RMIT University

COSC1176/1179 – Network Programming

Semester 1, 2018

Assignment

Aim: To apply the learnt networking concepts to a practical application

Method: This assignment will be attempted by students individually.

Due date: 18 May 11:59 pm (Week 10)

Submission Method

- Your submission should consist of three parts: (i) all java source files, (ii) a Makefile and (iii) documentation.
- You should put everything into one zip file, and submit the zip file. That file should be named as ***sxxxxxxx.zip***, where ***xxxxxxx*** is your student number.
- Submission is via Canvas.

Marking

- The assignment is marked out of 100, and contributes 25% to your final mark.
- You must demonstrate your solution in the lab to get it marked. (**No demo, no marks**.)

Late submission penalties apply

Unless an extension has been granted (for procedures and grounds see http://www.rmit.edu.au/students/assessment/extension), a penalty of 10% of the total project score will be deducted per calendar day, and no submissions will be accepted 5 days beyond the due date.

Special Consideration

For special consideration please visit http://www1.rmit.edu.au/students/specialconsideration. With the exception of dire circumstances, no extension requests will be considered within 5 working days of the submission date. ("Dire Circumstances" means things like hospitalization of you or a close relative, etc.) Persons requesting a late extension may be required to prove that a significant body of the work has already been completed.

Introduction

In this assignment, you are to develop a simple game played via the network.

General Specification

Your task is to implement a simple game with separate client and server machines. The player is sitting in front of the client machine (one player per client), and the server is administering the game. You have to implement the game in two stages.

> **Stage 1:** Single-player game: one client communicates with the server. When you have completed stage one, save a copy of it for submission.
>
> **Stage 2:** Multiplayer game: you need to modify the server so that it can handle several (up to six) clients simultaneously.

The game:
It is a simple guessing game. The server generates a secret code - an X digit number (where X is a number from 3 to 8 determined by the client at the start of the game). The client's task is to guess the combination. Each combo has only unique digits (no repeats). The client task is to guess the exact combination with the fewest number of guesses (up to a maximum of 10 guesses).
For each incorrect guess the client gets a clue in the form of two numbers:

Correct Positions: number of digits in guess that are in the correct position
Incorrect Positions: number of digits in guess that occur in the code but are in an incorrect position

If the client correctly guesses the code, the server announces the number of guesses made.
If the client fails to guess the code after 10 attempts, the server announces the code.

Multiplayer version:
The server maintains a lobby queue. Clients have to register with the server (using their first name) to be added to the lobby queue before they can play. Clients can register at any time.

The game is played in rounds. At the start of each round, the server takes the first three clients from the lobby (or all clients if there are less than three clients in the lobby), and starts the round. First the server announces the number of participants. Then the player in the group that was first in the lobby queue gets to choose the number of digits in the code (X).
The server then generates a random code with X unique digits. The game then enters the guessing phase. Each player can guess at any time (with their number of guesses tracked by the server).

Once all players have either:

> Correctly guessed the code,
> Reached their maximum guesses – 10,
> or
> Chosen to forfeit by pressing f (giving them a guess count of 11)

The server announces to all clients the number of guesses for each client (ranked from lowest to highest). Players can then choose to play again (p), or quit (q). The players that chose to play again are added back into the end of the lobby queue, and the process repeats with a new round.

Program components

The client

It connects to the server, signs up for each game round, (if player one, gets the code length X from client and sends to server), gets guesses (numbers) from the player (console) and forwards them to the server. It displays the server's messages to the player.

The server

It maintains a lobby queue, accepts players' (clients') requests to sign up, and selects the players for each round.

It also manages the game. First it announces the number of players, receives (X) the number of digits in the code from the first player and generates a number with X unique digits. In the guessing phase, it takes guesses from players, compares it to the secret code and sends the 2 digits back to the player (Correct Positions and Incorrect Positions) while updating their guess count.

After all players have completed their game (by correctly guessing/reaching max guesses/choosing to forfeit), the server announces the final rankings based on number of guesses, and allows players to either play again by entering p or quit by entering q.

Logging

The server logs the events in the game in both single player and multiplayer cases. There will be two separate logs, one for gaming and one for communication. The separation of logs is for security reasons, those having access to the communication log will not be able to learn about the intricacies of the game and vice versa. Each log entry should contain the date and time, the client ID or remote socket address and the action performed (e.g. "Guess made" or "Data sent").

The platform

You have to use Java for programming both client and server. The communication is via sockets. You need to choose the appropriate socket for each task, and explain why that socket is the most suitable.

Documentation

You need to prepare a short user instruction sheet, and a longer document describing how the program operates. You can attach a class diagram, if it makes it easier to explain your program's operation. You also need to insert comments into your program, as appropriate. The documentation should be no longer than two pages.

Miscellaneous

A simple text user interface will suffice. The user (player) should be notified about events and messages sent by the server. This includes information sent by the server about other players in the multiplayer game.

The program should have a reasonable timeout (up to one minute) when waiting for user input or server message, when appropriate.

Notes:

- In the multiplayer version, make sure that clients are handled correctly when signing up for the game and starting the game.
- Make sure communication has proper timeouts, but you keep the communication channel open when the server is moderately busy.
- Your program will be marked on the Moonshot servers.

The End