CSCI 4061: Assignment 4
(Socket programming)

Released : Nov 23 (Mon), 2015 Due : Dec 11 (Fri), 2015 11:55 PM

Total Points : 100

1. Introduction

The purpose of this homework is to get started with network programming. For this purpose, we are going to implement a multi-threaded twitterTrend server and twitterTrend client.

TwitterTrend server is a simple program that takes city names from client programs and serves top 3 trending keywords to client programs. The server is serving multiple concurrent clients in a pool of threads. One thread on the server program handles a client program. TwitterTrend client sends the request to get the 3 trending keywords from the server. This client program have a clientX.in file that contains one or multiple city names. In addition, the communication protocol (messages) should be implemented to send and receive the data through the network.

In this assignment 4, you are going to implement a client-server twitterTrend program.

1.1 Rules

You may work on this assignment individually or in pairs. You are strongly advised to work in pairs.

If you have any questions, please post those on moodle.

2. The Assignment

2.1 Setup

You will need to build upon your program delivered in assignment 3 or download our solution to build upon it. The PA3 solution will be uploaded on moodle on Wed (11/25).

2.2 Your Tasks

Your task is to implement a twitterTrend program. The program consists of a server program and a client program.

2.2.1 TwitterTrend server

The server program is called from the terminal as follows:

./twitterTrend_server *port_number num_threads*

The executable file MUST be named twitterTrend_server. In addition, it takes two command line arguments.

1) Port Number

The first parameter is the port number on which the twitterTrend server will run. This parameter is NOT optional. Last 4 digits of your student ID is recommended as a port number due to clash.

2) Num threads

The second parameter is the maximum number of threads the server may run concurrently. This parameter is optional with a default value of 5.

- **Dataset**

The twitter dataset is a simple comma separated text file named “TwitterDB.txt”, which contains a city name followed by 3 keywords per line. You need to look up this dataset by city name to find corresponding trending keywords. Since, reading through a file multiple times is not efficient, you need to store the contents of TwitterDB.txt into in–memory data structure of your choice, so that the look up is efficient.

- **Description**

The twitterTrend_server main thread should create a TCP server socket to accept connections from TCP clients. The twitterTrend_server should create num_threads threads in addition to its main thread. Then, the twitterTrend_server main thread should insert the incoming clients' sockets in a shared queue. Each child thread should access the shared queue to get one client to handle, by removing the client from the queue and then handling the client. After handling a client, a child thread should get another client to handle, until the queue is empty. Please note that server should run forever and wait for new requests, even if the queue becomes empty from time-to-time. All clients must be served; therefore, if the number of clients is larger than the maximum number of threads passed to the program, all clients must still be handled, meaning one thread may need to serve several clients. Note that the shared queue size is unbounded. As you can notice, **thread behavior is exactly the same as assignment 3**. The twitterTrend_server runs forever to continuously handle incoming clients. Obviously, you need to use synchronization mechanisms to manage the access to this shared queue of clients.

Clients should be served concurrently. Each child thread should be assigned an integer id that ranges from 1 to num_threads. Each client should be identified by its IP/port combination in the form "(IP,port_number)" e.g. (127.0.0.1,XXXXX). **When a thread starts handling a client, the message "Thread *tid* is handling client *cid*" should be printed to stdout, where tid is the thread id and cid is the client id.** Please note that after accepting the request using accept(), the kernel allocates a random port number for the server for the further communication. Therefore, you should print out this allocated port number for cid. For each client, twitterTrend_server should produce 3 keywords for the client. 3 keywords should be sent back to the client according to the communication protocol described below. **This indicates that a server thread is about to handle a client's requests using communication protocol -- detailed further in Section 2.2.4. After**

sending the keywords, the thread should print to stdout the message "Thread *tid* finished handling client *cid*" with the same notation used above.

2.2.2 TwitterTrend client

The client program is called from the terminal as follows:

./twitterTrend_client *server_host_name server_port_number file_path*

The executable file MUST be named twitterTrend_client. In addition, it takes three command line arguments. The first parameter is the server hostname. The second parameter is the server port number. The third parameter is a file path of the file that contains the city names to be processed. These parameters must be provided. The client should send a twitterTrend request to the specified server using the given input text file line by line. After receiving the response, each client should save the output in file called after the input file appended with .result. For example, if the input file is client1.in, then the client should save the output in a file called client1.in.result. Also this result file must be created in the same directory and contain the result in the form <cityname : keyword1, keyword2, keyword3> appended by a newline. As a network programmer, you are encouraged to test your client program on a different machine from which hosts the server program.

2.2.3 Communication Protocol

On the client side:

- Step 1: Establish connection with server.
- Step 2: Wait for handshake message from server.
- Step 3: Send handshake response to server.
- Step 4: Send one line length (city name) followed by the line.
- Step 5: Wait for the 3 keywords response.
- Step 6: Write the 3 keywords to the output file.
- Step 7: Repeat steps 4-6 for each line of text.
- Step 8: Send end of request message.

On the server side:

- Step 1: Listen for incoming connection at the specified port.
- Step 2: Establish new connection.
- Step 3: Insert the client in the queue.
- Step 4: Any available thread should pick the client to start handling.
- Step 5: Send handshake message to client.
- Step 6: Wait for handshake response from client.

- Step 7: Wait for twitterTrend request.
- Step 8: Find 3 keywords.
- Step 9: Send back the 3 keywords followed by end of response message.
- Step 10: Wait for another request repeating steps 7-9 till end of request message received.
- Step 11: Close the connection with the client.

2.2.4 Messages

A message contains three fields:

- Message ID: a 32-bit integer that specifies the type of message

  For example, 100 means it is a handshake message.
- Length of payload: a 32-bit integer that specifies the size of the payload in bytes.
- Payload: the actual content of your message, it could be empty in some cases

The following table shows a complete list of message type:

Message **ID**	**Message type**
100	Handshake
101	Handshake response
102	TwitterTrend request
103	Response message
104	End of request
105	End of response
106	Error message

Detailed explanation of each message type is provided below:

- **Handshake**: A handshake message should not have anything in the payload, so set length of payload to 0. Server send this message to the client.
- **Handshake response**: A handshake response message should not have anything in the payload, so set length of payload to 0. Client send this message to the server after receiving a handshake message.

- **Twittertrend request**: This carries the city name to be searched. The length field should contain the length of the city name in bytes. The payload should contain the actual text. Client sends this message to the server.
- **Response message**: This carries the 3 keywords. The length field should contain the length of that text in bytes. The payload should contain the actual 3 keywords separated by commas. Server sends this message to the client as a response to the twittertrend request.
- **End of request**: The end of request message is sent after receiving the last 3 keywords by the client. The end of request message should not have anything in the payload, so set length of payload to 0. Client sends this message to the server after finishing all the text lines.
- **End of response**: The end of response message is sent after the whole the 3 key words have been sent. The end of response message should not have anything in the payload, so set length of payload to 0. Server sends this message to the client after finishing submitting the response.
- **Error message**: In case an unrecognized or malformed message is received, the connection should be terminated and an error message should be sent over to the other side of the network connection. The payload should be printed out to the terminal on the receiver of such message. Both server and client can send this kind of messages to the other party in case of any error.

2.2.5 Extra credit (10 points)

To get extra credit, twitterTrend_client should allow an arbitrary number of input files per client in the command line arguments instead of only one input file as in the basic requirements of the assignment. All the input files for the same client should be handled using one connection to the server. Output of each file should be saved in a separate text file that is named with the same convention mentioned before.

2.3 Examples

2.3.1 Example of dataset file

Minneapolis,UMN,Lakes,Snow
Paris,EiffelTower,Cheese,Fashion
London,MU,Soccer,LondonEye
Melbourne,Rove,Blacktown,AUSvNZ

2.3.2 Example for client input

Contents of client1.in
Minneapolis
Paris
London
Melbourne

2.3.3 Example for client output

Contents of client1.in.result
Minneapolis : UMN,Lakes,Snow
Paris : EiffelTower,Cheese,Fashion
London : MU,Soccer,LondonEye
Melbourne : Rove,Blacktown,AUSvNZ

2.3.4 Example of commands

./twitterTrend_server 2345 2 (in machine 0)
./twitterTrend_client 127.0.0.0 2345 client1.in (in machine 1)
./twitterTrend_client 127.0.0.0 2345 client2.in (in machine 2)

.
.
.

2.3.5 Example of stdout in server

Thread 1 is handling client 127.0.0.1,XXXXX
Thread 2 is handling client 127.0.0.2,XXXXX
Thread 1 finished handling client 127.0.0.1,XXXXX
Thread 1 is handling client 127.0.0.3,XXXXX
Thread 2 finished handling client 127.0.0.2,XXXXX
Thread 1 finished handling client 127.0.0.3,XXXXX
Thread 2 is handling client 127.0.0.4,XXXXX
Thread 1 is handling client 127.0.0.5,XXXXX
Thread 2 finished handling client 127.0.0.4,XXXXX
Thread 1 finished handling client 127.0.0.5,XXXXX
.
.
.

2.3.6 Example for communication

In this section, protocol message will be presented in triple format (message id, length, payload)

server listens
client connects
server accepts connection
server sends handshaking: (100,0,)
client sends handshake response: (101,0,)
client sends twitterTrend request: (102,11,"Minneapolis")
server sends twitterTrend response: (103,14,"UMN,Lakes,Snow")
server sends end of response: (105,0,)
client sends end of request: (104,0,)
server closes the connection

2.3.7 Example for making concurrent clients

A shell script (testscript.sh) file is provided in the given PA4 directory to make multiple clients in same machine. You can refer and modify this file to test your codes for concurrent clients.

2.4 Assumptions

- You MUST BE STRICT in the message formats printed to stdout; print them EXACTLY as indicated above. THIS IS IMPORTANT. Regrading due to errors in these messages will NOT receive full credit for any reason. A minimum of 10 points will be deducted.
- If the city name is not in the dataset, print out the NA in your result file.
- You can assume that the number of input clients will not exceed 100.

3. Grading

3.1 Submission Instructions and Deliverables

You are required to submit online, a single file (.zip or .tar or .tar.gz) containing:

- All of your code - *.c and *.h files. Your code should compile and run on ANY CSELabs Linux machines.
- A Makefile that will compile your codes and produce programs called twitterTrend_server and twitterTrend_client. Your code should all compile, just by us typing make. If we can't compile your code by typing "make", then you're definitely going to lose points.
- A file named **exactly** README (**not README.txt or any other name**) with the following info. If it does not, you are not following directions and you will lose 5 points for not following directions. Change the commented code below to include your personal information (and that of your partner if you have one) in the same exact format:

/*CSci4061 F2015 Assignment 4
* name: full name1, full name2 (for partner)
* id: x500_1, x500_2 (your x500, not a 7 digit number)*/

If you complete the extra credit, please indicate it here. Any additional information about your program that you feel we should know (e.g. some comments related to compilation, running, some errors you have ...etc)

- Do not include the test cases and the executable.

3.2 Rubric

Following instructions and README	: 5 points
Documentation and coding style	: 15 points
Functionality (test cases)	: 80 points