REVISIÓN PRÁCTICA 2

Curso: **3º**

Curso Académico: **2018-2019**

Asignatura: **REDES**

Correo: **p92supeg@uco.es; i22fesad@uco.es**

Primer Cuatrimestre

Fecha: **23/10/18**

Guillermo Sugráñez Pérez y David Sánchez Fernández

ESCUELA POLITÉCNICA SUPERIOR. UNIVERSIDAD DE CÓRDOBA

UNIVERSIDAD D CÓRDOBA

ESCUELA POLITÉCNICA SUPERIOR DE CÓRDOBA
Universidad de Córdoba

CONTENIDOS

1. FUNCIONALIDAD REALIZADA 2

1.1. Funciones Auxiliares 2

1.2. Funciones de Usuario o Jugador 3

1.3. Funciones de Partida 4

1. FUNCIONALIDAD REALIZADA

En esta sección se listan y describen brevemente las funciones realizadas hasta la fecha relativas a la práctica 2, resumiendo así la funcionalidad que incorpora el código.

1.1. FUNCIONES AUXILIARES

Las funciones auxiliares son relativas al servidor, al cliente y al intercambio de mensajes que realizan entre ambos.

- **`int salirServidor(InfoServidor * servidor);`**

Permite salir del servidor liberando la memoria reservada para las estructuras en tiempo de ejecución.

- **`void nuevoCliente(InfoServidor * servidor);`**

Permite reservar memoria e inicializar una nueva instancia de cliente cuando un cliente nuevo se conecta al servidor.

- **`void salirCliente(int socket, InfoServidor * servidor);`**

Es la función inversa de la anterior. Libera la instancia de la estructura cliente cuando el cliente se desconecta del servidor.

- **`void mensajeDeCliente(int socket, char * mensaje, InfoServidor * servidor);`**

Esta función recoge el mensaje enviado por el cliente, determina qué acción está solicitando y llama a la función correspondiente.

- **`void difusion(const char * mensaje, InfoServidor * servidor);`**

Esta función permite enviar un mensaje a todos los clientes conectados.

- **`void extraerInstruccion(const char * cadena, char * subcadena);`**

Esta función permite extraer la instrucción o acción que desea realizar al cliente del mensaje que envía.

1.2. FUNCIONES DE USUARIO O JUGADOR

Estas funciones son las usadas para los usuarios (o jugadores), y permiten entrar y salir del juego, registrándose o logueandose en el servidor.

- **`int entrarUsuario(int socket, char * nombre, InfoServidor * servidor);`**

Esta función permite loguearse a un cliente si ya estaba registrado anteriormente. Es la que crea la estrucutra de usuario que permite almacenar la información del jugador, como por ejemplo su nombre y si está en una partida o no.

- **`int salirUsuario(Usuario * u, InfoServidor * servidor);`**

Esta función permite salir a un cliente del juego (que no de la partida) y liberar memoria de su estructura asociada.

- **`int registrarUsuario(char * mensaje);`**

Permite a un cliente registrarse con un nombre y una contraseña. Si el usuario ya estaba registrado mandará un error. Esta función tiene asociado un archivo de texto que permite tener el registro de todos los usuarios que han entrado alguna vez en el juego.

- **`int comprobarUsuario(const char * nombre);`**

Es una función auxiliar de registrar usuario. Comprueba que el usuario no existiera antes.

- **`Usuario * buscarUsuario(int socket, InfoServidor * servidor);`**

Esta función también es una función auxiliar. Permite encontrar al usuario especificando el socket asociado al cliente asociado a dicho usuario.

- **`int indicarUsuario(char * nombre, int socket, InfoServidor * servidor);`**

Esta función determina si el nombre de usuario introducido está registrado en el juego (en el archivo de texto auxiliar).

- **`int validarUsuario(char * mensaje, Usuario * u, InfoServidor * servidor);`**

Cuando se ha introducido un nombre de usuario correcto a la hora de loguearse, esta función permite comprobar si la contraseña asociada es la correcta y permitir o no el acceso al juego.

1.3. FUNCIONES DE PARTIDA

Una vez haya usuarios o jugadores registrados y logueados correctamente en el juego, estas funciones se encargarán de que puedan jugar unos contra otros:

- **int buscarPartida(Usuario * u, InfoServidor * servidor);**

Esta función permite meter al usuario en la cola para encontrar un oponente y empezar una partida. Cuando haya dos usuarios en cola que quieran jugar se creará una nueva partida, respetando el orden de petición de partida y asignando el turno al jugador que pidió entrar en la partida primero.

- **void meterCola(Usuario * u, InfoServidor * servidor);**

Esta función comprueba que hay espacio en cola y le asigna la posición correcta en la cola al usuario que quiera encontrar una partida.

- **void crearPartida(InfoServidor * servidor);**

Esta función reserva memoria e inicializa la estructura asociada para la partida, donde se guardan los tableros, los jugadores que la componen, el turno, etc.

- **void finalizarPartida(Partida * p, InfoServidor * servidor);**

Esta función es inversa a la contraria. Cuando finaliza una partida y el límite de partidas se había alcanzado impidiendo que hubiera nuevas, comprueba que se pueda crear una nueva partida.

- **void reorganizarCola(InfoServidor * servidor);**

Esta función se encarga de reorganizar la cola cuando dos usuarios o jugadores inician una partida o se desconectan del servidor, abandonando así la cola.

- **char ** crearTablero(int n);**

Esta función reserva memoria para un tablero de dimensión 12x12. Es 12x12 para ayudar a la codificación de los algoritmos del juego del buscaminas, pero solo se usan las 10x10 casillas reglamentarias.

- **void borrarTablero(char ** c);**

Función contraria a la anterior.

- **void tableroCadena(char ** c, char * respuesta);**

Esta función "traduce" la matriz que representa el tablero a una cadena para ser enviada al cliente correspondiente.

- **void crearBombas(char ** c, int numBombas);**

Crea 20 bombas y las distribuye de forma aleatoria en el tablero solución de la partida. Además del tablero solución, está el tablero que ven los jugadores, que es independiente de éste.

- **int descubrir(char * mensaje, Usuario * u, InfoServidor * servidor);**

Esta función descubre una posición del tablero solución. Se tienen que especificar correctamente y sino mandará un error.

- **int ponerBandera(char * mensaje, Usuario * u, InfoServidor * servidor);**

Función similar a la anterior. Establece una bandera A, o B en el tablero de juego (que no en el solución).