Sistemas Operativos

Trabajo Práctico Final

Programación de Sockets

Integrantes:

55099 - Matías Ota

55376 - Tobías Matorras Bratsche

55746 - Facundo González Fernández

Fecha de entrega:

15/02/2018

Indice:

Objetivos:

El objetivo del trabajo consistió en la implementación de un sistema cliente/servidor para la reserva de asientos en una sala de cine. Como requerimiento indispensable del trabajo es la utilización de sockets, pero además agregamos lo siguiente:

- Creación y manejo de un nuevo proceso por medio de pipes.
- Creación y manejo de threads.
- Utilización de semáforos.

Compilación y ejecución:

Para la correcta compilación es necesario instalar las dependencias de sqlite3 que en distros Linux basadas en *Debian* serian por el comando

```
apt-get install libsqlite3-dev
```

Una vez resueltas las dependencias se deben ejecutar dentro del directorio raíz los siguientes comandos (cmake minima version necesaria 3.5):

```
cmake .
make
```

Ejecutamos el server sin ningun argumento (opcionalmente se puede ingresar como primer argumento el puerto TCP a escuchar).

```
./server
```

y en otra terminal ejecutamos el cliente (opcionalmente se puede ingresar como primer argumento el host y segundo argumento el puerto):

```
./client
```

El cliente se puede ingresar a dos modos: el modo usuario y el modo administrador:

Al comenzar en modo usuario, se deberá ingresar un nombre a modo de inicio de sesion. Una vez ingresado, podrá elegir entre las siguientes opciones:

```
1.Buy a ticket              (comprar un ticket)
2.View tickets              (ver tickets)
3.Cancel a reservation      (cancelar una reserva)
4.Exit                      (salir)
```

Si se ingresa la opción de comprar un ticket, se imprimira en pantalla las distintas películas disponibles que tiene el usuario para elegir. Una vez seleccionada, se presentan la funciones que tiene el film seleccionado. La función muestra el dia de la semana y la sala donde se proyectara. Finalmente, se muestran los asientos libres de la sala para que el usuario elija fila y columna. Una vez terminado, se realiza la reserva y se imprime el ticket.
Seleccionando la opción de ver tickets, el programa presentará en pantalla los tickets de las funciones reservadas por el usuario.
Para cancelar una reserva, se ingresa la opción correspondiente, y se imprimen los tickets de las funciones reservadas. El usuario deberá elegir aquella que quiera cancelar para que el programa termine la operación.
Por último, la opción de salir permite regresar a la sección anterior.

Si se selecciona el modo administrador el programa presentará las siguientes opciones:

```
1.Add showcase              (agregar una función)
2.Remove showcase           (remover una función)
3.Exit                      (salir)
```

Si se ingresa la opción de agregar una función, el programa le pedirá al usuario que ingrese los siguientes datos: Nombre de la pelicula, número de día de la semana (Siendo Domingo = 0 - Sábado = 6), número de sala. Una vez ingresado los datos, el programa creará la función correspondiente.
En caso de que se desee remover una función, en primer lugar se deberá ingresar el nombre de la película de la función. Una vez hecho esto, se imprimirán las funciones que contienen a dicha pelicula. El usuario seleccionará de estas funciones la que desee eliminar.
Por último, la opción de salir permite regresar a la sección anterior.

Funcionamiento:

El trabajo está compuesto de 3 ejecutables que son: *database*, *server* y *client*.

Server:

El server al iniciar crea un nuevo proceso mediante *fork()* en el que ejecuta al programa database usando *execve()* y utiliza *pipes* para el manejo de la entrada estándar y la salida estándar, una vez iniciado este el servidor comienza a escuchar por defecto en el puerto TCP número 12345. Al conectarse un nuevo cliente genera un nuevo thread que se encarga exclusivamente a la atención del cliente que envía a los pipes del proceso de database de forma totalmente transparente, para evitar que dos clientes accedan a las pipes de la database se utiliza un semáforo obligando que solo un cliente acceda a los pipes al mismo tiempo.

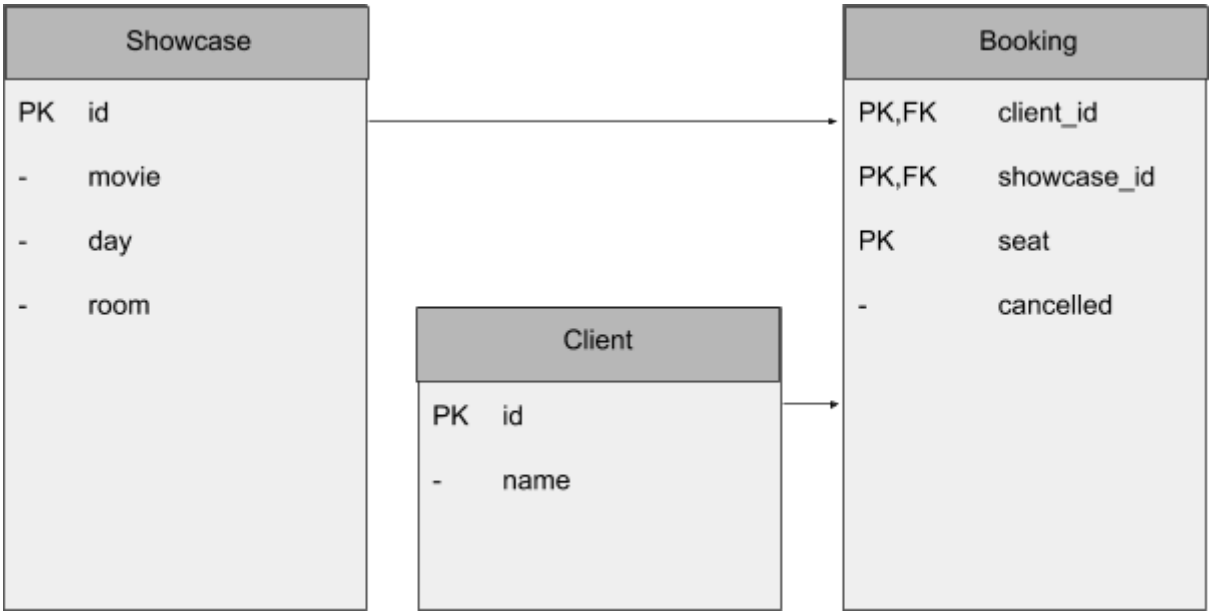
Client:

Al iniciarse el cliente, establece una conexión con el servidor utilizando un socket TCP que tiene por defecto el puerto 12345. De esta forma se comunicará con el servidor para que este le pueda brindar información y realizar operaciones dentro de la base de datos.

Database:

El programa database se encarga del manejo de la base de datos sqlite3 que se almacena en el archivo *cinema.db*.

El diseño del la base de datos es la siguiente:



Dentro de la tabla *Showcase* se encuentran los campos *id*, que funciona como clave primaria de cada tupla, *movie*, correspondiente a un text con el nombre de la película, luego se encuentra *day* que es un integer que hace referencia al día de la semana, y por último *room* con un integer para el número de sala.
La estructura de la tabla *client* es muy simple ya que no se utiliza un sistema de inicio de sesion con contraseña, compuesto solamente por el *id* y el *name* que es un text.
Y por último, la tabla *booking* contiene dos claves foráneas correspondientes al showcase y al cliente que realizó la reserva. Estas dos y el número de asiento forman la clave primaria. Y finalmente, el campo de *cancelled* da la pauta de si la reserva se encuentra o no cancelada. Esto último lo hace utilizando un integer.
La tabla de *booking* almacena aquellos asientos que hayan sido reservados, y de ser cancelados simplemente altera el valor de cancelled dentro de la tupla. Como cada sala tiene la misma cantidad de asientos totales vacíos en el momento de la creación de un showcase, no hace falta almacenarlos dentro de la tabla. Para calcular qué asientos se encuentran disponibles se buscan los que están reservados y se resta del total.

Protocolo:

La interacción con el programa *database* se realiza por medio de entrada estándar y salida estándar. Para facilitar la comunicación entre los procesos se definió un simple protocolo orientado a texto inspirado en el protocolo POP3 de mails. Los comandos se reciben por entrada estándar con el siguiente formato:

```
<número del comando>\n
<argumento 1>\n
<argumento 2>\n
...
<argumento n>\n
.\n
```

La respuesta del programa saldrá por salida estándar con el siguiente formato:

```
<estado de la respuesta>\n
<respuesta 1>\n
<respuesta 2>\n
...
<respuesta n>\n
.\n
```

En ambos casos la secuencia de caracteres: **\n.\n** determina el final del mensaje. Tanto el número del comando como el estado de las respuesta son enteros positivos.

Los números de los comandos están definidos con los siguientes argumentos y sus correspondientes respuestas:

0) *ADD_CLIENT*. Agrega un cliente.
Argumentos:
    primer argumento: nombre del cliente
Respuesta:
    retorna siempre OK
1) *ADD_SHOWCASE*. Agrega una función.
Argumentos:
    primer argumento: nombre de la pelicula
    segundo argumento: el día de la semana (valor numérico)
    tercer argumento: número de la sala
2) *REMOVE_SHOWCASE*. Remueve una función.
Argumentos:

primer argumento: nombre de la pelicula
segundo argumento: el dia de la semana (valor numérico)
tercer argumento: numero de la sala

3) *GET_MOVIES*. Obtiene el listado de peliculas.
Respuesta:
valor de respuesta N: nombre de la pelicula N

4)*GET_SHOWCASES*. Obtiene el listado de funciones.
Argumentos:
primer argumento: nombre de la pelicula
Respuesta:
primer valor de respuesta: nombre de la pelicula
segundo valor de respuesta: dia
tercer valor de respuesta: número de la sala

5) *GET_SEATS*. Obtiene los asientos de la sala.
Argumentos:
primer argumento: nombre de la pelicula
segundo argumento: el dia de la semana (valor numérico)
tercer argumento: numero de la sala
Respuesta:
valor de la respuesta n: 0-1 (0: en que caso de estar ocupado el asiento n, 1: de estar libre el)

6) *ADD_BOOKING*. Realiza una reserva.
Argumentos:
primer argumento: nombre del cliente
segundo argumento: nombre de la pelicula
tercer argumento: el dia de la semana (valor numérico)
cuarto argumento: numero de la sala

7)*REMOVE_BOOKING*. Cancela una reserva.
Argumentos:
primer argumento: nombre del cliente
segundo argumento: nombre de la pelicula
tercer argumento: el dia de la semana (valor numérico)
cuarto argumento: numero de la sala

8) *GET_BOOKING*. Obtiene las reservas.
Argumentos:
primer argumento: nombre del cliente

Respuesta:

primer argumento: nombre de la pelicula
segundo argumento: el dia de la semana (valor numérico)
tercer argumento: numero de la sala
cuarto argumento: numero de asiento

9) *GET_CANCELLED*. Obtiene las reservas canceladas.

Argumentos:

primer argumento: nombre del cliente

Respuesta:

primer argumento: nombre de la pelicula
segundo argumento: el dia de la semana (valor numérico)
tercer argumento: numero de la sala
cuarto argumento: numero de asiento

Los numeracion de los estados de respuesta con una breve descripción:

0) *RESPONSE_OK* sin errores
1) *RESPONSE_ERR* ocurrió un error sin definir
2) *ALREADY_EXIST* para comandos add, en caso que ya exista
3) *FAIL_TO_OPEN* no se pudo abrir el archivo de la base de datos
4) *FAIL_QUERY* fallo la consulta a la base de datos
5) *BAD_BOOKING* la reserva no existe
6) *BAD_CLIENT* el cliente no existe
7) *BAD_SHOWCASE* no existe la sala o película

Bibliografía

Kerrisk, M.,(2010),*Linux Programming Interface,*San Francisco Estados Unidos ,No Starch Press
SQLite Documentation,SQLite.https://www.sqlite.org//docs.html
Stackoverflow questions “Pipe in C..”, stackoverflow. http://bit.ly/2Hdm5pD