Sistemas Operativos y Redes (E0224)
Año 2021

Trabajo Práctico Integrador

Grupo N°4:

Ignacio Hamann - 68410/3
Juan Pablo Elisei - 68380/5
Tomás Tavella - 68371/4

Resumen

En este informe se desarrollan los detalles de diseño e implementación de un juego de naipes españoles "Escoba de 15" en el lenguaje de programación C (y bibliotecas estándar de Linux). Con este fin, se utiliza un servidor concurrente a base de *sockets* de red que acepta hasta 4 conexiones (jugadores) simultáneos conectados bajo el protocolo TCP.

Facultad de Ingeniería
Universidad Nacional de La Plata

Índice

1. Enunciado

Se desea realizar un servidor TCP que permita jugar una partida del tradicional juego de cartas "Escoba de 15", desde un cliente `telnet`.

El servidor debe esperar conexiones entrantes desde el port 1234 y deberá incorporar, como mínimo, las siguientes capacidades:

a) Permitir jugar una partida entre 2, 3 o 4 jugadores.

b) Cuando se conecte el primer jugador, se deberá ofrecer la posibilidad de seleccionar si la partida aceptará 2, 3 o 4 jugadores.

c) Una vez seleccionada la cantidad de jugadores, deberá crear e inicializar los recursos necesarios y esperar que se presenten el resto de los jugadores.

d) Una vez que se hayan conectado el resto de los jugadores, se repartirán las cartas y se avisará al primer jugador conectado que es el que inicia la partida ("mano"). El orden de participación del resto de los jugadores deberá ser el mismo que el orden de conexión.

e) El servidor enviará a cada jugador conectado la información sobre cuales naipes le tocaron en el reparto y además cuales son los naipes que están sobre la mesa.

f) El servidor habilitará al jugador que tiene el turno de juego a enviar su jugada, una vez recibida la reenviará a todos los jugadores.

g) Luego informará a cada jugador cuáles son los naipes que tiene en su poder, cuáles son los naipes que quedan en la mesa y quién tiene el próximo turno.

h) Si un jugador intenta enviar su jugada cunado no le toque el turno, el servidor ignorará el intento.

i) El juego finaliza cuando no hay más cartas para repartir.

j) Cuando finaliza el juego, el servidor informa a todos los jugadores, las cartas recolectadas por cada jugador y la cantidad de escobas para poder realizar un recuento manual de puntaje.

La descripción anterior es general, y puede implementarse de la manera que se desee, teniendo en cuenta que los clientes `telnet` o `nc` mostraran en pantalla solamente lo que reciban sin realizare ningún formateo. Se sugiere que la información a los clientes sea enviada en formato de texto.

1.1. Manejo de errores

Si un jugador intenta levantar un conjunto de cartas que se exceden de 15, se anula la jugada y se le indica que la comience de nuevo. Si intenta levantar dos veces la misma carta, le indica que no es válido y le pide que levante otra carta.

1.2. Otras consideraciones

Este juego tiene muchas variantes, pero en este caso se trata de utilizar las reglas mas sencillas. Tener en cuenta que el interés de la cátedra es que apliquen los conocimientos sobre TCP/IP y comunicaciones entre procesos.

El servidor debe ser un servidor concurrente, donde a medida que se conectan los distintos jugadores. se crea un hijo para atender a cada jugador. Los hijos deben comunicarse entre sí, mediante mecanismos de IPC como memoria compartida, colas de mensajes o semáforos.

2. Interpretación del problema

Para este programa, se debe crear un servidor concurrente que haga uso del protocolo TCP y permita jugar una partida de la escoba de 15.

Dado que no se creo ningún programa para el cliente, los mismos se deben conectar a través de los comandos `netcat` o `telnet`, por lo que el formateo de los mensajes debe realizarse del lado del servidor. Todos los clientes deben ser atendidos por hijos, siendo el primero el que determina la cantidad de jugadores de la partida.

Debe usarse algún método de sincronización o IPC para la comunicación entre hijos y determinación de los turnos de juego, ignorando jugadas no válidas por parte de los demás jugadores.

Al finalizar la partida, el servidor quedará a la espera de nuevos jugadores para iniciar una nueva partida.

3. Resolución

Se utilizaron las siguientes bibliotecas de C para poder llevar a cabo la resolución del problema planteado:

- `<sys/ipc.h>`: biblioteca de *System V* para la comunicación entre procesos.
- `<sys/shm.h>`: biblioteca de *System V* para la memoria compartida.
- `<sys/msg.h>`: implementación de *System V* para colas de mensajes.
- `<sys/socket.h>`: biblioteca de *System V* para utilizar sockets.
- `<sys/wait.h>`: biblioteca de *System V* para ultilar la funcion `wait()`.
- `<netinet/in.h>`: biblioteca que contiene varias definiciones útiles de macros y tipos de datos para manejo de sockets de red.
- `<arpa/inet.h>`: biblioteca que contiene definiciones de varias funciones útiles para el manejo de direcciones IP.

Se decidió guardar los datos de cada jugador (nombre, cantidad y valor de las cartas levantadas, y escobas) y la jugada realizada (cartas levantadas o descartadas) como memoria compartida, para simplificar la comunicación entre procesos, y que no sea necesario mandar tantos datos en los mensajes entre procesos cuando se debe anunciar de quien es el turno o que jugada realizó.

El padre será el encargado de sincronizar la partida, repartir las cartas, y avisarle a los hijos que operación tienen que realizar. Los hijos por su parte esperarán un mensaje del padre una vez estén conectados los jugadores, y anunciarán de quien es el turno, pedirán la jugada o mostraran que jugada se realizó de acuerdo a lo recibido.

Una vez terminada la partida (chequeado porque se repartieron las 40 cartas y no se pueden repartir más), el padre enviará un mensaje de finalización, para que los hijos muestren los puntajes, se desasocien de la memoria compartida y terminen, y quedará a la espera de una nueva conexión para iniciar otra partida.

3.1. Pseudocódigo

A continuación se muestra un pseudocódigo que explica a grandes rasgos el funcionamiento del programa del servidor:

```
1
2     INICIO
3         Declarar y asignar variables, macros, estructuras y funciones;
4         Obtener la clave de las memorias compartidas y colas de mensajes (en el caso de no
      obtenerlas imprimir error);
```

```
    Llamar al sistema para obtener el ID de las memorias compartidas y colas de mensajes (en el caso de que no obtenerlas imprimir error);
    Asociar el espacio de memoria compartida con un puntero (si no puede asociar imprimir error);
    Obtener colas de mensajes (si no puede obtener imprimir error);
    Crear socket para el servidor;
    Crear partida y conexion de jugadores;
    Mientras(El servidor este activo){
        Mientras(No se hayan terminado de conectar los jugadores){
            Esperar a que se conecte un jugador;
            Imprimir desde donde se hizo la conexion;
            Si(Es el primer jugador){
                Solicitar la cantidad de jugadores que van a jugar (si no se ingresa un numero entre 2 y 4 volver a solicitar);
            }
            Si(Se crea un hijo y se esta en el hijo){
                Solicitar ingresar un nombre (si no ingresa se le asigna un nombre por defecto);
                Imprimir mensaje de esperando a los demas jugadores;
                Avisar al padre que el hijo esta listo;
            }
            Sino{
                Aumentar la varible turno para llevar la cuenta de clientes conectados;
                Cerrar conexion no utilizada;
            }
        }
    }
    Si(Estoy en el padre){
        Inicializar la mesa sin cartas;
        Esperar a los jugadores;
        Mientras(Queden cartas por repartir){
            Repartir 3 cartas a cada jugador;
            Si(Es la primera mano){
                Repartir 4 cartas en la mesa;
            }
            Contar cuantas cartas hay repartidas;
            Imprimir el numero de ronda en el servidor;
            Para(Las 3 rondas){
                Para(Los 4 jugadores){
                    Enviar a todos los jugadores sus cartas;
                    Enviar a los jugadores que toca jugar;
                    Si al jugador que le tocaba jugar levanto se imprime que fue el ultimo en levantar;
                    Enviar jugada a los demas jugadores;
                }
            }
        }
        Asignar las cartas sobrantes en mesa al ultimo jugador en levantar;
        Enviar a todos los jugadores que finalizo la partida;
    }
    Sino{
        Inicializar cartas levantadas sin cartas;
        Hacer{
            Si(Se recibe 'A' del padre){
                Contar cartas en la mesa;
                Si(Quedan cartas sobre la mesa){
                    Enviar cuales son las cartas;
                }
                Sino{
                    Enviar que no hay cartas sobre la mesa;
                }
                Si(El jugador tiene cartas en la mano){
                    Enviar cuales son las cartas;
                }
                Sino{
                    Enviar que no tiene cartas en la mano;
                }
                Si(es el turno del jugador){
                    Imprimir que se espera la jugada;
                }
                Sino{
                    Imprimir de quien es el turno;
                }
            }
            Si(Se recibe 'T' del padre){
                Si(Es el turno del jugador){
                    Contar cartas en la mesa;
                    Si(Quedan cartas sobre la mesa){
                        Enviar cuales son las cartas;
                    }
                    Sino{
                        Enviar que no hay cartas sobre la mesa;
                    }
                    Si(El jugador tiene cartas en la mano){
                        Enviar cuales son las cartas;
                    }
                    Sino{
                        Enviar que no tiene cartas en la mano;
                    }
                    Mientras(La jugada no sea valida){
                        Si(Quedan cartas sobre la mesa){
```

```
                                Preguntar si el jugador quiere levantar o descartar;
                            }
                            Sino{
                                Decirle que descarte;
                            }
                            Si(Levanta){
                                Pedir que seleccione una carta de la mano y las
    correspondientes de la mesa;
                                Si(la suma no da 15){
                                    La jugada no es valida;
                                }
                                Sino{
                                    Enviar la jugada al padre;
                                    Contar cartas en la mesa;
                                    Si(No quedan cartas sobre la mesa){
                                        El jugador hizo escoba;
                                    }
                                    La jugada es valida;
                                }
                            }
                            Si(Descarta){
                                Pedir que seleccione una carta de la mano para descartar;
                                Enviar la jugada al padre;
                                La jugada es valida;
                            }
                        }
                    }
                }
                Si(Se recibe 'L' del padre){
                    Enviar la jugada de levante que hizo el jugador anterior;
                    Si (Hizo escoba){
                        Enviar que hizo escoba;
                    }
                }
                Si(Se recibe 'D' del padre){
                    Enviar la jugada de descarte que hizo el jugador anterior;
                }
            }Mientras(La partida no finalizo)
            Para(Las 4 jugadores){
                Enviar las cartas que levanto cada jugador y las escobas;
            }
            Liberar memoria compartida;
            return 0;
        }
        Cierre y eliminacion de memoria compartida y colas de mensajes
    FIN
```

3.2. Compilación y ejecución del programa

Con el código completo, para compilar el programa a un archivo binario ejecutable se llama al comando `gcc` (*GNU Compiler Collection*):

```
$ gcc servidorEscoba.c -o servidorEscoba
```

Con lo que se obtiene el archivo binario ejecutable `servidorEscoba`, el cual se ejecuta desde la terminal de la siguiente manera:

```
$ ./servidorEscoba
```

Luego, se deben ejecutar los clientes utilizando el puerto 1234, ya sea con el comando `netcat` o `telnet`:

```
$ netcat «Direccion IP del servidor» 1234
$ telnet «Direccion IP del servidor» 1234
```

Es importante ejecutar primero el servidor y luego los clientes.

4. Conclusiones

En este trabajo debieron aplicarse la mayoría de los contenidos de esta materia para su resolución, como ser la creación de sub-procesos dentro de un programa y la comunicación entre ellos, la creación y utilización de espacios de memoria compartida para transferir datos entre los distintos procesos, y la comunicación entre sistemas mediante sockets TCP y conexiones cliente-servidor, además de la atención de varios clientes en simultáneo mediante un servidor concurrente.