CSCI 641 Programming Skills: Efficient Design in Modern C++

Graduate Assignment - Who wants to be a Millionaire

Maha Krishnan Krishnan
Abdul Hakim Shanavas

Introduction

Goals

The goal of this graduate assignment lab is to gain experience with fundamental socket programming and concurrency topics in modern C++ such as:

1. Concurrency and threads
2. Socket Programming

Overview

This quiz game is inspired by the TV show Who wants to be a Millionaire.
Multiple clients should be able to connect to the server and play this game simultaneously.
Each client will play their own game. One server must serve multiple clients at the same time
using socket programming and threads. Create one thread for each client and the threads will run
in the background.

Game Rules

1. The game contains 10 questions with four options for each question.
2. The player must answer all the 10 questions correctly to win this game.
3. Player will have 3 life lines for each game.
4. Player's life line will be auto decremented each time the player answers the question wrong.
5. When a player has answered a question wrong, the same question will be repeated if the player
has still life lines left or else the game gets over.
6. As the game progresses, the level of difficulty increases.
7. Each game is divided into three sections.
8. First section - Questions 1 to 4

Second section - Questions 5 to 7

Third section - Questions 8 to 10

9. When the player completes a section, the player would take home minimum the amount
of that section last question money. Eg., if the player has successfully answered the first section correctly,
the player would take home minimum the amount won in first section even if the player loses the game before finishing
the next section.

10. Detailed prize money for each question is given below in the table.

Answered Correctly	Prize Won
1st Question	$ 2,000
2nd Question	$ 4,000
3rd Question	$ 8,000
4th Question	$ 16,000
5th Question	$ 32,000
6th Question	$ 64,000
7th Question	$ 128,000
8th Question	$ 250,000
9th Question	$ 500,000
10th Question	$ 1,000,000

Design

You should follow the design given in the documentation for the game.
Below are the guidelines for designing and implementing this lab.

1. There should be separate class for the game logic.
2. The game questions should be initialized only once. This can be done
using a static method or a singleton class.
3. Initializing the questions and answers can be done as soon as the the server is fired up.
4. You are free to have your own private variables needed for the game logic.

5. You should not use any global variables.
6. Each player/client should have an object of the game logic.
7. server.cpp and client.cpp are the two main files for the server and client respectively.
8. Server main class should only be responsible for the following:
    1. Initializing game's questions and answers
    2. Listening and accepting incoming client connections
    3. Creating a game object
    4. Firing a thread for each game object and detaching the thread.
9. Client should only be responsible for the following:
    1. Connecting with server
    2. If connection successful, should send messages to server entered by player
    3. Receive messages from the server and display it to the user
    4. Close socket connections and finish the program if the game is over.
    5. To play again, the player has to run the program again.
    6. There should be no game logic implemented at the client side. Everything should be done at the server side.

Starter Code

You are being provided with a structure for the lab. You are free to use more classes. However, your client and server's main program should be client.cpp and server.cpp. You are also provided Makefile for both server and client that support Unix machines.

You are also provided a set of 10 questions along with their options and answers to use for the program.

You are also free to choose your own question bank for the game.

Your lab should compile and run in the CS machines.

Implementation

Input

This program has to take inputs from the player such as player's name and answer to the question.

There should be command line arguments for both server and client. If the program is provided with

any command line arguments, then a usage message must be displayed and exit the program i.e.,

Usage: server
Usage: client

Output

You are provided with a few output files for both client and server for verifying and matching your output.

Road Map

Here is a suggested road map for how you should develop things:

1. Implement the game logic and you should be able to play the game in the local machine.
2. Implement the basic server and client connection where server just echoes the messages sent by client.
3. Connect your game logic and the server program.
4. Change your client program to receive messages from server for the game logic and send messages taken as an input from
the player.
5. Your server should now be able to accept one connection and allow the client to play the game.
6. Now change your server implementation to serve multiple clients and allow n players to play their game simultaneously.
7. Test it in CS machines