Intro. to Network Programming 2022 Fall

Homework 1 – Game 1A2B: Part 1

Description

In this project, you are asked to use only **C/C++** to design a **1A2B game** server and client. Your program should be able to handle multiple connections and receive user commands from **standard input**. After receiving the command, the server sends the corresponding message back.

The logic of this project:

Client to do: Receive message from User -> Send to Server -> Receive response from Server -> Show the response

Server to do: Receive message from Client -> Do corresponding work -> Send response to Client

Requirement

The service accepts the following commands and **at least 10 clients**. When a client enters a command incompletely e.g., missing parameters, the server should show **command format** for the client.

Command format	Description	Result	
register <username> <email> <password>	Register with username, email and password. <username> and <email> must be unique, <password> has no limitation. If username or email is already used, show corresponding failed message, otherwise it is success. **Note:** You have to send this command and get associated message by **UDP**.	Success	Register successfully.
		Fail (1)	Username is already used.
		Fail (2)	Email is already used.
login <username> <password>	Login with username and password. Fail (1): User already logs in. Fail (2): Username not found. Fail (3): Password not correct. **Note:** You have to send this command and get associated message by **TCP**.	Success	Welcome, <username>.
		Fail (1)	Please logout first.
		Fail (2)	Username not found.
		Fail (3)	Password not correct.
logout	Logout account. If the user hasn't logged in yet, show failed message, otherwise logout successfully. **Note:** You have to send this command and get associated message by **TCP**.	Success	Bye, <username>.
		Fail	Please login first.
game-rule	Show the game rules. Users who are not logged in can still use this command. **Note:** You have to send this command and get associated message by **UDP**.	1. Each question is a 4-digit secret number. 2. After each guess, you will get a hint with the following information: 2.1 The number of "A", which are digits in the guess that are in the correct position. 2.2 The number of "B", which are digits in the guess that are in the answer but are in the wrong position. The hint will be formatted as "xAyB". 3. 5 chances for each question.	

<table>
<tr><td rowspan="7">start-game <4-digit number></td><td rowspan="7">Start a 1A2B game.

If the user hasn't logged in yet, show Fail (1).

In this command, <4-digit number> is an optional parameter.

If parameter <4-digit number> is given, check whether it is a 4-digit number. If yes, use this number as the question; if not, raise Fail (2).

If parameter <4-digit number> is not given, randomly initialize a 4-digit number as the question.

Fail (1): User hasn't log in.

Fail (2): Incorrect question

Start the game if no fails.

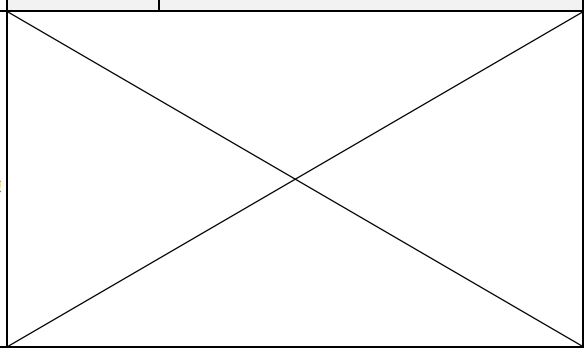
If the guess isn't a 4-digit number, raise Fail (3), and this guess won't be included in the 5-times limitation.

Fali (3): Invalid guess.

The server should show the corresponding messages depending on the user's guess and the game rule.

Note: You have to send this command and get associated message by TCP.</td><td>Fail (1)</td><td>Please login first.</td></tr>
<tr><td>Fail (2)</td><td>Usage: start-game <4-digit number></td></tr>
<tr><td>Start</td><td>Please typing a 4-digit number:</td></tr>
<tr><td>Fail (3)</td><td>Your guess should be a 4-digit number.</td></tr>
<tr><td>Wrong</td><td>The corresponding hint. E.g., 0A4B / 1A3B / 2A2B / 3A1B</td></tr>
<tr><td>Correct</td><td>You got the answer!</td></tr>
<tr><td>Guess more than 5 times</td><td>You lose the game!</td></tr>
<tr><td>exit</td><td>The server closes the connection with the client, and the server will continue to run.

If the user on this client is logged in when the server receives this command, please log the account out first. (Do not need to print "Bye, <username>.")

Note: You have to send this command by TCP.</td><td colspan="2">  </td></tr>
</table>

Grade (100%)

Socket connection and **exit** command - (30%)

Subsequent commands will only be scored if **socket connection** and **exit** command are completed.

- **register** command - (15%)
- **login** / **logout** command - (20%)
- **game-rule** command - (5%)
- **start-game** command - (20%)
- Demo - (10%)

Senario 1

bash$ ./client 127.0.0.1 8888

*******Welcome to Game 1A2B*******

register

Usage: register <username> <email> <password>

register Bob bob@qwer.asdf 123456

Register successfully.

register Bob asdf@asdf.asdf 123456

Username is already used.

register Tom bob@qwer.asdf 123456

Email is already used.

login

Usage: login <username> <password>

login Bob

Usage: login <username> <password>

login Bob 654321

Password not correct.

login Tom 654321

Username not found.

login Bob 123456

Welcome, Bob.

login Bob 123456

Please logout first.

logout

Bye, Bob.

logout

Please login first.

exit

Scenario 2

bash$ ./client 127.0.0.1 8888

*******Welcome to Game 1A2B*******

register Tom asdf@qwer.asdf 123456

Register successfully.

game-rule

1. Each question is a 4-digit secret number.

2. After each guess, you will get a hint with the following information:

2.1 The number of “A”, which are digits in the guess that are in the correct position.

2.2 The number of “B”, which are digits in the guess that are in the answer but are in the wrong position.

The hint will be formatted as “xAyB”.

3. 5 chances for each question.

start-game

Please login first.

login Tom 123456

Welcome, Tom.

start-game

Please typing a 4-digit number:

0000

0A4B

1111

1A0B

2222

1A0B

1222

2A0B

1233

3A0B

You lose the game!

start-game abcd

Usage: start-game <4-digit number>

start-game 5678

Please typing a 4-digit number:

5678

You got the answer!

exit

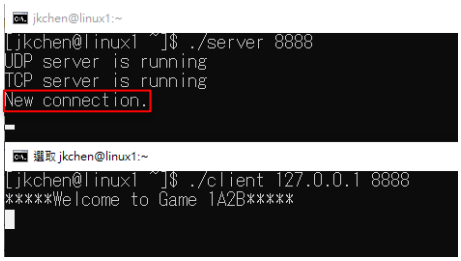
General

1. Please make sure your program can run on the **Linux** workstation of **NCTU CSCC**.

2. Please make sure your server uses **SO_REUSEADDR**, thus other servers can bind to this port after your server closes.

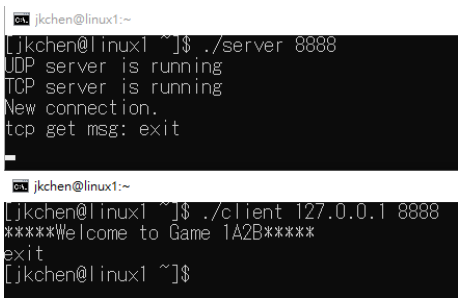
3. You need to write your own **makefile** and name the executables **server** and **client** respectively.

4. To run your server and client, you must provide **port number** and **IP + port number** respectively.



   When client connect to server, the server print message "**New connection.**" and the client will show the response message "*******Welcome to Game 1A2B*******" send from the server.

5. After receiving the **exit** command, the server closes the connection with the client, but the server continues to run to accept connections from other users.



6. The due date for HW1 is **10/18(Tue.) 23:59**. For more information about submission, please refer to **Submission** below.

7. The demo date for HW1 is **10/20(Thu.)**. The demo appointment will be announced on **10/19(Wed.)**.

8. If you have any questions, please use **Microsoft Teams** to ask. If you send email to TAs, you won't get response.

Submission

Please upload a zip file called "**HW1_StudentID.zip**" e.g., HW1_310551077.zip to **E3**. And the zip file should include three files: **server.c/server.cpp, client.c/client.cpp,** and **makefile**.

1. **Submission that don't follow the rule** will **get 20% punishment** on the grade.

2. Late submission will get **2 points deducted** for each day late. The late submission demo will be held on **11/10**.

3. You will get **0 points** on this project for **plagiarism** or **submission late for over 20 days**.

Demonstration

We provide two files: **test.sh** and **connection.txt**. When check whether the command works on your program, we will run a script like **test.sh** to compare the difference between the outputs of your client program with the answer like **connection.txt.**

You can use this script to check by yourself before submission. To run **test.sh**: **./test.sh <IP> <PORT>**