laSalle
UNIVERSITAT RAMON LLULL

Sistemas Operativos 2018-2019

Sesión 05: Sockets (II) 26/10/18

Objetivo

En sesión anterior hemos visto formas de comunicar distintos procesos vía sockets y nos hemos centrado en el proceso cliente.

El objetivo de esta sesión es llevar a cabo una primera introducción a los *sockets* en el servidor, un mecanismo de comunicación entre procesos que pueden estar ejecutándose en el mismo o en distintos sistemas.

Motivación

Más concretamente, con esta sesión, el alumno ha de ejercitar:

- Creación de sockets (*socket*)
- Asociación (*bind*)
- Establecimiento de conexiones (*connect*)
- Recepción de conexiones (*listen*)
- Aceptación de conexiones (*accept*)
- Cierre de la conexión (*close*)
- Paso de parámetros a un ejecutable (*argc*, *argv*)

Documentación previa

Para completar esta sesión se recomienda la lectura de las siguientes referencias:

> SALVADOR J., CANALETA X. (2014). *Programació en UNIX per a pràctiques de Sistemes Operatius*, Publicacions d'Enginyeria i Arquitectura La Salle (Edició PDF)

Batalla naval

Batalla naval es un juego de 2 jugadores en que cada uno de ellos distribuye un conjunto de barcos de guerra en un tablero que es visible únicamente para él. Cada jugador deberá acertar las posiciones en que el otro jugador ha colocado sus barcos, tocándolo hasta hundirlo. El jugador que hunda todos los barcos del oponente será el ganador.

En esta sesión hay que implementar un programa servidor que permita jugar partidas de Batalla naval contra el cliente. Cada programa permitirá a un usuario jugar contra el otro, introduciendo para ambos casos posiciones por teclado donde atacar del tablero del adversario. Es decir, el juego es Jugador vs Jugador, y no Jugador vs Máquina.

El tablero

Para poder jugar, se debe introducir una distribución de los barcos, que vendrá dada por un fichero de texto de 4 filas y 5 columnas. Habrá un total de 4 barcos con las siguientes dimensiones:

- 1 barco de 1 única casilla.
- 2 barcos de 2 casillas.
- 1 barco de 3 casillas.

Los barcos se pueden distribuir vertical y horizontalmente, pero no pueden tocarse entre ellos (debe haber 1 casilla de separación).

Un posible fichero de distribución sería el siguiente:

1	0	0	0	0
0	0	1	1	1
1	0	0	0	0
1	0	1	1	0

La partida

Una vez establecida la conexión entre el programa servidor y cliente, ambos se enviarán el nombre de su jugador. Primero, el cliente enviará el nombre de su jugador al servidor y después el programa servidor enviará el nombre de su jugador al programa cliente.

El jugador del programa cliente siempre empezará la partida.

En el turno de cada jugador, este enviará dos coordenadas, primero la fila y luego la columna donde cree que hay un barco. El envío se realiza número a número, pudiendo recibir como respuesta del otro jugador las siguientes palabras:

- "[Agua]": El jugador ha fallado, el turno pasa a ser del oponente.
- "[Tocado]": El jugador ha acertado, le toca volver a escoger dos coordenadas.
- "[KO]": El jugador ha hundido el último barco del oponente, gana el juego y se acaba la partida (el programa) cerrando la comunicación.
- En caso de recibir un envío incorrecto se contesta con un mensaje de error.

Ejemplo de ejecución

Los mensajes que empiezan con (Cliente) se muestran en el terminal donde se ha ejecutado el programa cliente, y los mensajes que empiezan con (Server) se muestran en el termina donde se ha ejecutado el programa servidor.

```
./server Pernía 127.0.0.1 8500 board.txt
(Server) Jugador Pernía listo.
(Server) Esperando un adversario...

./cliente Canaleta 127.0.0.1 8500 board.txt
(Server): Adversario conectado.
(Server): Jugando contra Canaleta.
(Cliente): Conexión establecida con el adversario.
(Cliente): Jugando contra Pernía.
(Cliente): Introduce la fila: 1
(Cliente): Introduce la columna: 1
```

```
(Server): Agua (1,1)
(Cliente): Agua...
(Cliente): Turno de Pernía.
(Server): Me toca.
(Server): Introduce la fila: 0
(Server): Introduce la columna: 1
(Cliente): Tocado (0,1). [7/8]
(Server): Tocado!
(Server): Introduce la fila: 2
(Server): Introduce la columna: 2
(Cliente): Tocado (2,2). [6/8]
(Server): Tocado!
(Server): Introduce la fila: 1
(Server): Introduce la columna: 2
(Cliente): Agua (1,2)
(Server): Agua...
(Server): Turno de Canaleta.
(Cliente): Introduce la fila: 4
(Cliente): Introduce la columna: 5
(Server): Tocado (4,5) [7/8].
(Cliente): Tocado!
(Cliente): Introduce la fila: 4
(Cliente): Introduce la columna: 4
(Server): Tocado (4,5) [6/8].
(Cliente): Tocado!
.
.
.
(Server): Introduce la fila: 2
(Server): Introduce la columna: 5
(Cliente): Tocado (2,5) [0/8].
(Cliente): He perdido... Le debo una birra a Pernía.
(Server): Victoria contra Canaleta! Ha sido fácil.
(Server): Esperando un adversario...
.
.       // otra partida
.
(Servidor): Esperando un adversario...
```

Consideraciones

- Se puede asumir que el programa servidor se ejecutará antes que el programa cliente.
- Se puede asumir que el formato de los parámetros de entrada siempre será correcto.
- El servidor deberá permitir que haya conexiones en espera mientras existe una conexión gestionándose.
- El formato del fichero de entrada no contendrá errores.
- No está permitido el uso de funciones “system” o “popen” o análogas de la misma familia.
- No está permitido el uso de variables globales. Excepto aquellas necesarias para el control de signals.
- Toda entrada y salida se ha de realizar con file descriptors no está permitido el uso de printf, scanf, FILE*, getchar, o similares.
- Se ha de compilar utilizando los flags –Wall y –Wextra.
- Cualquier práctica que contenga warnings será no apta.
- Se han de liberar todos los recursos, en caso contrario la práctica será no apta.
- En la primera línea de los .c debe poner comentados los nombres y logins de los miembros del grupo. En caso contrario la práctica no se corregirá.
- Las conexiones tienen que quedar lo mejor cerradas posibles, contemplando también cierres inesperados. No se aceptará que queden colgadas.