Programación de Sistemas Distribuidos

Curso 2022/2023

Práctica 1

Diseño e implementación de un sistema de juegos on-line

Esta práctica consiste en el diseño e implementación de un sistema de juegos on-line. Concretamente, es necesario desarrollar tanto la parte cliente, como el servidor, para permitir partidas remotas del juego *conecta-3x3*.

Al juego *conecta-3x3* se juega con un tablero, colocado en posición vertical, el cual está dividido en columnas. Cada columna contiene un número determinado de huecos donde pueden colocarse las fichas. Estas fichas se introducen por la parte superior de una columna y caen hacia abajo, de forma que una ficha puede estar en cualquiera de las posiciones de una columna, dependiendo de las fichas que hayan sido introducidas previamente.

Para jugar una partida de *conecta-3x3* son necesarios 3 jugadores. Cada jugador tiene un color único para sus fichas. En su turno, cada jugador debe introducir una única ficha en una de las columnas del tablero. Éste puede elegir cualquier columna salvo aquéllas que estén llenas, es decir, ya contienen el número máximo de fichas y no es posible introducir nuevas. El juego se desarrolla de forma que cada jugador introduce una ficha y cambia el turno al siguiente jugador.

El objetivo del juego es conseguir colocar 3 fichas en posiciones consecutivas, ya sea horizontal, vertical o diagonalmente. El juego se da por finalizado cuando ocurre una de las siguientes situaciones: i) Un jugador gana; ii) El tablero se llena de fichas y no existe un ganador. En este caso, se produce un empate. Para el desarrollo de esta práctica **NO será necesario implementar la lógica del juego**. El objetivo de la misma se centra en la parte encargada de las comunicaciones entre los clientes (jugadores) y el servidor.

Para familiarizarse con el juego *conecta-3x3*, se proporciona el ejecutable del juego implementado en modo local.

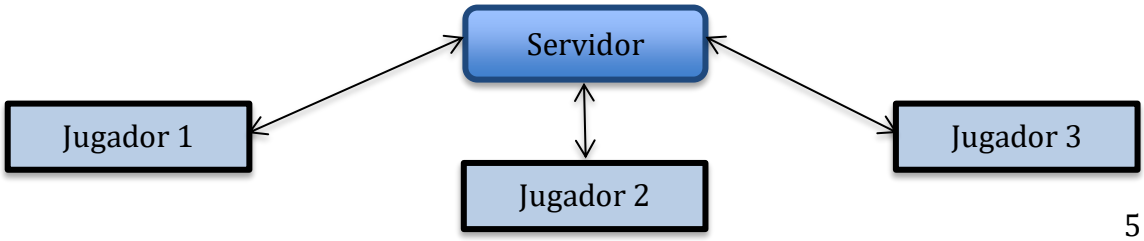
1.- Estructura general del sistema

La estructura general del sistema a desarrollar cuenta con dos partes, la parte cliente y la parte servidor. En esta práctica, ambas partes se desarrollarán en **C utilizando sockets**.

La parte cliente será el programa que ejecuten los jugadores. Esta parte no se encarga, en ningún momento, de controlar la lógica del juego. Su función es mostrar por pantalla el estado del juego actual, esto es, el contenido del tablero. Además, recogerá los movimientos introducidos por el jugador.

La parte servidor controlará la lógica del juego (p.e. controlar que no se realice un movimiento no válido, calcular cuándo se ha terminado la partida, etc…). Además, se encargará de controlar los turnos de los jugadores involucrados en la partida.

Para que pueda dar comienzo una partida, será necesario que 3 clientes estén conectados al servidor. De lo contrario, la partida no podrá comenzar. El siguiente esquema muestra la arquitectura del juego.



En la comunicación entre cliente y servidor podemos definir 3 elementos que deberán transmitirse entre estas partes. El fichero `gameTypes.h` contiene los tipos utilizados para el desarrollo de la práctica. Seguidamente se describen los más relevantes.

- **Código**: Número que indica el estado de la partida, a través del cual, el cliente conocerá el turno del jugador y cuándo acaba la partida. Se recomienda utilizar el tipo (`unsigned int`).
  - `TURN_MOVE`: Turno del jugador, realizar movimiento.
  - `TURN_WAIT`: Jugador debe esperar porque le toca mover a un rival.
  - `GAMEOVER_WIN`: Jugador gana, fin de partida.
  - `GAMEOVER_DRAW`: Empate, fin de partida.
  - `GAMEOVER_LOSE`: Jugador pierde, fin de partida.

- **Mensaje**: Cadena de texto que contiene mensajes sobre el estado de la partida. Este mensaje se enviará en dos partes. La primera consiste en un número de 4 bytes (`unsigned int`) que contiene la longitud (en número de caracteres) del mensaje a enviar. La segunda parte consiste en el propio mensaje (`tString`), el cual está formado por una secuencia de caracteres.

- **Tablero** (`tBoard`): Tablero con el estado de la partida que consiste en un array de caracteres. Las casillas vacías se representan con el carácter `EMPTY_CELL`. Además, `PLAYER_1_CHIP`, `PLAYER_2_CHIP` y `PLAYER_3_CHIP` representan los caracteres para mostrar las fichas del jugador 1, jugador 2 y jugador 3, respectivamente.

```
typedef char tBoard [BOARD_WIDTH * BOARD_HEIGHT];
```

El fichero `game.h` contiene las cabeceras de las funciones encargadas de la lógica del juego. Además, este fichero contiene una descripción detallada de los parámetros de entrada y salida de cada función. Seguidamente, se describen estas funciones:

```
void initBoard (tBoard board);
```

Esta función inicializa un tablero. Debe ser invocada antes de comenzar una partida.

```
void printBoard (tBoard board, char* message);
```

Imprime el tablero por pantalla y su mensaje asociado al estado de la partida.

```
tMove insertChip (tBoard board, tPlayer player, unsigned int column);
```

Introduce una ficha en la columna indicada por el jugador, actualizando el tablero en caso de haber realizado un movimiento válido.

```
int checkWinner (tBoard board, tPlayer player);
```

Comprueba si el jugador `player` es el ganador. Esta función deberá invocarse después de haber realizado un movimiento válido.

```
int isBoardFull (tBoard board);
```

Comprueba si el tablero está lleno de fichas, de forma que, en caso afirmativo, no es posible realizar más movimientos.

```
void showError(const char *msg);
```

Muestra un error por pantalla y finaliza la ejecución. Esta función debe invocase cuando se produzca un error no recuperable.

1.1 Implementación del servidor

El servidor se ejecutará recibiendo como parámetro, únicamente, el puerto donde realizará la escucha de las conexiones de los clientes.

Seguidamente, establecerá conexión con tres clientes (jugadores), asignando un socket a cada una de estas conexiones. A partir de ese momento, el servidor ya podrá iniciar una partida con estos tres jugadores.

El fichero `serverGame.h` contendrá las cabeceras de las funciones que deberán implementarse en el fichero `serverGame.c`. Generalmente, el objetivo de estas funciones es simplificar el código de la parte servidor. Por ejemplo, funciones que se encargan de enviar/recibir un mensaje. Puesto que los mensajes se transmiten de manera frecuente con los jugadores, estas funciones ayudan a reducir el código fuente y aumentan la claridad del programa.

El servidor deberá ser capaz de mantener varias partidas simultáneamente. Para ello, será necesario el uso de *threads*. La estructura `tThreadArgs`, definida en el fichero `gameTypes.h`, puede utilizarse para gestionar la comunicación de cada partida. Con el fin de simplificar el desarrollo de esta parte, no será necesario limitar el número de conexiones con los clientes.

Una vez finalice cada partida, el servidor deberá escribir al final del fichero `SCORES_FILE_NAME` (definido en `gameTypes.h`), el resultado de la misma. Así, se indicará en una línea quién ha ganado, quién ha perdido, o si la partida ha acabado en tablas. En esta línea debe aparecer el nombre de los 3 jugadores.

Finalmente, el servidor gestionará el *Top 3*, es decir, un array con los tres mejores jugadores. Se puede considerar que los nombres de los jugadores son únicos y no se repiten. Cada vez que un jugador gane una partida, obtendrá un punto. Esta lista se inicializará cuando arranque el servidor y se imprimirá su contenido por pantalla después de cada partida.

1.2 Implementación del cliente

La parte cliente se ejecuta recibiendo dos parámetros: la IP del servidor y el puerto donde éste permanece escuchando conexiones de los jugadores.

Cada cliente realizará una única conexión con el servidor, la cual se deberá cerrar una vez finalice la partida. Además, el cliente no controlará en ningún momento el estado del tablero, de esta forma, la lógica del juego se gestionará completamente en el servidor. Por ejemplo, aspectos tales como comprobar si la columna introducida por el jugador está llena o si ésta no pertenece al intervalo [0-`BOARD-WIDTH`) se controlarán en el servidor.

El fichero `clientGame.h` contendrá las cabeceras de las funciones utilizadas por la parte cliente. El fichero `clientGame.c` deberá contener la implementación de estas funciones. Por ejemplo, la función *readMove* se proporciona ya implementada. Esta función se encarga de leer el movimiento de un jugador.

1.3 Estructura de una partida

Una partida entre tres jugadores se compone de dos fases. La primera se corresponde con el intercambio de nombres, mientras que la segunda se corresponde con el desarrollo del juego.

En la primera fase, el cliente deberá pedir al jugador que introduzca el nombre por teclado. Para ello se podrá hacer uso, por ejemplo, de la función `fgets`. Seguidamente, se envía al servidor el nombre del jugador. Una vez enviado, se debe recibir, por separado, el nombre de sus dos rivales.

Cada mensaje, tal y como se comentó en la sección 1, se compone de un número de 4 bytes (`unsigned int`) y una cadena de caracteres que contiene el texto con información sobre el estado de la partida.

Las siguientes capturas de pantalla detallan la información mostrada, al ejecutar la primera fase de la partida, en cada una de las partes.

Servidor

```
Albertos-MacBook-Pro:PSD_Prac1_solucionSocketsMultithread cana$ ./serverGame 10000
Player 1 is connected!
Player 2 is connected!
Player 3 is connected!
Name of player 1 received: Spiderman
Name of player 2 received: Magneto
Name of player 3 received: Homer
```

Jugador 1

```
Albertos-MacBook-Pro:PSD_Prac1_solucionSocketsMultithread cana$ ./clientGame 192.168.1.135 10000
Connection established with server!
Enter player name:Spiderman
You are playing against Magneto and Homer
Game starts!
```

Jugador 2

```
Albertos-MacBook-Pro:PSD_Prac1_solucionSocketsMultithread cana$ ./clientGame 192.168.1.135 10000
Connection established with server!
Enter player name:Magneto
You are playing against Spiderman and Homer
Game starts!
```

Jugador 3

```
Albertos-MacBook-Pro:PSD_Prac1_solucionSocketsMultithread cana$ ./clientGame 192.168.1.135 10000
Connection established with server!
Enter player name:Homer
You are playing against Spiderman and Magneto
Game starts!
```

Para establecer el orden de los jugadores en una partida se tendrá en cuenta su orden de llegada. Así, el primer jugador que realice la conexión con el servidor será quien comience la partida. A partir de este punto, **cada vez que el servidor envía información al cliente, esta información estará formada por**:

- Código: `unsigned int`.
- Mensaje: `unsigned int` indicando la longitud del mensaje y el propio mensaje.
- Tablero: `tBoard`.

Sin embargo, el cliente sólo enviará el movimiento realizado a través de un número de 4 bytes (`unsigned int`).

Por ejemplo, cuando se inicia la partida, el primer jugador recibe la siguiente información del servidor:

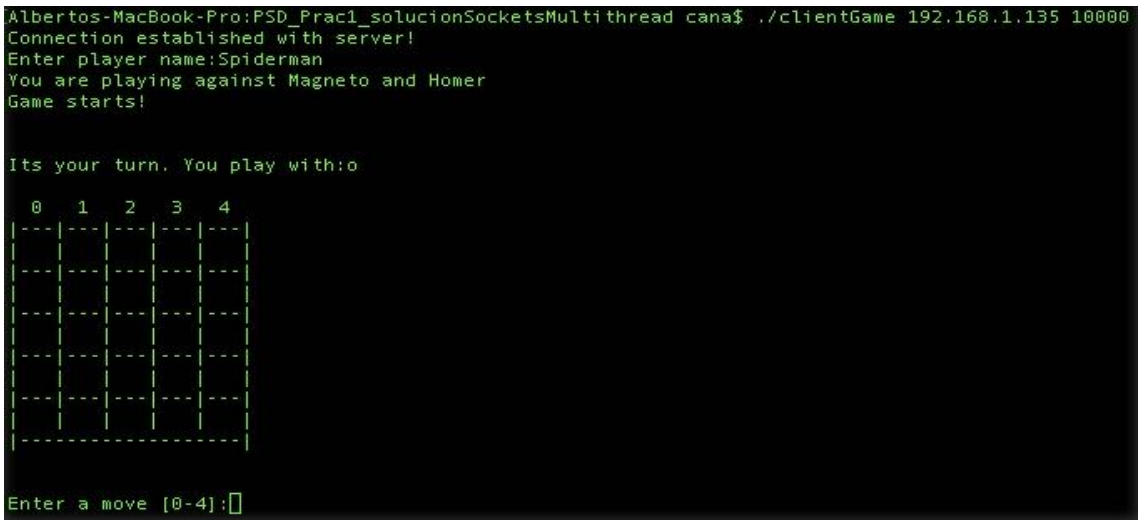
- Code: `TURN_MOVE`
- Msg: 30 – Its your turn. You play with:o
- Board: (tablero vacío)

Mientras que el jugador 2 recibe:

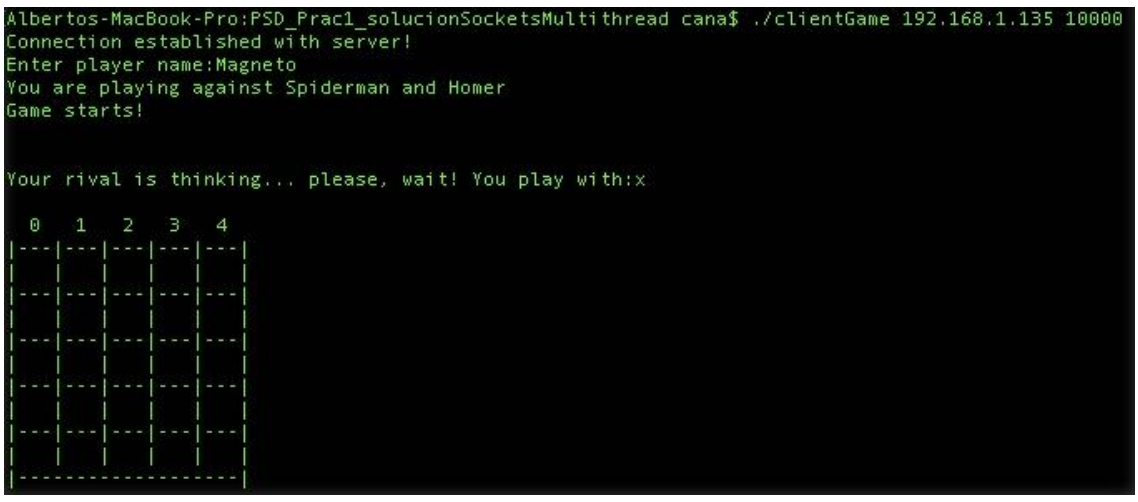
- Code: `TURN_WAIT`
- Msg: 55 – Your rival is thinking... please, wait! You play with:x
- Board: (tablero vacío)

Una vez el servidor envía la información inicial a los jugadores, cada pantalla muestra lo siguiente:

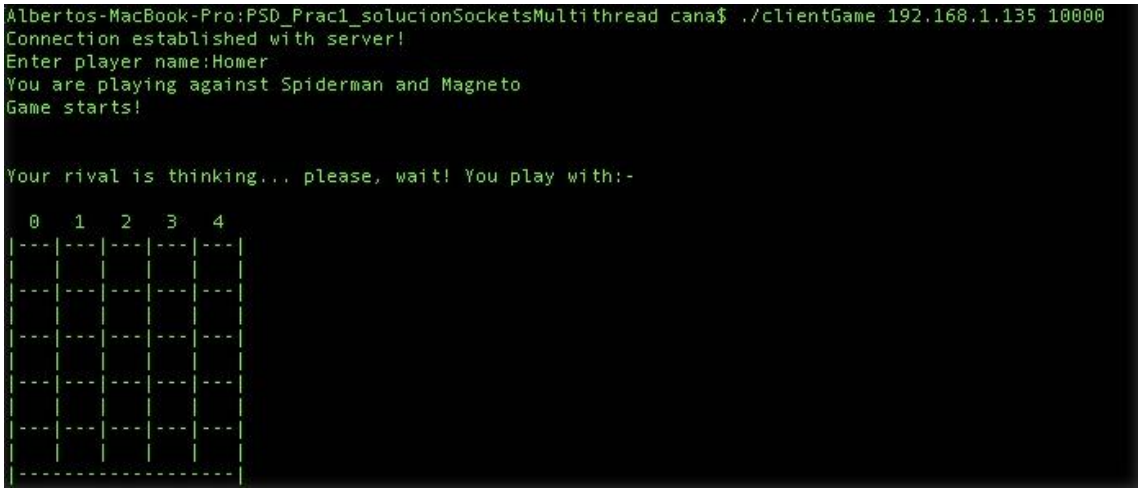
Jugador 1



Jugador 2



Jugador 3



En este punto, tal y como puede apreciarse en las capturas de pantalla, *jugador 1* tiene el turno para realizar su movimiento. Es importante destacar que una vez el jugador ha realizado un movimiento válido, el servidor envía información a los tres jugadores, ya que **el tablero se actualiza en el servidor, no en el cliente**. De esta forma, por cada movimiento válido realizado, el servidor envía (código + mensaje + tablero) a cada cliente. Sin embargo, si un jugador realiza un movimiento no válido, el servidor lo comunica únicamente al jugador con el turno, de forma que se repite la comunicación hasta que se realice un movimiento válido.

La parte cliente no necesita manipular, ni debe, el tablero. Una vez reciba la información del servidor, deberá invocar la función

```
void printBoard (tBoard board, char* message);
```

para que se imprima por pantalla tanto el estado actual del tablero como el mensaje con la información de la partida.

Una vez procesado el movimiento por el servidor, se realizan las siguientes comprobaciones:

- Si el movimiento es correcto
  - ¿Es fin de partida?
    - Si gana el jugador actual, se transmite el estado de la partida a los 3 jugadores, indicando quién gana y quién pierde.
    - Si hay empate, se transmite el estado de la partida a los 3 jugadores, indicando que todos empatan.
  - ¿No es fin de partida?
    - Se cambia el turno del jugador actual.
    - Se envía la información correspondiente **a cada jugador**.
    - Se espera el movimiento del jugador actual.
- Si el movimiento no es correcto (columna llena o fuera de rango):
  - Se pide al jugador actual que realice un nuevo movimiento.

2.- Ficheros a entregar

Los ficheros necesarios para la realización de esta práctica se encuentran en el fichero `PSD_Prac1_Sockets.zip`. Se deberá entregar, además, un fichero llamado `autores.txt` que contenga el nombre completo de los integrantes del grupo.

La entrega de esta práctica se llevará a cabo mediante un **único fichero comprimido** en formato `zip`. Es importante matizar que la práctica entregada debe contener los ficheros necesarios para realizar la compilación, tanto del cliente como del servidor.

En caso de que cualquiera de las partes entregadas no compile, se tendrá en cuenta la penalización correspondiente.

NO se permite incluir nuevos ficheros para el desarrollo de este apartado.

3.- Plazo de entrega

La práctica debe entregarse a través del Campus Virtual **antes del día 13 de Octubre de 2022, a las 15:55 horas**.

No se recogerá ninguna práctica que no haya sido enviada a través de la plataforma indicada o esté entregada fuera del plazo indicado.