression “/coin/”



* Pipe output to cut between 1st and 2nd occurrence of delimiter “



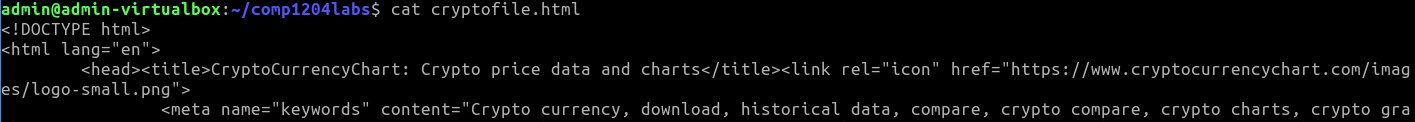
* Pipe output to cut between 4th and 5th occurrence of delimiter /



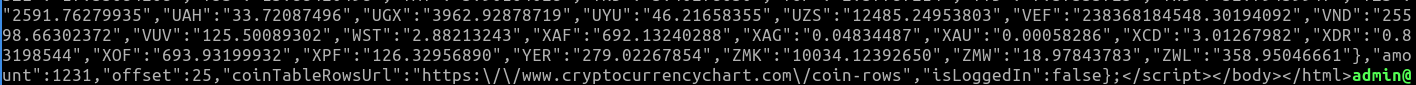
Clean up html file to get crypto coin name and assign to variable name



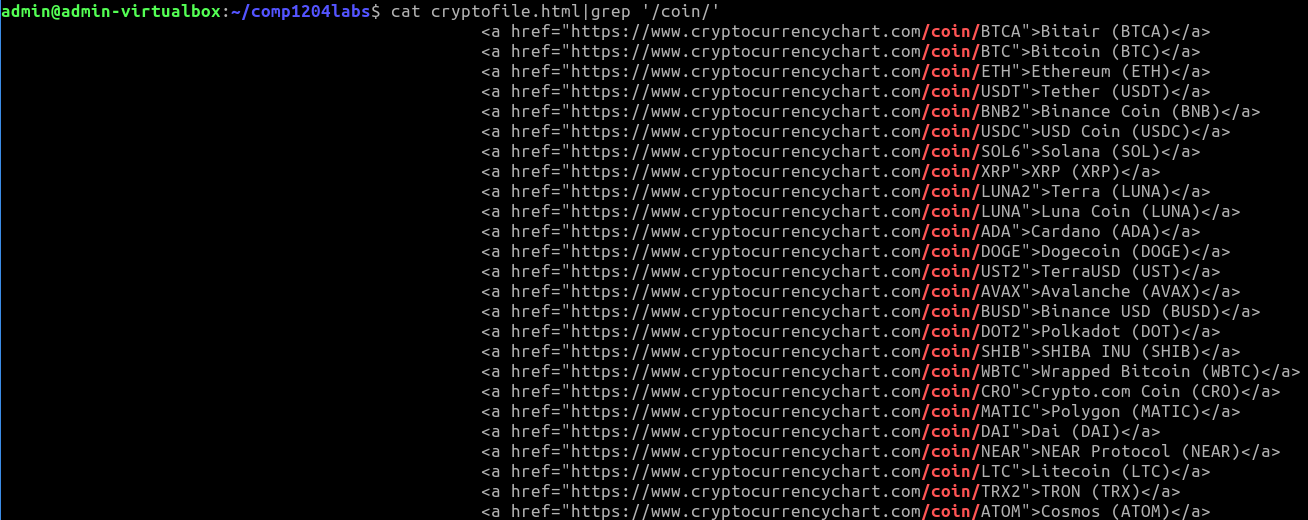
* Read html file



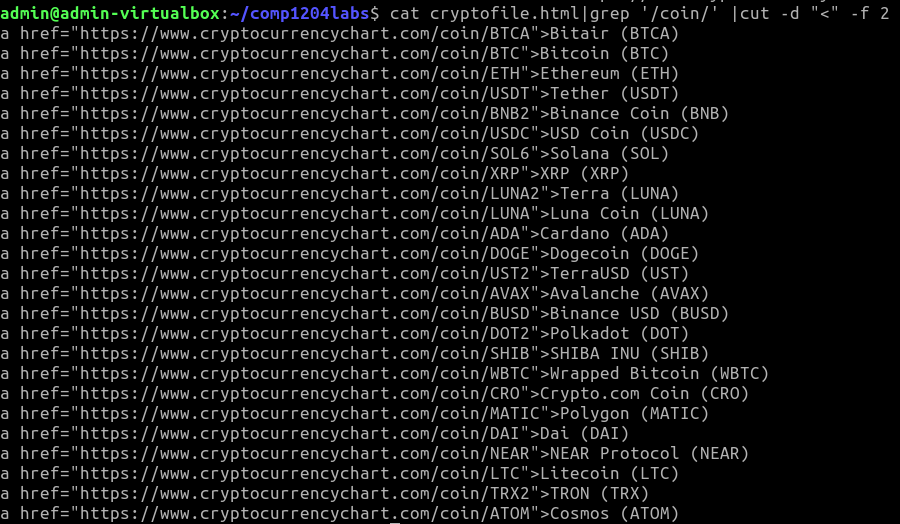
:



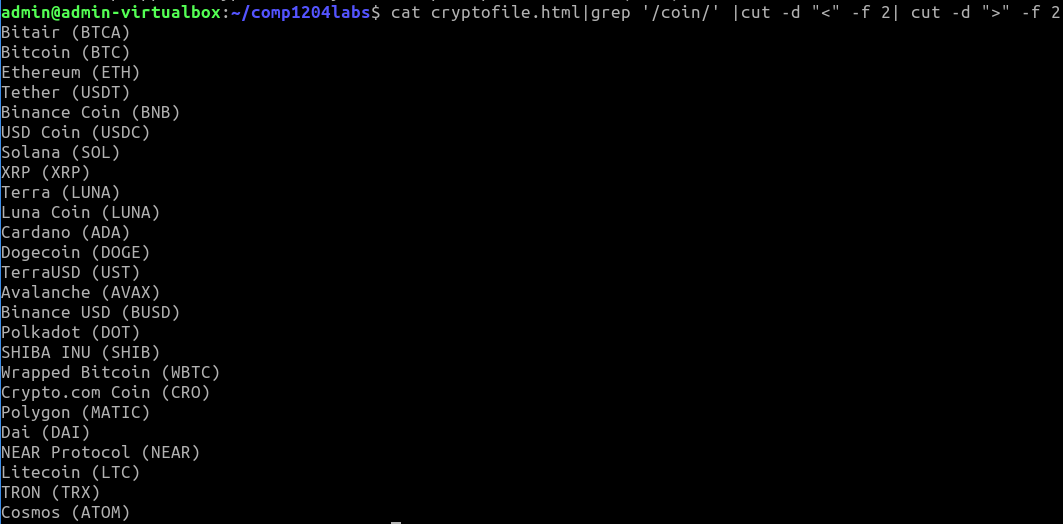
* Pipe output to grep line with expression “/coin/”



* Pipe output to cut between 1st and 2nd occurrence of delimiter <



* Pipe output to cut between 1st and 2nd occurrence of delimiter >



* Pipe output to search for “ (“ and replace with “-(“ globally



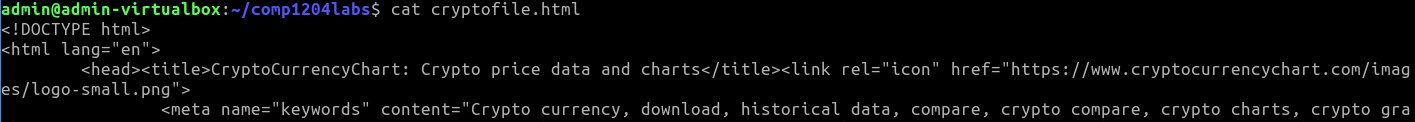
* Pipe output to cut up to 1st occurrence of delimiter (



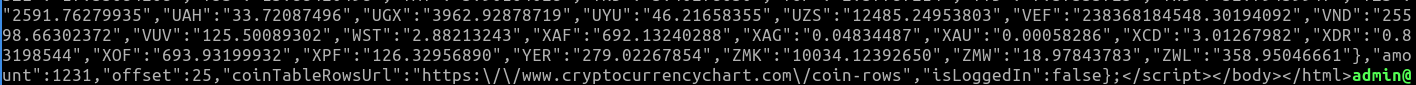
Clean up html file to get crypto coin live price and assign to variable live



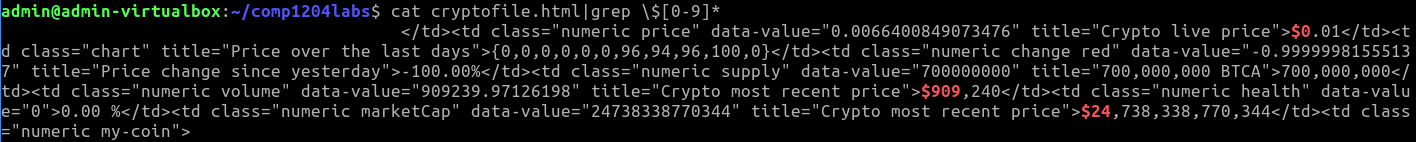
* Read html file



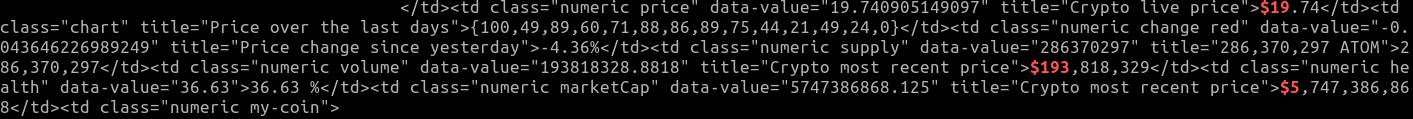
:



* Pipe output to grep line with expression with character $ followed by zero or more numbers in range 0 to 9



:



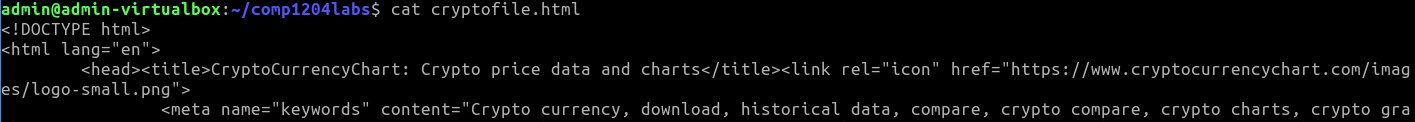
* Pipe output to cut between 3rd and 4th occurrence of delimiter “



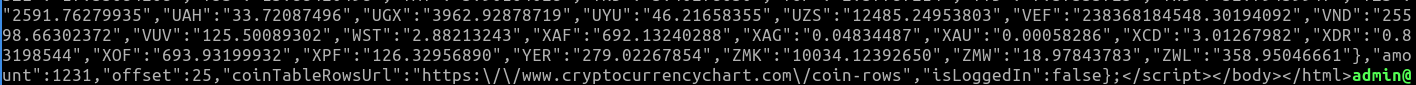
Clean up html file to get crypto coin live price and assign to variable change



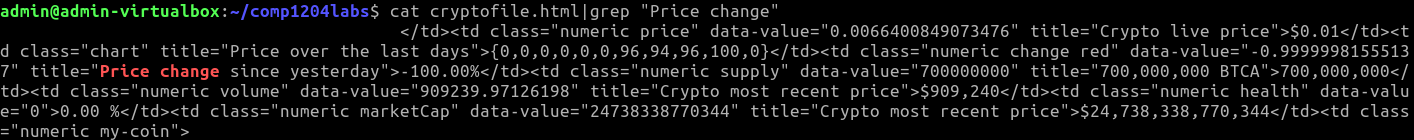
* Read html file



:



* Pipe output to grep line with expression “Price change”



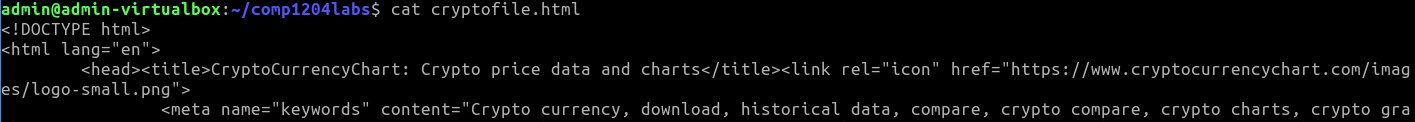
* Pipe output to cut between 13th and 14th occurrence of delimiter “



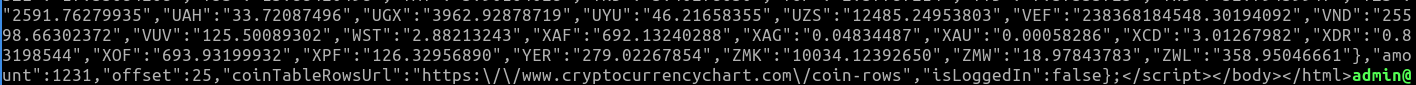
Clean up html file to get crypto coin supply and assign to variable supply



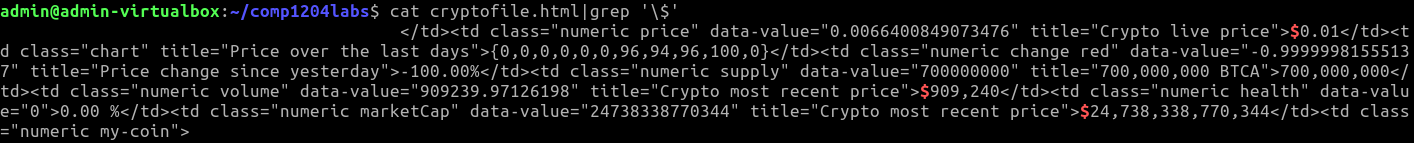
* Read html file



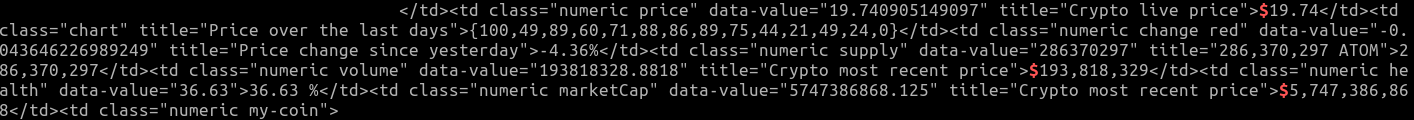
:



* Pipe output to grep line with expression “$”



:



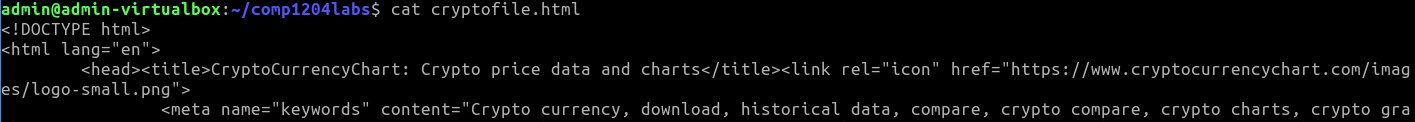
* Pipe output to cut between 19th and 20th occurrence of delimiter “



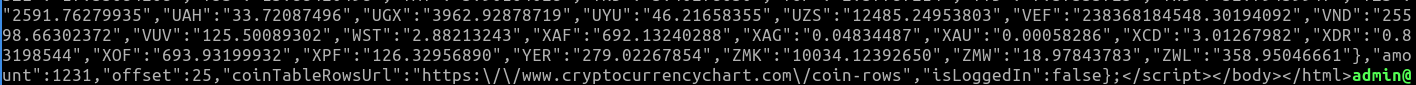
Clean up html file to get crypto coin market capitalization and assign to variable marketCap



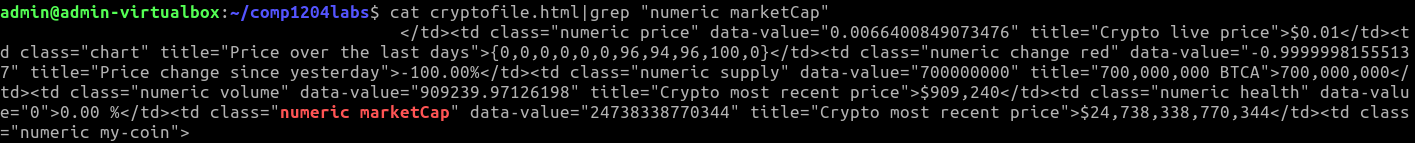
* Read html file



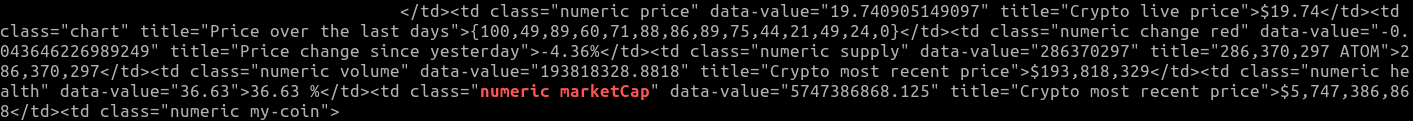
:



* Pipe output to grep line with expression “numeric marketCap”



:



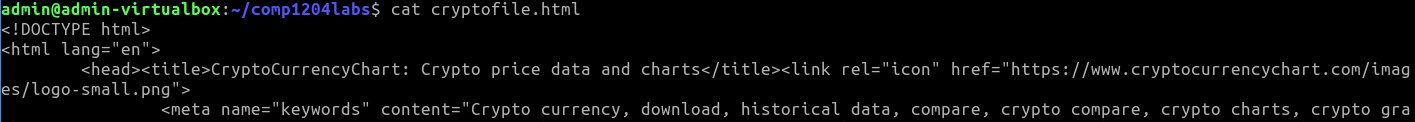
* Pipe output to cut between 35th and 36th occurrence of delimiter “



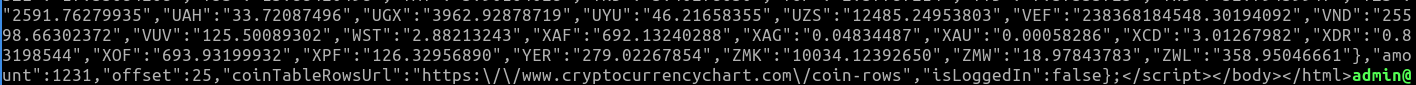
Clean up html file to get crypto coin trade volume and assign to variable volume



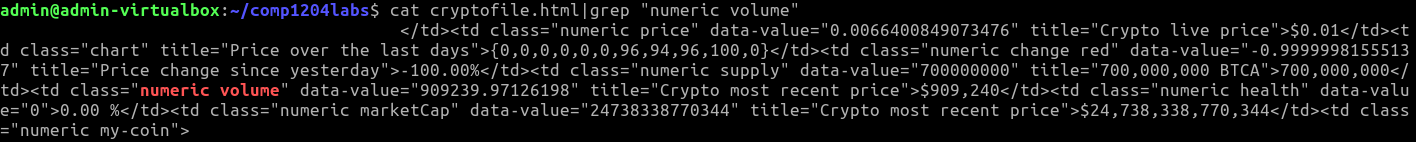
* Read html file



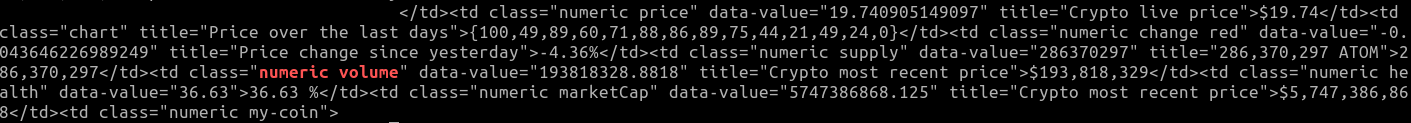
:



* Pipe output to grep line with expression “numeric volume”



:



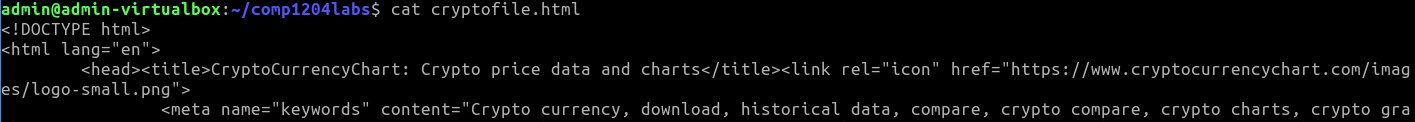
* Pipe output to cut between 25th and 26th occurrence of delimiter “



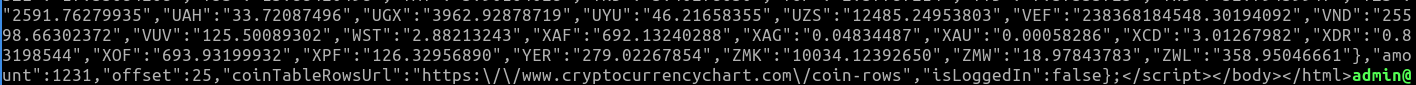
Clean up html file to get crypto coin trade activity and assign to variable activity



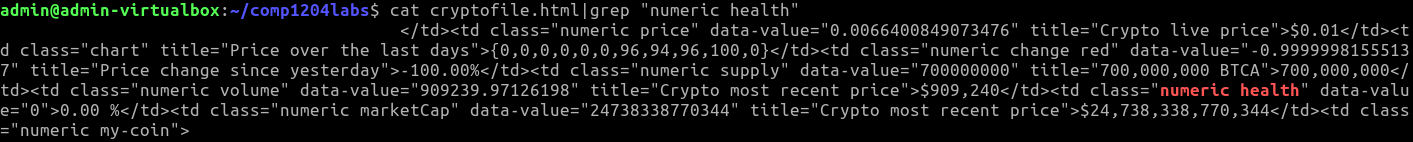
* Read html file



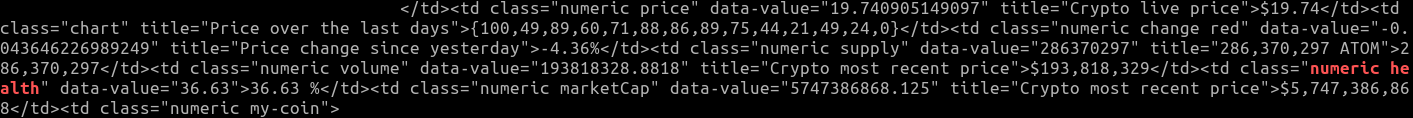
:



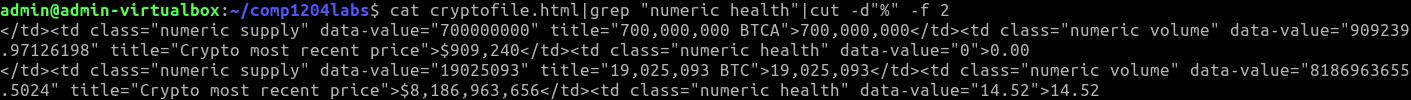
* Pipe output to grep line with expression “numeric health”



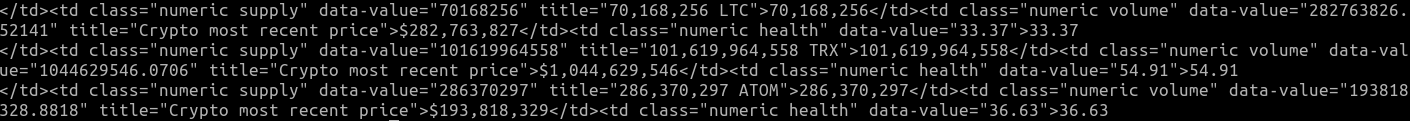
:



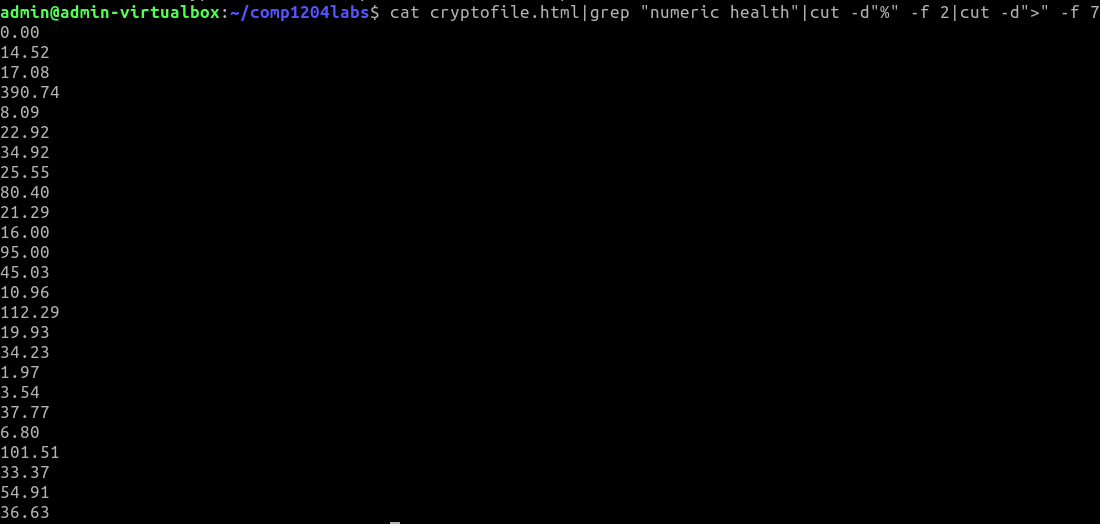
* Pipe output to cut between 1st and 2nd occurrence of delimiter %



:



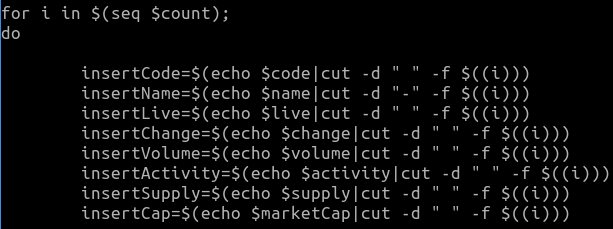
* Pipe output to cut between 6th and 7th occurrence of delimiter >



Clean up html file to get number of lines in crypto coin code and assign to variable count



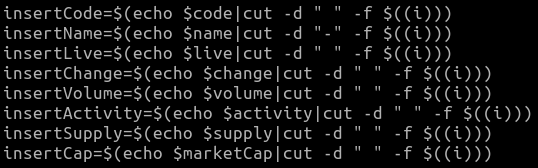
Loop through all variables in data cleaned and stored in variables to extract single data for inserting into database



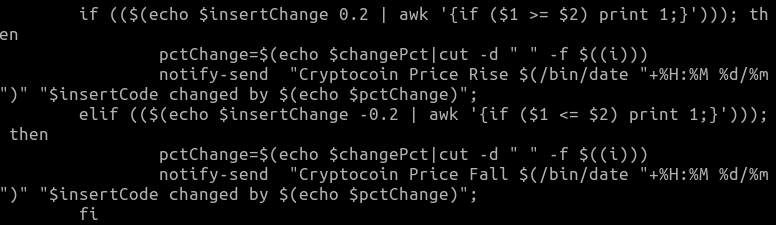
* Loop i from sequence generated from 1 up to value stored in variable count



* Echo the value in each variable, pipe output to cut between (i-1)th and ith occurrence of delimiter space or -, to get individual data for each attribute and store in respective variables

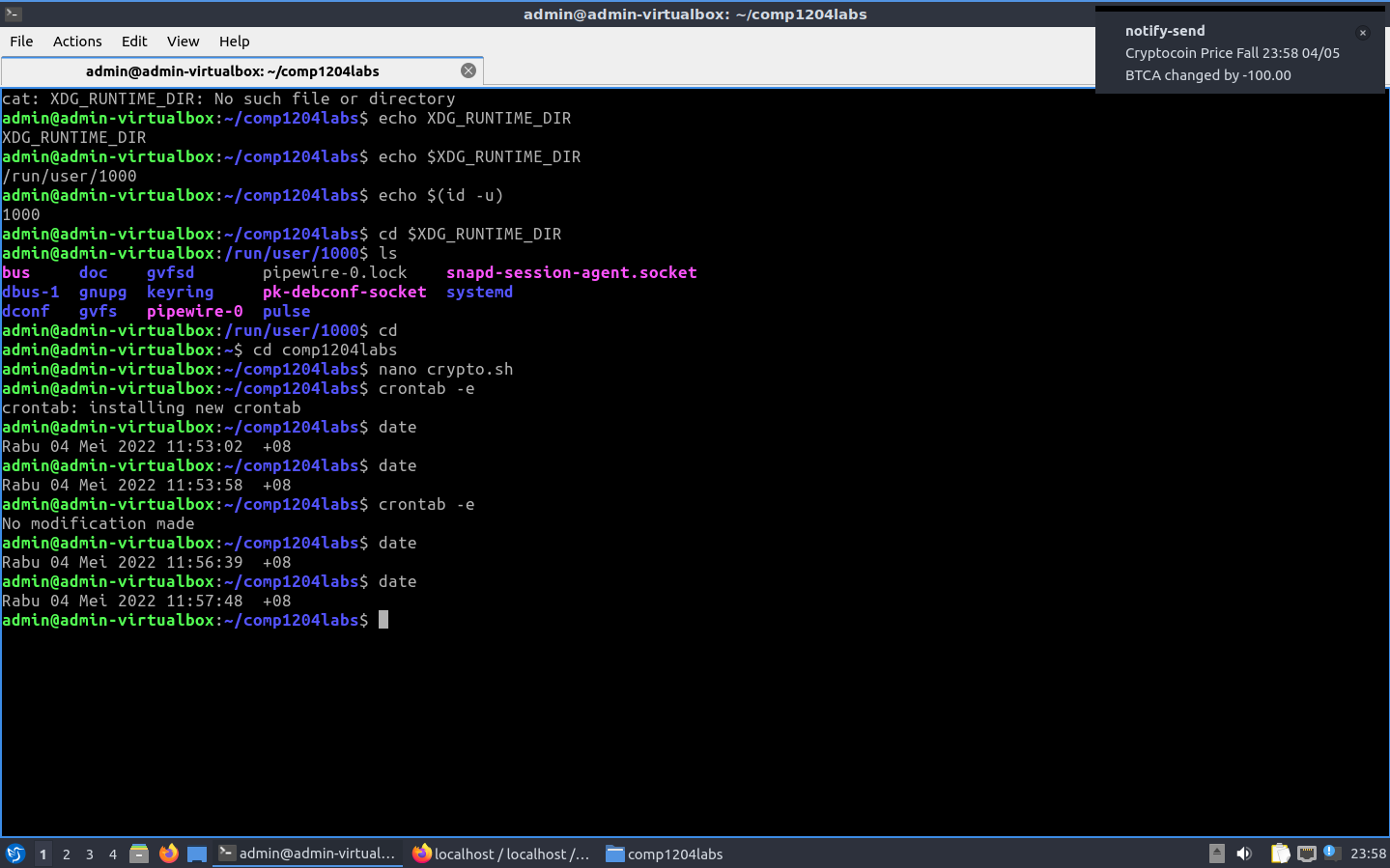


Compare value of change with 0.2 or -0.2 using awk to compare float data, and print 1 for true if change is greater than or equals to 0.2 or less than or equals to -0.2. Extract change in percentage form, notify-send to send notification with summary including datetime and body consisting of the code of crypto coin and percentage change



Output:





# Crontab Setup

Edit current crontab



Edit crontab file to set process at specified absolute path to run every 4 hours (must end with newline)



To run crontab with notify-send displayed, set XDG\_RUNTIME\_DIR to /run/user/1000 where XDG\_RUNTIME\_DIR is the environment variable of user-specific directory (referencing <https://stackoverflow.com/a/53598510>)

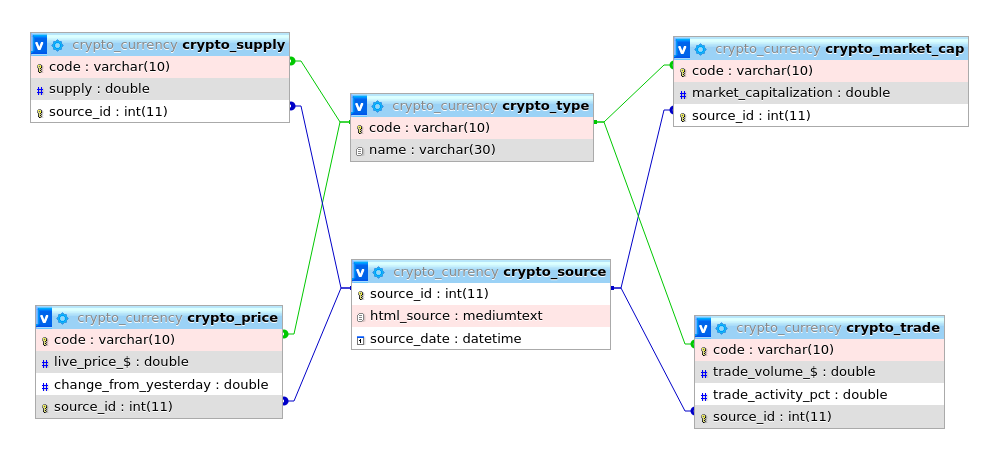


New crontab automatically installed when save and exit



# MySQL for Data Storage

## Database Structure



1

1

1

1

1

1

1

1

1..\*

1..\*

1..\*

1..\*

0..\*

0..\*

0..\*

0..\*

## Creating database and tables

Create database ‘crypto\_currency’ if it does not exist

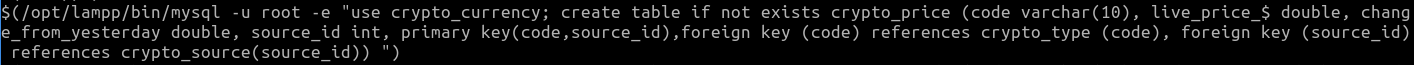


Create table ‘crypto\_source’ if it does not exist, with attributes auto incremented ‘source\_id’ as primary key, ‘html\_source’ of type mediumtext and ‘source\_date’ of type datetime

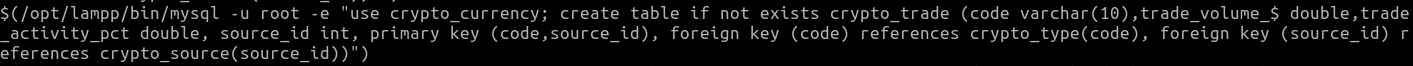
Create table ‘crypto\_type’ if it does not exist, with attributes ‘code’ of type varchar(10) as primary key and ‘name’ of type varchar(30)



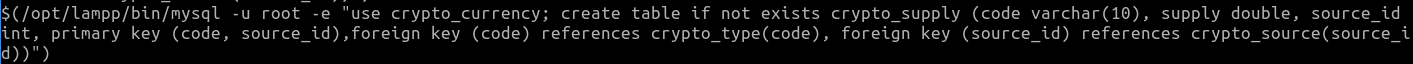
Create table ‘crypto\_price’ if it does not exist, with attributes ‘code’ of type varchar(10) which is a foreign key referencing ‘code’ in table ‘crypto\_type’, ‘live\_price\_$’ of type double, ‘change\_from\_yesterday’ of type double and ‘source\_id’ of type int which is a foreign key referencing ‘source\_id’ in table ‘crypto\_source’, where ‘code’ and ’source\_id’ make up the composite primary key



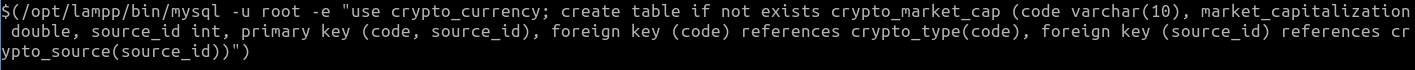
Create table ‘crypto\_trade’ if it does not exist, with attributes ‘code’ of type varchar(10) which is a foreign key referencing ‘code’ in table ‘crypto\_type’, ‘trade\_volume\_$’ of type double, ‘trade\_activity\_pct’ of type double and ‘source\_id’ of type int which is a foreign key referencing ‘source\_id’ in table ‘crypto\_source’, where ‘code’ and ’source\_id’ make up the composite primary key



Create table ‘crypto\_supply’ if it does not exist, with attributes ‘code’ of type varchar(10) which is a foreign key referencing ‘code’ in table ‘crypto\_type’, ‘supply’ of type double and ‘source\_id’ of type int which is a foreign key referencing ‘source\_id’ in table ‘crypto\_source’, where ‘code’ and ’source\_id’ make up the composite primary key



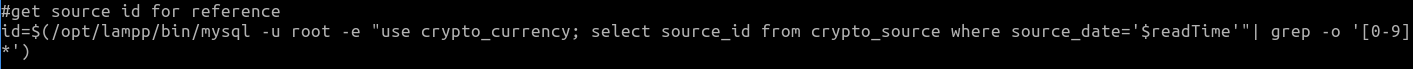
Create table ‘crypto\_market\_cap’ if it does not exist, with attributes ‘code’ of type varchar(10) which is a foreign key referencing ‘code’ in table ‘crypto\_type’, ‘market\_capitalization’ of type double and ‘source\_id’ of type int which is a foreign key referencing ‘source\_id’ in table ‘crypto\_source’, where ‘code’ and ’source\_id’ make up the composite primary key



## Inserting data into database

Insert value stored in variable code into column ‘html\_source’, value stored in variable readTime into column ‘source\_date’ into table ‘crypto\_source’

Select source\_id from table ‘crypto\_source’ where the ‘source\_date’ = value of date time stored in variable readTime, pipe result of query to grep only expression with 0 to many integers in range 0 to 9, store value in variable id



Insert value stored in variable insertCode into column ‘code’, value stored in variable insertName into column ‘name’ into table ‘crypto\_type’



Insert value stored in variable insertCode into column ‘code’, value stored in variable insertLive into column ‘live\_price\_$’, value stored in variable insertChange into column ‘change\_from\_yesterday’, value stored in variable id into column ‘source\_id’ into table ‘crypto\_price’



Insert value stored in variable insertCode into column ‘code’, value stored in variable insertVolume into column ‘trade\_volume\_$’, value stored in variable insertActivity into column ‘trade\_activity\_pct’, value stored in variable id into column ‘source\_id’ into table ‘crypto\_trade’



Insert value stored in variable insertCode into column ‘code’, value stored in variable insertSupply into column ‘supply’, value stored in variable id into column ‘source\_id’ into table ‘crypto\_supply’

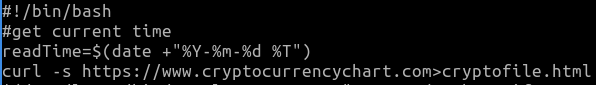


Insert value stored in variable insertCode into column ‘code’, value stored in variable insertCap into column ‘market\_capitalization’, value stored in variable id into column ‘source\_id’ into table ‘crypto\_market\_cap’



# Script for Plotting

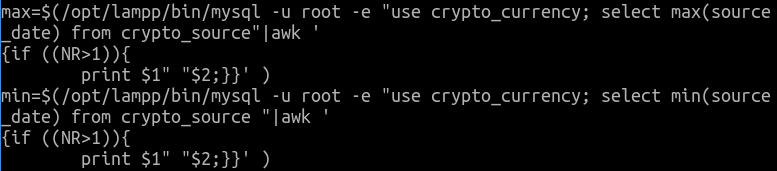
Specify path of bash interpreter



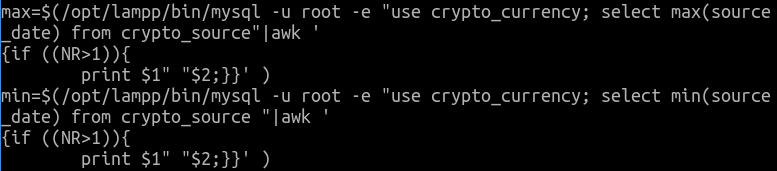
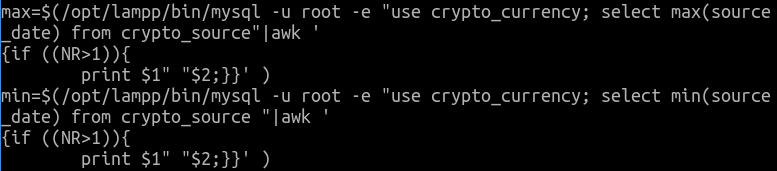
Create directory type to store txt files for type, directory price to store dat files for price



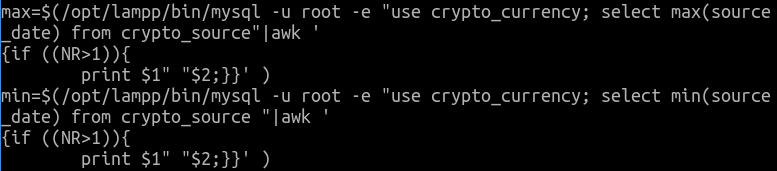
Get maximum and minimum datetime and manipulate results to be used as range for the x-axis



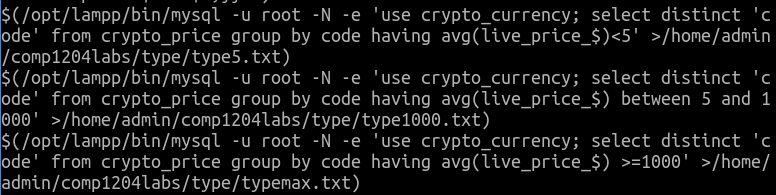
* Select max/min ‘source\_date’ from table ‘crypto\_source’



* Pipe output to awk and print first and second parameter if record number is greater than 1, storing the value into variable max/min



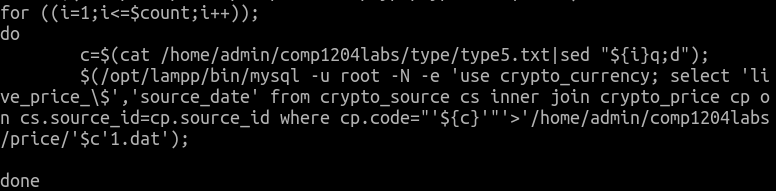
Select ‘code’ from table crypto\_price which when grouped by ‘code’ having average ‘live\_price\_$’ less than 5/ between 5 to 1000/ greater than 1000, write results to three different files in directory type



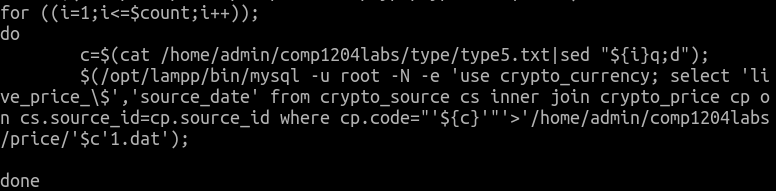
Count the number of lines in file type5.txt, storing the value in variable count



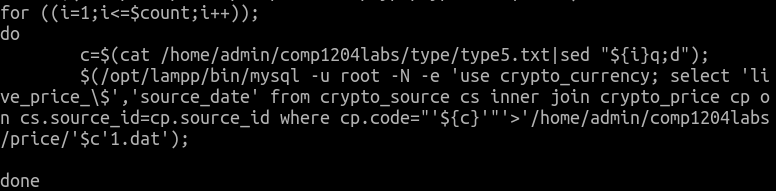
Loop through each line in file



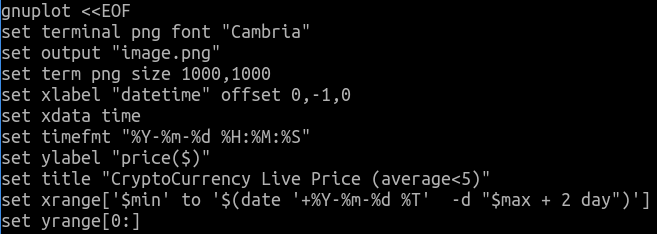
Read file type5.txt and pipe output to sed to get text at line i, storing the value into variable c



Select without column header ‘live\_price\_$’ and ‘source\_date’ from table ‘crypto\_source’ inner join table ‘crypto\_price’ on ‘source\_id’ where code = value of variable c, write result to dat file named according to the code of the cryptocoin in directory price



Set up graph for plotting using gnuplot including details of the output file, label and offset position of x-axis, label of y-axis, specify data type on x-axis as time with format, title of graph, range of x-axis from min to 2 days ahead of max, range of y-axis from 0



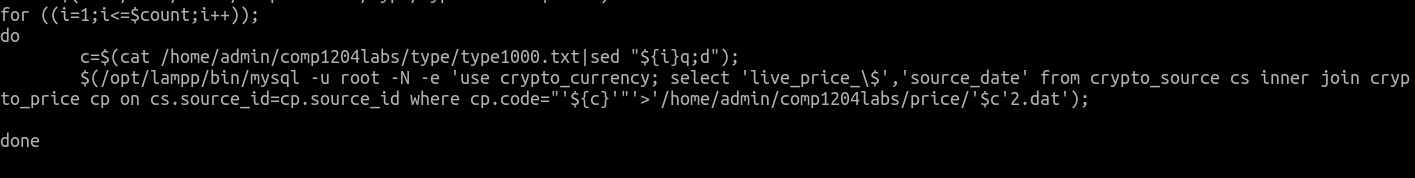
Loop through all files found in directory price with name ending with ‘1.dat’, plot data in column 2 against data in column 1 with line points and substring of the name of file to get code of cryptocurrency as title for each curve plotted



Count the number of lines in file type1000.txt, storing the value in variable count



Loop through each line in file



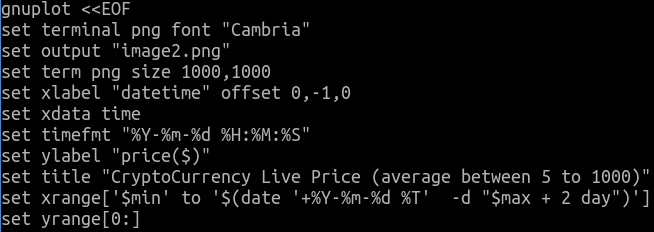
Read file type1000.txt and pipe output to sed to get text at line i, storing the value into variable c



Select without column header ‘live\_price\_$’ and ‘source\_date’ from table ‘crypto\_source’ inner join table ‘crypto\_price’ on ‘source\_id’ where code = value of variable c, write result to dat file named according to the code of the cryptocoin in directory price



Set up graph for plotting using gnuplot including details of the output file, label and offset position of x-axis, label of y-axis, specify data type on x-axis as time with format, title of graph, range of x-axis from min to 2 days ahead of max, range of y-axis from 0



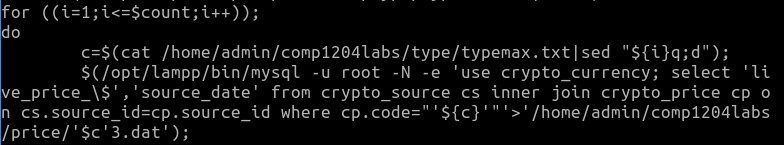
Loop through all files found in directory price with name ending with ‘2.dat’, plot data in column 2 against data in column 1 with line points and substring of the name of file to get code of cryptocurrency as title for each curve plotted



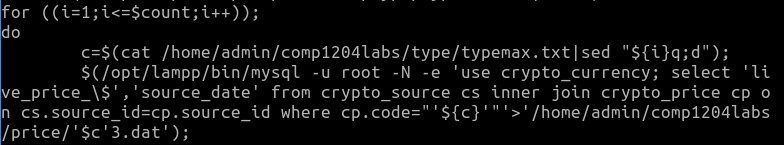
Count the number of lines in file typemax.txt, storing the value in variable count



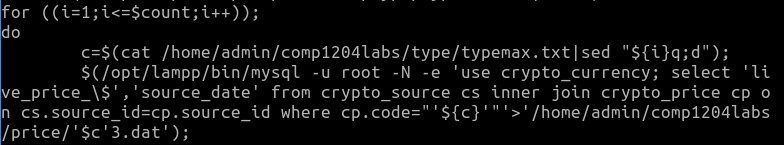
Loop through each line in file



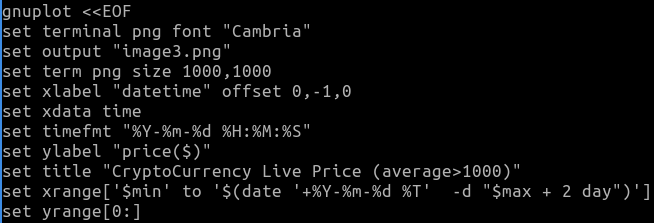
Read file type1000.txt and pipe output to sed to get text at line i, storing the value into variable c



Select without column header ‘live\_price\_$’ and ‘source\_date’ from table ‘crypto\_source’ inner join table ‘crypto\_price’ on ‘source\_id’ where code = value of variable c, write result to dat file named according to the code of the cryptocoin in directory price



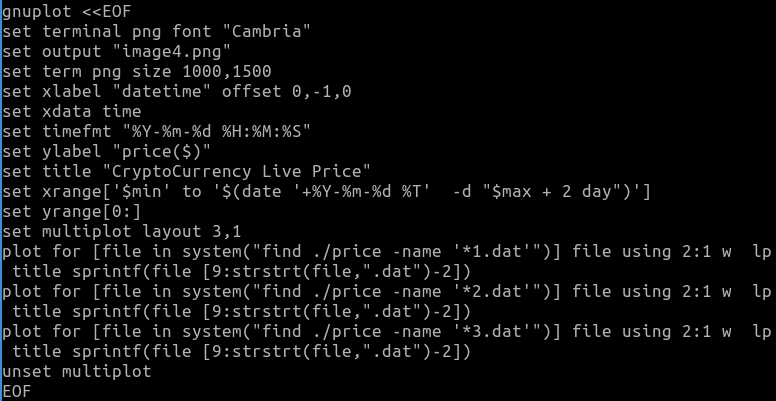
Set up graph for plotting using gnuplot including details of the output file, label and offset position of x-axis, label of y-axis, specify data type on x-axis as time with format, title of graph, range of x-axis from min to 2 days ahead of max, range of y-axis from 0



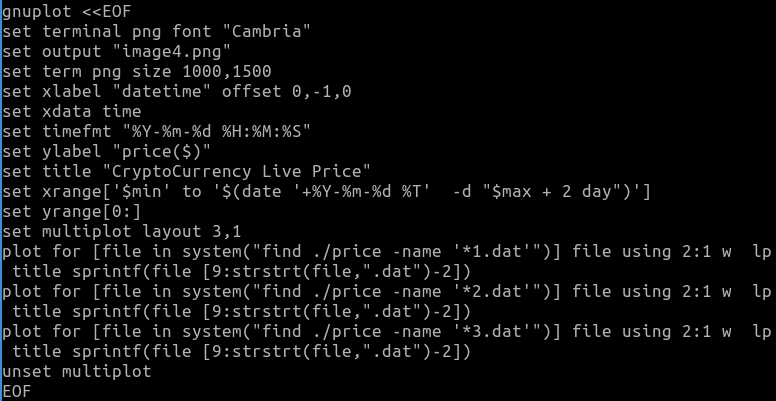
Loop through all files found in directory price with name ending with ‘3.dat’, plot data in column 2 against data in column 1 with line points and substring of the name of file to get code of cryptocurrency as title for each curve plotted



Set up graph for plotting using gnuplot including details of the output file, label and offset position of x-axis, label of y-axis, specify data type on x-axis as time with format, title of graph, range of x-axis from min to 2 days ahead of max, range of y-axis from 0



Set multiplot to plot multiple graphs on one output file, arranged as 3 graphs in each row and 1 in each column



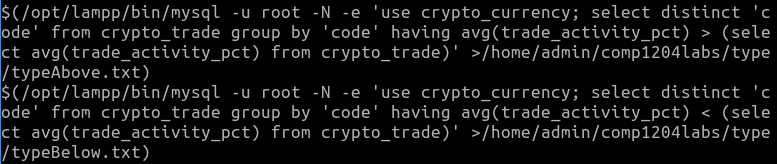
Remove directory price recursively



Create directory trade to store dat files for trade activity



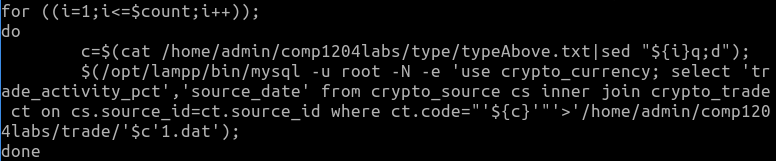
Select ‘code’ from table crypto\_trade which when grouped by ‘code’ having average ‘trade\_activity\_pct’ greater than / less than average ‘trade\_activity\_pct’ of all data in table ‘crypto\_trade’, write results to two different files in directory type



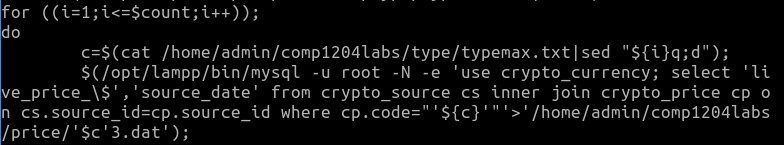
Count the number of lines in file typeAbove.txt, storing the value in variable count



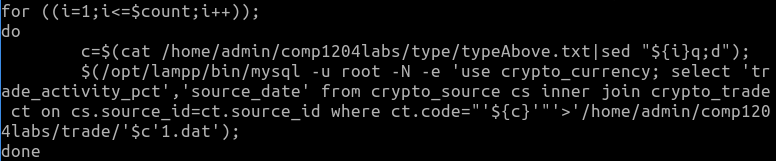
Loop through each line in file



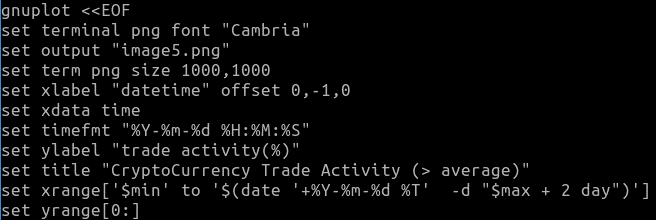
Read file typeAbove.txt and pipe output to sed to get text at line i, storing the value into variable c



Select without column header ‘trade\_activity\_pct’ and ‘source\_date’ from table ‘crypto\_source’ inner join table ‘crypto\_trade’ on ‘source\_id’ where code = value of variable c, write result to dat file named according to the code of the cryptocoin in directory trade



Set up graph for plotting using gnuplot including details of the output file, label and offset position of x-axis, label of y-axis, specify data type on x-axis as time with format, title of graph, range of x-axis from min to 2 days ahead of max, range of y-axis from 0



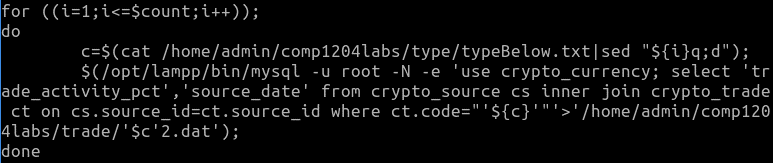
Loop through all files found in directory trade with name ending with ‘1.dat’, plot data in column 2 against data in column 1 with line points and substring of the name of file to get code of cryptocurrency as title for each curve plotted



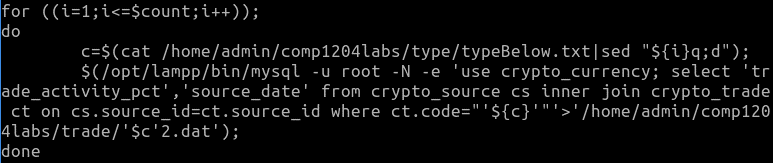
Count the number of lines in file typeBelow.txt, storing the value in variable count



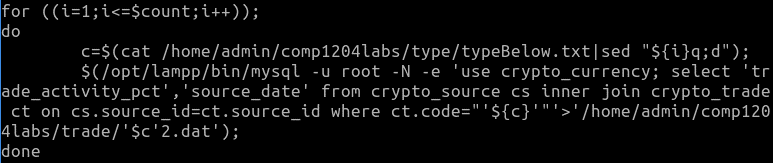
Loop through each line in file



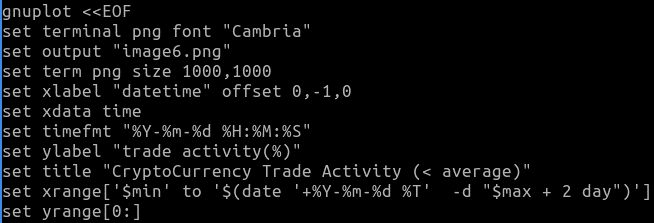
Read file typeBelow.txt and pipe output to sed to get text at line i, storing the value into variable c



Select without column header ‘trade\_activity\_pct’ and ‘source\_date’ from table ‘crypto\_source’ inner join table ‘crypto\_trade’ on ‘source\_id’ where code = value of variable c, write result to dat file named according to the code of the cryptocoin in directory trade



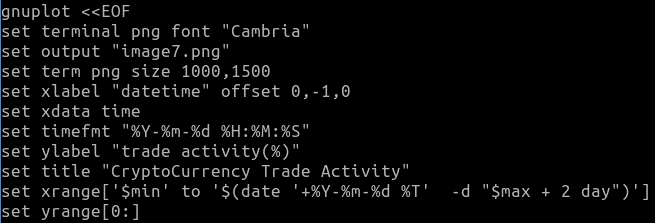
Set up graph for plotting using gnuplot including details of the output file, label and offset position of x-axis, label of y-axis, specify data type on x-axis as time with format, title of graph, range of x-axis from min to 2 days ahead of max, range of y-axis from 0



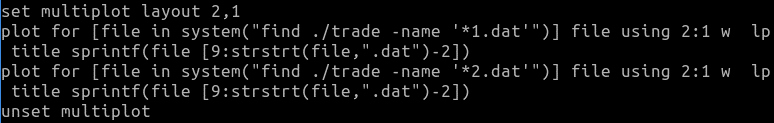
Loop through all files found in directory trade with name ending with ‘2.dat’, plot data in column 2 against data in column 1 with line points and substring of the name of file to get code of cryptocurrency as title for each curve plotted



Set up graph for plotting using gnuplot including details of the output file, label and offset position of x-axis, label of y-axis, specify data type on x-axis as time with format, title of graph, range of x-axis from min to 2 days ahead of max, range of y-axis from 0



Set multiplot to plot multiple graphs on one output file, arranged as 2 graphs in each row and 1 in each column



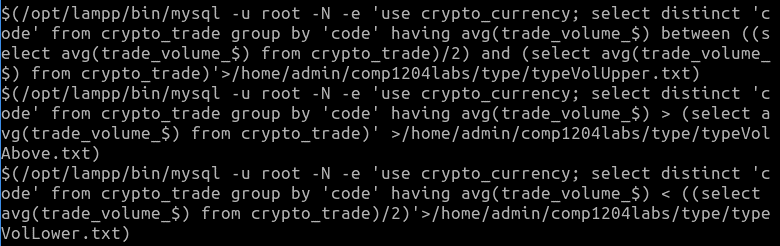
Remove directory trade recursively



Create directory tradeVol to store dat files for trade volume



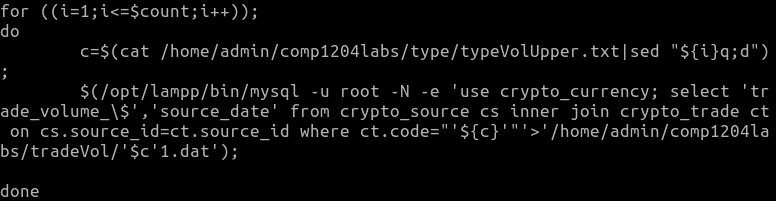
Select ‘code’ from table crypto\_trade which when grouped by ‘code’ having average ‘trade\_volume\_$’ between half of average ‘trade\_volume\_$’ of all data in table ‘crypto\_trade’ and average ‘trade\_volume\_$’ of all data in table ‘crypto\_trade’ / greater than average ‘trade\_volume\_$’ of all data in table ‘crypto\_trade’ / less than half of average ‘trade\_volume\_$’ of all data in table ‘crypto\_trade’, write results to three different files in directory type



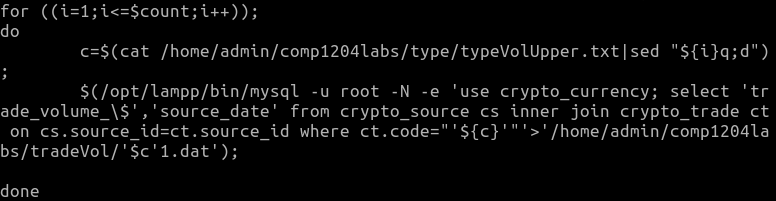
Count the number of lines in file typeVolUpper.txt, storing the value in variable count



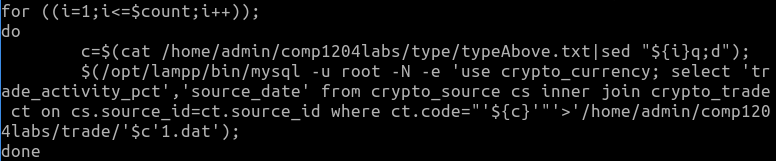
Loop through each line in file



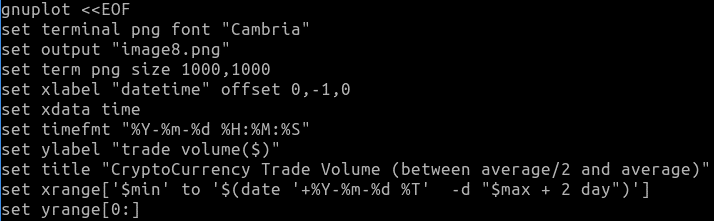
Read file typeVolUpper.txt and pipe output to sed to get text at line i, storing the value into variable c



Select without column header ‘trade\_volume\_$’ and ‘source\_date’ from table ‘crypto\_source’ inner join table ‘crypto\_trade’ on ‘source\_id’ where code = value of variable c, write result to dat file named according to the code of the cryptocoin in directory tradeVol



Set up graph for plotting using gnuplot including details of the output file, label and offset position of x-axis, label of y-axis, specify data type on x-axis as time with format, title of graph, range of x-axis from min to 2 days ahead of max, range of y-axis from 0



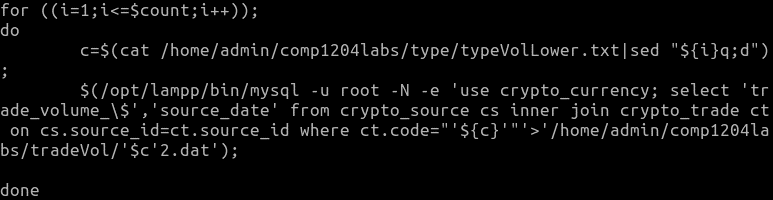
Loop through all files found in directory tradeVol with name ending with ‘1.dat’, plot data in column 2 against data in column 1 with line points and substring of the name of file to get code of cryptocurrency as title for each curve plotted



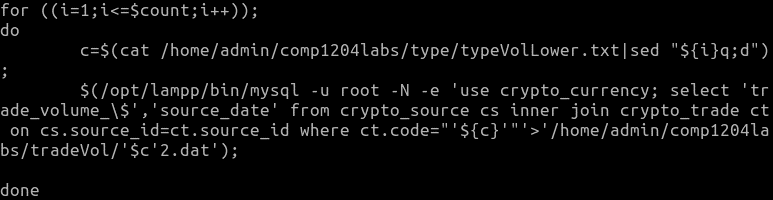
Count the number of lines in file typeVolLower.txt, storing the value in variable count



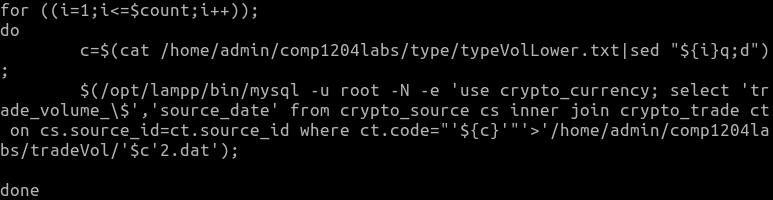
Loop through each line in file



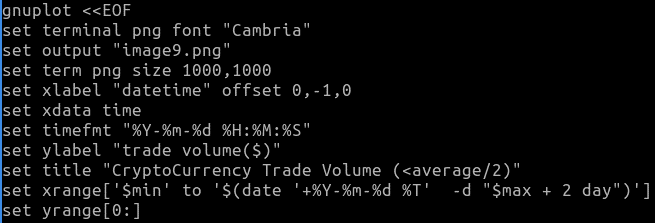
Read file typeVolLower.txt and pipe output to sed to get text at line i, storing the value into variable c



Select without column header ‘trade\_volume\_$’ and ‘source\_date’ from table ‘crypto\_source’ inner join table ‘crypto\_trade’ on ‘source\_id’ where code = value of variable c, write result to dat file named according to the code of the cryptocoin in directory tradeVol



Set up graph for plotting using gnuplot including details of the output file, label and offset position of x-axis, label of y-axis, specify data type on x-axis as time with format, title of graph, range of x-axis from min to 2 days ahead of max, range of y-axis from 0



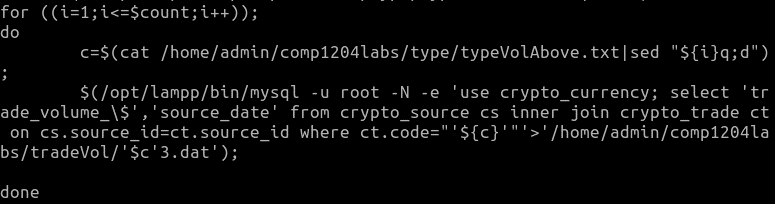
Loop through all files found in directory tradeVol with name ending with ‘2.dat’, plot data in column 2 against data in column 1 with line points and substring of the name of file to get code of cryptocurrency as title for each curve plotted



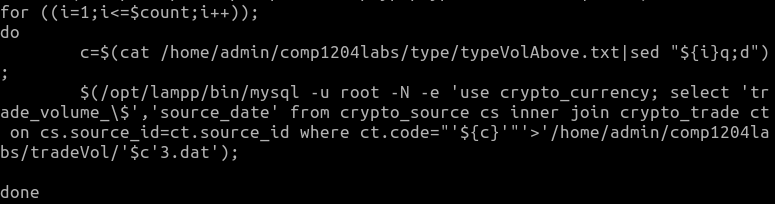
Count the number of lines in file typeVolAbove.txt, storing the value in variable count



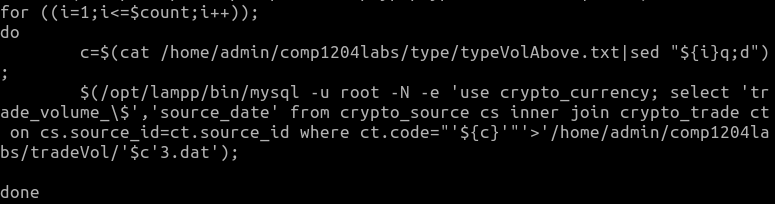
Loop through each line in file



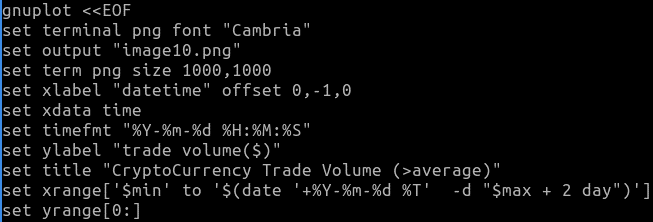
Read file typeVolAbove.txt and pipe output to sed to get text at line i, storing the value into variable c



Select without column header ‘trade\_volume\_$’ and ‘source\_date’ from table ‘crypto\_source’ inner join table ‘crypto\_trade’ on ‘source\_id’ where code = value of variable c, write result to dat file named according to the code of the cryptocoin in directory tradeVol



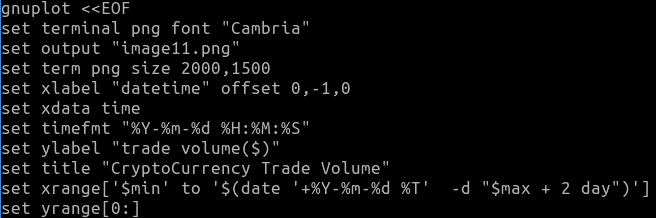
Set up graph for plotting using gnuplot including details of the output file, label and offset position of x-axis, label of y-axis, specify data type on x-axis as time with format, title of graph, range of x-axis from min to 2 days ahead of max, range of y-axis from 0



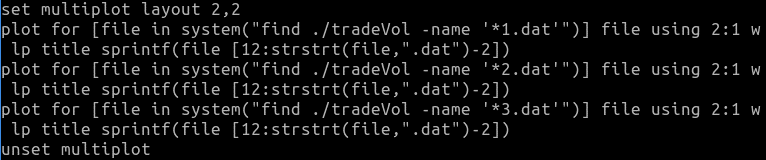
Loop through all files found in directory tradeVol with name ending with ‘3.dat’, plot data in column 2 against data in column 1 with line points and substring of the name of file to get code of cryptocurrency as title for each curve plotted



Set up graph for plotting using gnuplot including details of the output file, label and offset position of x-axis, label of y-axis, specify data type on x-axis as time with format, title of graph, range of x-axis from min to 2 days ahead of max, range of y-axis from 0



Set multiplot to plot multiple graphs on one output file, arranged as 2 graphs in each row and 2 graphs in each column



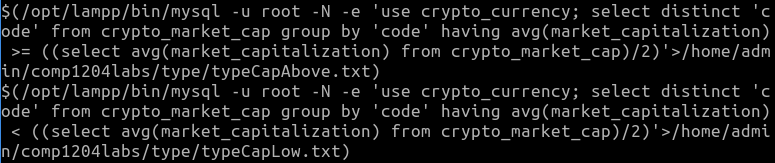
Remove directory tradeVol recursively



Create directory marketCap to store dat files for market capitalization



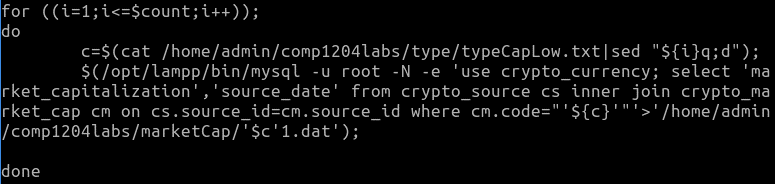
Select ‘code’ from table crypto\_market\_cap which when grouped by ‘code’ having average ‘market\_capitalization’ greater than or equals to half of average ‘market\_capitalization’ of all data in table ‘crypto\_market\_cap’ and average ‘market\_capitalization’ of all data in table ‘crypto\_market\_cap’ / less than half of average ‘market\_capitalization’ of all data in table ‘crypto\_market\_cap’, write results to two different files in directory type



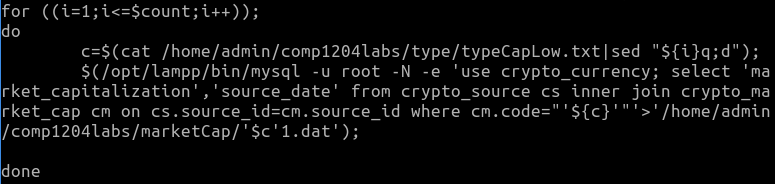
Count the number of lines in file typeCapLow.txt, storing the value in variable count



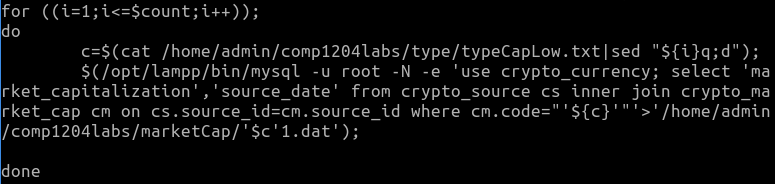
Loop through each line in file



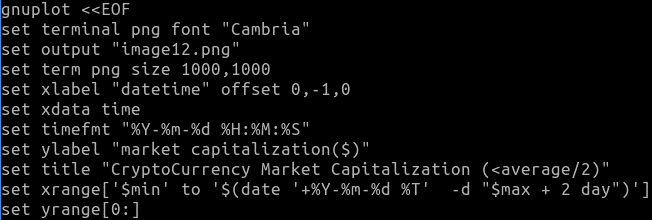
Read file typeCapLow.txt and pipe output to sed to get text at line i, storing the value into variable c



Select without column header ‘market\_capitalization’ and ‘source\_date’ from table ‘crypto\_source’ inner join table ‘crypto\_market\_cap’ on ‘source\_id’ where code = value of variable c, write result to dat file named according to the code of the cryptocoin in directory marketCap



Set up graph for plotting using gnuplot including details of the output file, label and offset position of x-axis, label of y-axis, specify data type on x-axis as time with format, title of graph, range of x-axis from min to 2 days ahead of max, range of y-axis from 0



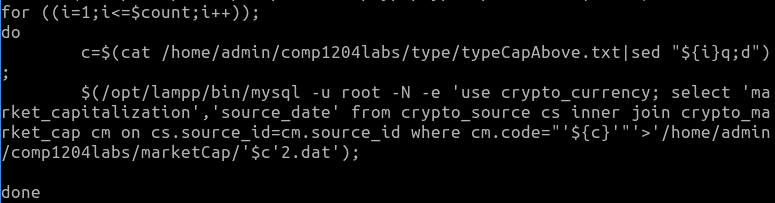
Loop through all files found in directory marketCap with name ending with ‘1.dat’, plot data in column 2 against data in column 1 with line points and substring of the name of file to get code of cryptocurrency as title for each curve plotted



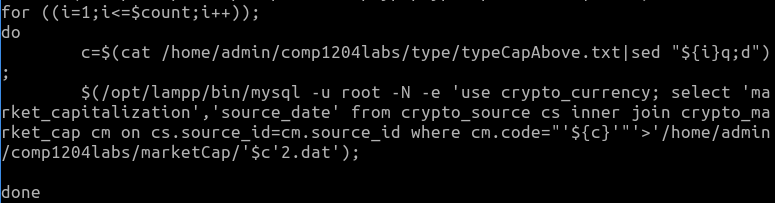
Count the number of lines in file typeCapAbove.txt, storing the value in variable count



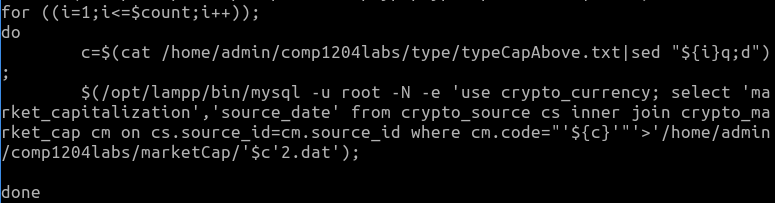
Loop through each line in file



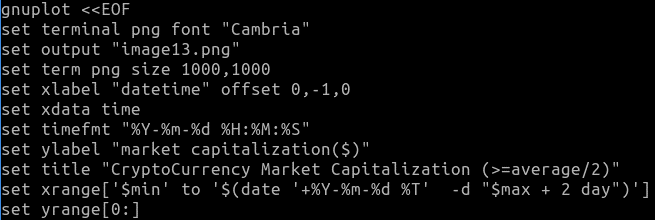
Read file typeCapAbove.txt and pipe output to sed to get text at line i, storing the value into variable c



Select without column header ‘market\_capitalization’ and ‘source\_date’ from table ‘crypto\_source’ inner join table ‘crypto\_market\_cap’ on ‘source\_id’ where code = value of variable c, write result to dat file named according to the code of the cryptocoin in directory marketCap



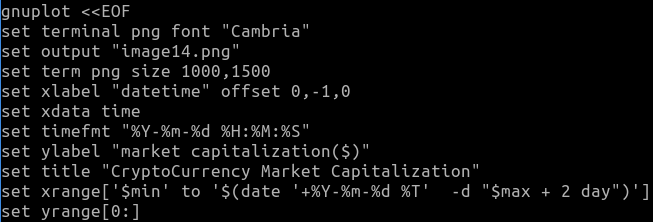
Set up graph for plotting using gnuplot including details of the output file, label and offset position of x-axis, label of y-axis, specify data type on x-axis as time with format, title of graph, range of x-axis from min to 2 days ahead of max, range of y-axis from 0



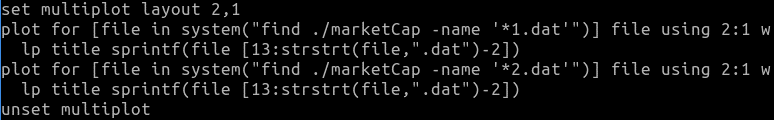
Loop through all files found in directory marketCap with name ending with ‘2.dat’, plot data in column 2 against data in column 1 with line points and substring of the name of file to get code of cryptocurrency as title for each curve plotted



Set up graph for plotting using gnuplot including details of the output file, label and offset position of x-axis, label of y-axis, specify data type on x-axis as time with format, title of graph, range of x-axis from min to 2 days ahead of max, range of y-axis from 0



Set multiplot to plot multiple graphs on one output file, arranged as 2 graphs in each row and 1 graph in each column



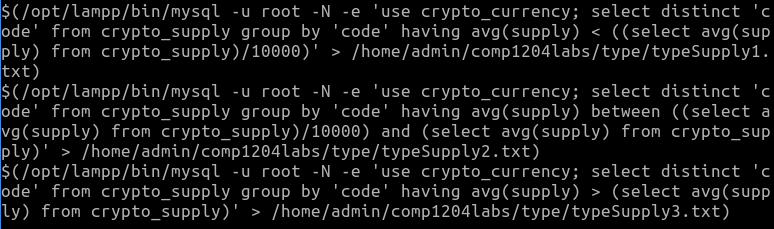
Remove directory marketCap recursively



Create directory supply to store dat files for cryptocoin supply



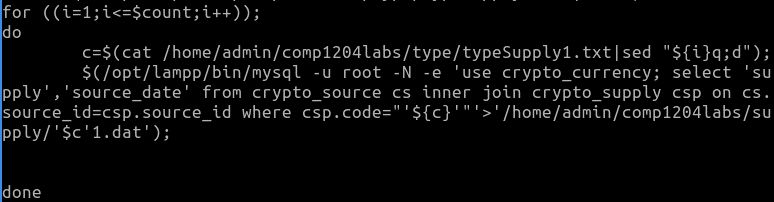
Select ‘code’ from table crypto\_supply which when grouped by ‘code’ having average ‘supply’ less than average ‘supply’ of all data in table ‘crypto\_supply’ divided by 10000 / between average ‘supply’ of all data in table ‘crypto\_supply’ divided by 10000 and average ‘supply’ of all data in table ‘crypto\_supply’ / greater than average ‘supply’ of all data in table ‘crypto\_supply’, write results to three different files in directory type



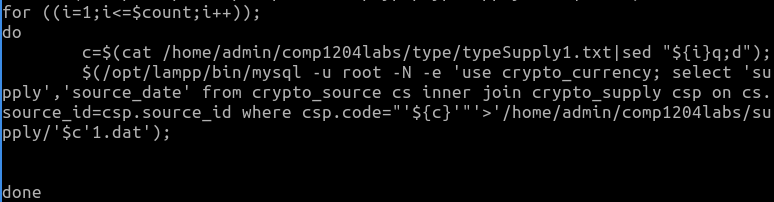
Count the number of lines in file typeSupply1.txt, storing the value in variable count



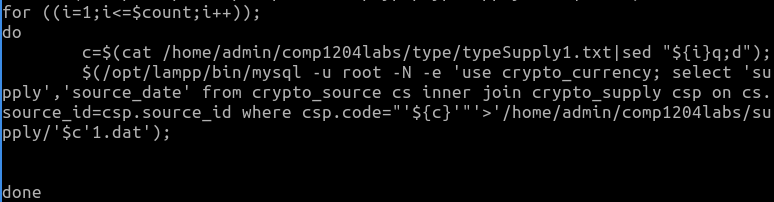
Loop through each line in file



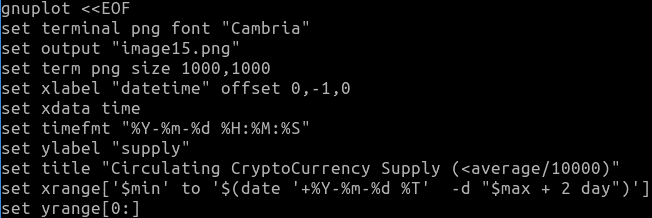
Read file typeSupply1.txt and pipe output to sed to get text at line i, storing the value into variable c



Select without column header ‘supply’ and ‘source\_date’ from table ‘crypto\_source’ inner join table ‘crypto\_supply’ on ‘source\_id’ where code = value of variable c, write result to dat file named according to the code of the cryptocoin in directory supply



Set up graph for plotting using gnuplot including details of the output file, label and offset position of x-axis, label of y-axis, specify data type on x-axis as time with format, title of graph, range of x-axis from min to 2 days ahead of max, range of y-axis from 0



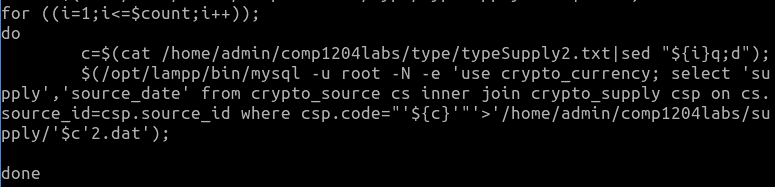
Loop through all files found in directory supply with name ending with ‘1.dat’, plot data in column 2 against data in column 1 with line points and substring of the name of file to get code of cryptocurrency as title for each curve plotted



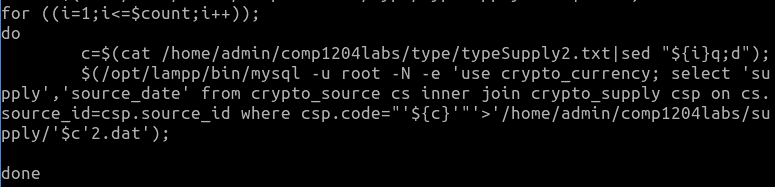
Count the number of lines in file typeSupply2.txt, storing the value in variable count



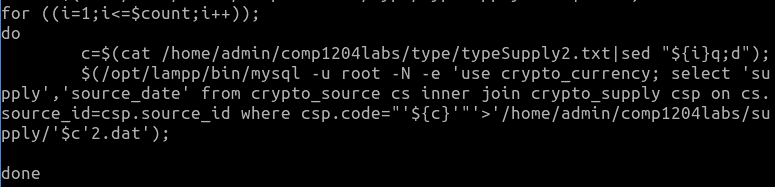
Loop through each line in file



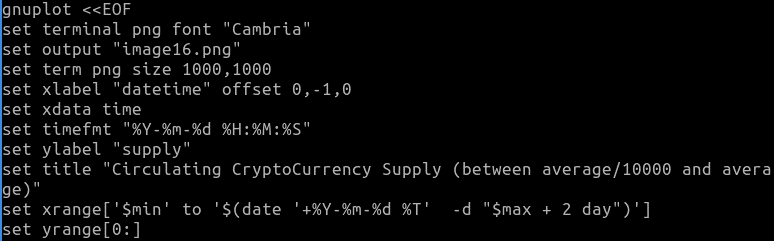
Read file typeSupply2.txt and pipe output to sed to get text at line i, storing the value into variable c



Select without column header ‘supply’ and ‘source\_date’ from table ‘crypto\_source’ inner join table ‘crypto\_supply’ on ‘source\_id’ where code = value of variable c, write result to dat file named according to the code of the cryptocoin in directory supply



Set up graph for plotting using gnuplot including details of the output file, label and offset position of x-axis, label of y-axis, specify data type on x-axis as time with format, title of graph, range of x-axis from min to 2 days ahead of max, range of y-axis from 0



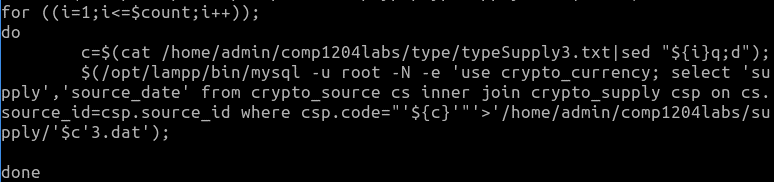
Loop through all files found in directory supply with name ending with ‘2.dat’, plot data in column 2 against data in column 1 with line points and substring of the name of file to get code of cryptocurrency as title for each curve plotted



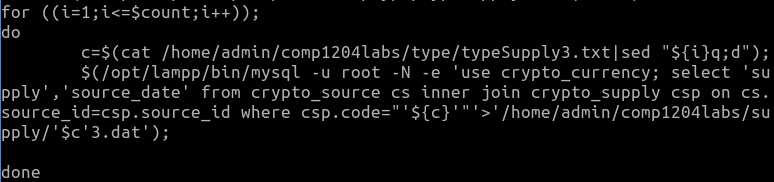
Count the number of lines in file typeSupply3.txt, storing the value in variable count



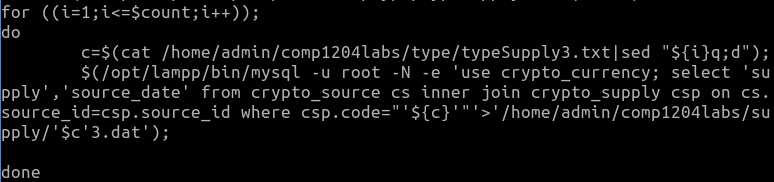
Loop through each line in file



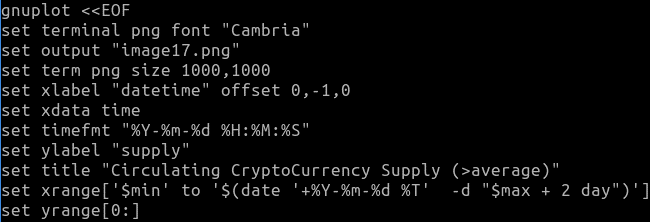
Read file typeSupply3.txt and pipe output to sed to get text at line i, storing the value into variable c



Select without column header ‘supply’ and ‘source\_date’ from table ‘crypto\_source’ inner join table ‘crypto\_supply’ on ‘source\_id’ where code = value of variable c, write result to dat file named according to the code of the cryptocoin in directory supply



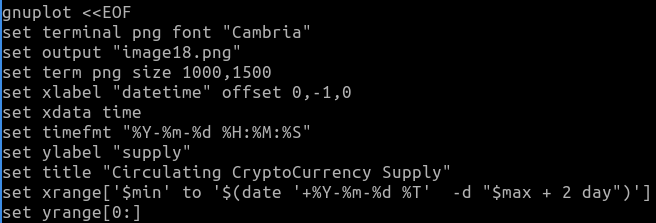
Set up graph for plotting using gnuplot including details of the output file, label and offset position of x-axis, label of y-axis, specify data type on x-axis as time with format, title of graph, range of x-axis from min to 2 days ahead of max, range of y-axis from 0



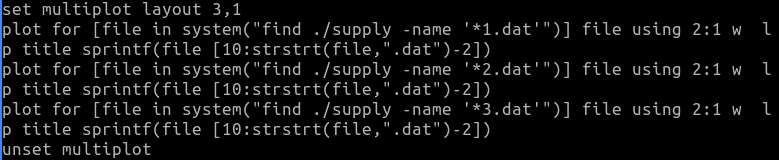
Loop through all files found in directory supply with name ending with ‘3.dat’, plot data in column 2 against data in column 1 with line points and substring of the name of file to get code of cryptocurrency as title for each curve plotted



Set up graph for plotting using gnuplot including details of the output file, label and offset position of x-axis, label of y-axis, specify data type on x-axis as time with format, title of graph, range of x-axis from min to 2 days ahead of max, range of y-axis from 0



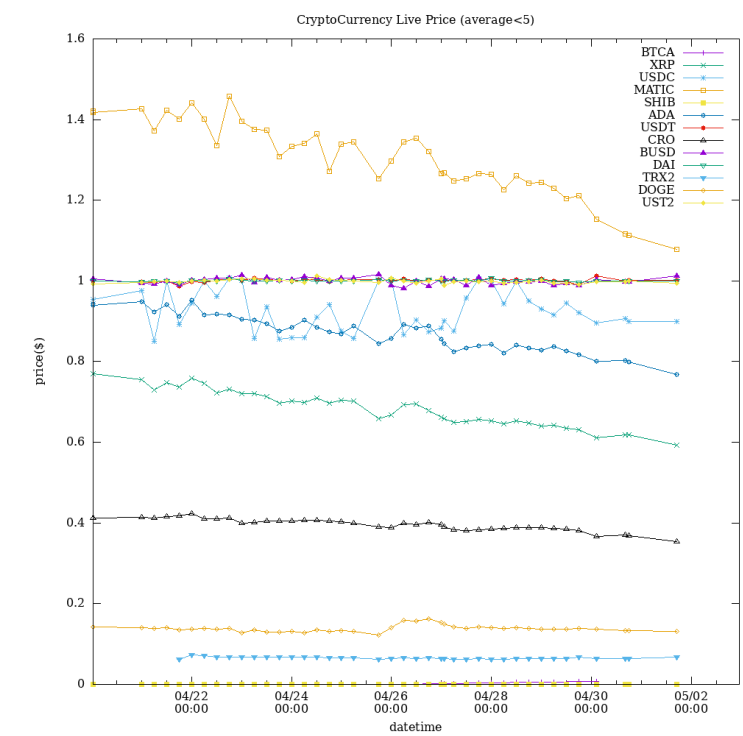
Set multiplot to plot multiple graphs on one output file, arranged as 3 graphs in each row and 1 graph in each column



Remove directory supply and directory type recursively

Output of plot

Plot:



title

with title

yrange

ylabel

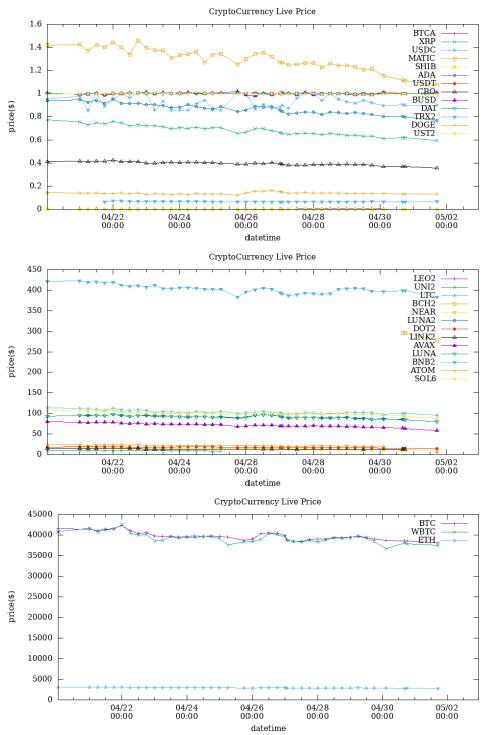
xrange

xlabel

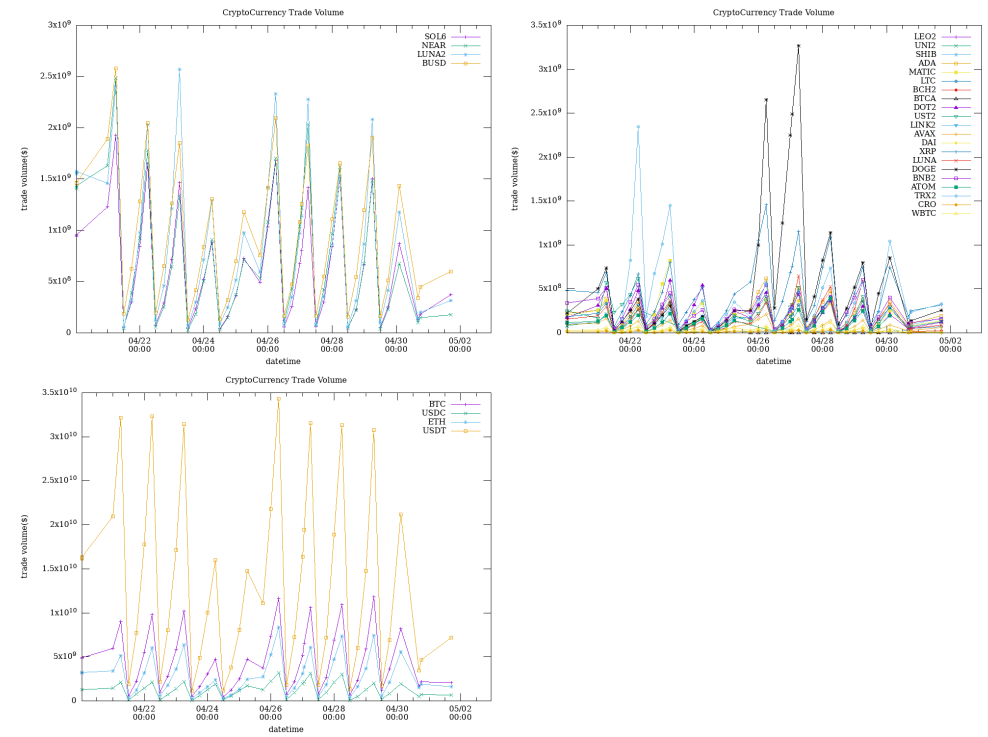
with line points

Multiplot:

Layout 3, 1



Layout 2, 2



# Use of Git for Version Control