Introduction to Network Programming

Final Exam

Date: 2020/12/28 Time: 15:40-18:20

Problems:

1. (50%) Chat-room：
   - You have to create a TCP server which is a chat room. All clients are in the chat room.
   - Server will give client a name by connection order.
     For example, the first client connects to the server, it will be named user1, the second client will be user2, the third client will be user3.
   - Each client will receive a message of "Welcome, userX." when enter the chat room server.
   - Clients can send a private message to another user. Any client can not send a message to himself.

   Ex.

   Command format:

   - Server
     - ./server {port}
   - Client
     - ./client {server-ip} {server-port}

   Notes:

   - mute command – (5%)
   - unmute command – (5%)
   - yell command – (15%)
   - tell command – (15%)
   - system message – (5%)
   - exit command – (5%)

Command	Description	Output	
mute	The client will not receive any message, including public and private message.	Success	Mute mode.
		Fail	You are already in mute mode.
unmute	The client can receive messages again.	Success	Unmute mode.
		Fail	You are already in unmute mode.
yell <message>	Send a message to others in the chat room.	Success	<username>: <message> Ex. user1: Hello world.
tell <receiver> <message>	<sender> sends a private message to <receiver> in the chat room. <sender> and <receiver> are usernames.	Success	<sender> told you: <message> ex. When user1 sends a private message to user2, user2 will receive: **user1 told you: It is nice to see you.**
		Fail	<receiver> does not exist.
Exit	Close chat room and disconnect from the server.		

Scenario：

User0	**User1**	**User2**
bash$ ./client 127.0.0.1 7890 ********** * Welcome to the BBS server. * ********** Welcome, user0.	**bash$ ./client 127.0.0.1 7890** ********** * Welcome to the BBS server. * ********** Welcome, user1. **yell Hello world**	**bash$ ./client 127.0.0.1 7890** ********** * Welcome to the BBS server. * ********** Welcome, user2.
user1: Hello world **mute** Mute mode.	**yell Hello world2**	user1: Hello world user1: Hello world2
unmute Unmute mode. user1: Hello world3	**yell Hello world3**	user1: Hello world3
	tell user2 It is nice to see you.	user1 told you: It is nice to see you.
exit	**exit**	**exit**

p.s Bold means command.

2. (50%)Share memory (mutex)：

- You have to implement a TCP Server with two accounts, **ACCOUNT1, ACCOUNT2** respectively. Initial accounts are zeros. Give client a name by connection order.
  For example, the first client connects to the server, it will be named A, the second client will be B, the third client will be C, the fourth client will be D.
- When a client connects/disconnects to the server, server should output message: **New connection from ip:port (user#) / (user#) ip:port disconnected** (system message)

Note：

Don't deposit a negative number into accounts or withdraw excess money from accounts.

The client does the following functions:

- The service accepts the following commands and **only 4 clients**:
  When a client enters command incompletely, e.g., missing parameters, the server should show command format for client.

Command	Description	Output	
show-accounts	Show the money in ACCOUNT1, ACCOUNT2.	Success	ACCOUNT1: <money> ACCOUNT2: <money>
deposit <account> <money>	Deposit the amount of money into <account> by client. **<money> is only positive integer.** Ex. 1.Client C deposits $100 into ACCOUNT1. -> deposit ACCOUNT1 100	Success	Successfully deposits <money> into <account>.
		Fail	Deposit a non-positive number into accounts.
withdraw <account> <money>	Withdraw the amount of money from <account> by client. **<money> is only positive integer.** Ex. Client D withdraws $300 from ACCOUNT1. -> withdraw ACCOUNT1 300	Success	Successfully withdraws <money> from <account>.
		Fail(1)	Withdraw excess money from accounts.
		Fail(2)	Withdraw a non-positive number into accounts.
exit	Disconnect from the server.		

- **Scenario：**

A	**B**
bash$ ./client 127.0.0.1 7890 ********** * Welcome to the TCP server. * ********** **deposit ACCOUNT1 1000** Successfully deposits 1000 into ACCOUNT1.	**bash$ ./client 127.0.0.1 7890** ********** * Welcome to the TCP server. * ********** **withdraw ACCOUNT1 500** Successfully withdraws 500 from ACCOUNT1.
deposit ACCOUNT2 1000 Successfully deposits 1000 into ACCOUNT2. **deposit ACCOUNT1 -3** Deposit a non-positive number into accounts. **deposit ACCOUNT1 0** Deposit a non-positive number into accounts.	**withdraw ACCOUNT1 1500** Withdraw excess money from accounts. **withdraw ACCOUNT1 0** Withdraw a non-positive number into accounts.
withdraw ACCOUNT1 -100 Withdraw a non-positive number into accounts.	**show-accounts** ACCOUNT1: 500 ACCOUNT2: 1000
exit	**exit**

Notes:

- system message – (5%)
- show-accounts command – (10%)
- deposit command – (15%)
- withdraw command – (15%)
- exit command – (5%)

Submission:

You have to submit your file in the following format.

<your_student_id>/

├── P1/ --- This directory is for storing your answer to the problem 1.

Put your **server codes/client codes** and a readme file that describes how to compile your program (Makefile is better) here.

└── P2/ --- This directory is for storing your answer to the problem 2.

Put your **server codes/client codes** and a readme file that describes how to compile your program (Makefile is better) here.

Type **zip –r <your_student_id> <your_student_id>** and upload the **<your_student_id>.zip** to new E3.