ICS 53, Fall-2022
Assignment 5: Client/Server programming

You will write a **networked client/server application** which allows a client program to query a server program to fetch stock market information. Your programs will allow a user to examine stock prices for two major stocks namely Pfizer(PFE) and Moderna(MRNA). The user will enter commands into the client program which runs on one machine. For each command entered by the user, the client program sends a message containing the query command. The server program receives the query and responds with a message containing the requested information. Below, we will describe the details of query commands and the format of the stock data.

done **1. Running the Client/Server**

The application is actually two different programs, the client program and the server program. These two programs should both be running as two separate processes which are invoked by the user in a shell. These two programs can run on the same or different machines. Since the client and server are communicating using TCP/IP, you will need to select a port to use for communication. ICS Computer Support has advised us that ports 30000 – 60000 are all available for your use, so you can use any one of those ports.

2. Stock Data Format

Your program will use historical stock prices of MRNA and PFE as the stock prices used for buy and sell transactions. In addition to downloading this assignment from Canvas, you will need to download the following files: MRNA.csv and PFE.csv (please note these datasets are only for the purpose of this assignment and are not based on real stock prices of MRNA and PFE). These two files contain stock price data for MRNA and PFE respectively. When your program is executing, historical price data will need to be read from these three files before any transactions can be made. Each file contains stock price information on each trading day between 7/2/2019 and 6/30/2021. Each file is in ”CSV” format, which stands for ”comma-separated value”. The contents of the file are tables with a series of columns. The first line contains the titles of each column and every line after the first contains the data on each column. The column data on each line is separated by a comma. The data that we are interested in on each line is the date (first column) and the closing price (fifth column). Your program will need to read the closing price on each day and store it so that it can be used as the stock price for buy and sell transactions. You will notice that the files do not include information for all days such as weekends and holidays, so there are gaps in the data.

3. Commands Summary

Your program should accept the following commands related to historical price data. Your code should only use prices in the "Close" column. Prices should be rounded up to the second digit after the decimal point(191.449997 -> 191.45).

- **PricesOnDate**: This command should display the stock prices of both stocks on a given date. This command takes 1 argument, the date.

- **MaxPossible**: This command calculates the maximum possible profit/loss for a single share of a given stock in a given time span. In other words, if you buy your stock when the stock price is minimum/maximum and sell it when the stock price is maximum/minimum within the given time span, what will the profit/loss be. Note that the selling date must be later than the buying date. This command takes 4 arguments: profit/loss flag, name of the stock, start date, and end date.

- **quit:** This command ends the program.

done

4. Starting the Server

If we assume that the name of the server's compiled executable is "server" then you would start the server by typing the following at a linux prompt,

"./server PFE.csv MRNA.csv 30000"

(The last argument is the port number that the server will listen to. The previous arguments indicate the CSV files to be read by the server. You can assume that the first file name read from the command line is PFE and the second is MRNA.)

The server must first read the contents of the CSV files provided in the command line arguments. When the reading is finished, it should start listening to the client requests on the port mentioned as the last command line argument and should print "server started\n". It should wait to receive messages from a client.

done

5. Starting the Client

If we assume that the name of the client's compiled executable is "client" then you would start the client by typing the following at a linux prompt:

“./client server.ics.uci.edu 30000”

(The first argument is the domain name of the server and the second argument is the port number the server is listening to.)

Remember server.ics.uci.edu is just an example of the domain name of the machine that server is running on. If you run both client and server on the same machine, you can use “localhost” instead to loop back the connection between client and server. When the client starts it should print a “> ” prompt on the screen and wait for input from the user. (Note: “> “ is one > and one space.)

6. Message Format

Each message sent between the client and the server must contain two pieces of information, a string, and the length of the string. In messages sent from the client, the string will be the query command entered by the user. In messages sent from the server, the string will be the requested stock price (for the Prices command) or the profit value (for the MaxProfit command). Each message must be shorter than 256 bytes long and must be formatted as follows:

- Byte 0: Length of the string (n)
- Bytes 1 – n: Characters of the string

7. User interface of the Client

Requests for stock information are made by the user entering the query commands at the client’s prompt. When this information is entered at the client’s prompt, the client will send a request message containing the query command to the server, and the client will await a response from the server. The server will send a response message containing the corresponding information.. The client will print the received information to the screen, print a prompt on a new line, and wait for more input from the user. An example of a user interaction with the client is shown below. It is derived from the first row of database examples.

> PricesOnDate 2019-07-02
PFE: 187.18 | MRNA: 44.98
> PricesOnDate 2020-05-08
PFE: 202.90 | MRNA: 38.58
> MaxPossible profit PFE 2019-09-11 2019-10-15
14.41
> MaxPossible loss MRNA 2020-04-16 2020-08-23
6.37

> quit
←you are back to the linux prompt.
The client will continue in this loop until the user enters "quit" which will cause the client's process to exit. (Note: No need to print anything after quit was entered.)

8. User Interface of the Server

Once running, the server will only accept requests sent from one client over the network. When the server receives a request from a client, the server will print the *requested query command* in the message on the screen on a new line. An example of the printed output of the server when communicating with the client is shown below.

server started
PricesOnDate 2019-07-02
PricesOnDate 2020-05-08
MaxPossible profit PFE 2019-09-11 2019-10-15
MaxPossible loss MRNA 2020-04-16 2020-08-23

The server will continue responding to requests and printing the associated information until its process is killed externally, for instance by a ctrl-c typed at the keyboard. Quit command entered on the client **does not** get transmitted to the server or quit the server.

10. Implementation Details

- If the command entered into the client is invalid for any reason, just print "Invalid syntax\n".
- The commands and arguments should be case-sensitive.
- When the date is provided as an argument to a command, it should be provided in "YYYY-MM-DD" format.
- When a stock name is provided as an argument, it should be provided as its ticker symbol, PFE, or MRNA.
- If the date value typed into the client is not present in the stock's file which is known to the server, just print "Unknown" and continue the program.

11. Submission Instructions

Your source code must be two C files (client.c and server.c). Be sure that your program must compile on openlab.ics.uci.edu using gcc version 11.2.0. You can check your gcc version with the "gcc -v" command. Make sure both files can compile with "gcc filename.c". Submissions will be done through Gradescope. The first line of your submitted file should be a comment which includes your name and UCNETID as well as your partner's (if you are working with a partner).