EX4: Guess the Number

In this exercise, you'll implement a simple game using TCP and select().

You'll write gameServer.c.

The server waits for players' guesses. Once the server reads a guessing from the player, it reads it.

The server should be run like this:

./server <port> <seed> <max-number-of-players>

The server listens on port 'port' and can talk with up to 'max-number-of-players' concurrently, each on a different socket. The server should use the given 'seed' for the srand() function.

When a player successfully connects to the server, the server does the following:

1. Assigns an id to the player, this id is between 1 to 'max-number-of-players'
2. Sends to this player the message "Welcome to the game, your id is ID", where ID is the assigned id.
3. Sends to all other players (except the newly joined player): "Player ID joined the game", where ID is the assigned id.

When the server gets a guessing from a player, the server does the following:

1. The server sends the following to all players: "Player ID guessed X" (where ID is the player's id and X is the guessing from the player.)
2. If the guessing is incorrect, the server sends the following to all players:
   "The guess X is too high", or
   "The guess X is too low"
3. If the guessing is correct
   a. The server sends the following to all the players:
      "Player ID wins", where the ID is the id of the winner.
      "The correct guessing is X".
   b. The server closes all clients' sockets after sending the messages.
   c. The server generate a new random number and wait for new players to join.

When a player disconnects to the server, the server does the following:

1. Mark the assigned id as available
2. Sends to all other players (except the disconnected player): “Player ID disconnected”, where ID is the id of the disconnected player.

All messages should be ended with \n.

Players can also disconnect from the server on their own.

You should catch CTRL-C, free everything and exit.

Note that there is only one thread (the main thread), therefore, any I/O operation should never block.

You should use select to handle all sockets’ descriptors for reading and writing so you should maintain both readset and writeset. Before any I/O operation, you should verify that the socket is ready for the specific operation.

You should maintain a data structure to save active sockets descriptors, massages to send, and any other information you may need.

For example, if the server reads a guessing from player ‘4’, the server’s response should be saved in the queue of all players, and on the next call to select, all other sockets should be checked for writing. If a player’s queue is empty, you should not check the player’s descriptors for writing.

If the server writes a message in multiple lines, the server should print only one line in each select iteration.

Try to write efficient code :)

In case of any failure, use perror for system calls or fprintf and exit.

Whenever the welcome socket is ready to read, print:

“Server is ready to read from welcome socket <sd>\n

Whenever your socket is ready to read or write from any client’s socket, print:

“Server is ready to read from player <id> on socket <sd>\n” or

“Server is ready to write to player <id> on socket <sd>\n”

Your program should never be in “busy wait”, so if you see a lot of printings, you have a bug.

There should be only one call to select in your code.

You must verify that the input is valid, i.e., there are 3 arguments, all arguments are numbers, port is between 1 to 2^16, and max-number-of-players is greater than 1.

In any of the above errors, print "Usage: ./server <port> <seed> <max-number-of-players>", and exit.

Test your server:

You 'telnet localhost <port>' where port is the port your server listens to.

Try to add many clients, before and during a game.

When a client is disconnecting, verify that if there is a client that is waiting to play, this client gets connected to the game.

In order to disconnect a client, type CTRL-5 and then type quit.

I also published my solution, so you can test your output against it, I hope it has no bugs 😊

Have fun

GOOD-LUCK!