Entradas de Cine

Ejercicio N°3

Objetivos	● Diseño y construcción de sistemas con acceso distribuido ● Encapsulación de Threads y Sockets en Clases ● Definición de protocolos de comunicación ● Protección de los recursos compartidos ● Uso de buenas prácticas de programación en C++
Instancias de Entrega	**Entrega 1:** clase 8 (09/10/2018). **Entrega 2:** clase 10 (23/10/2018).
Temas de Repaso	● Definición de clases en C++ ● Contenedores de STL ● Excepciones / RAII ● Move Semantics ● Sockets ● Threads
Criterios de Evaluación	● Criterios de ejercicios anteriores ● Eficiencia del protocolo de comunicaciones definido ● Control de paquetes completos en el envío y recepción por Sockets ● Atención de varios clientes de forma simultánea ● Eliminación de clientes desconectados de forma controlada

Índice

Introducción **2**

Servidor **2**
- Formato de Línea de Comandos 2
- Códigos de Retorno 2
- Salida Estándar de Errores 3

Formato de los Archivos de Entrada **3**
- Archivo de Salas 3
- Archivo de Películas 4
- Archivo de Funciones 4

Cliente **5**
- Formato de Línea de Comandos 5
- Códigos de Retorno 5
- Salida Estándar de Errores 5

Comandos **6**
- Listar Películas por Idioma 6
- Listar Películas por Edad 6
- Listar Películas por Género 7
- Listar Funciones Disponibles para un Día 7
- Ver Asientos de una Función 8
- Reservar Asiento 8

Entrada y Salida Estándar **9**

Ejemplos de Ejecución **9**
- Ejemplo de Consultas 10
- Ejemplo de Funciones 10
- Ejemplo de Reservas 11

Recomendaciones **12**

Restricciones **12**

Referencias **12**

Introducción

Se deberá desarrollar una aplicación distribuida con una arquitectura cliente-servidor que permita a los usuarios de las aplicaciones cliente reservar asientos en un cine.

Se permitirá consultar por las películas en cartelera por género, idioma, restricciones de edad, y las funciones disponibles en una fecha dada.

Además, el sistema debe permitir al usuario visualizar qué asientos están ocupados en una determinada función y reservar un asiento dado, siempre que no esté reservado previamente.

Servidor

El servidor deberá mantener la información de las salas que tiene el cine, las películas que están en cartelera, y los horarios de las funciones de cada película. Los administradores del cine serán encargados de editar la información sobre películas, salas y funciones disponibles, y hoy en día utilizan una planilla de cálculo para manejar esa información.

Esta información es fácil de exportar a archivos en formato CSV (*Comma Separated Value*), y por ende estará disponible en tres archivos de entrada con dicho formato. Los datos deberán ser cargados en el sistema al principio de la corrida del servidor.

Formato de Línea de Comandos

El servidor recibirá el puerto en el que debe escuchar por nuevas conexiones como primer parámetro, y los archivos CSV como los parámetros siguientes:

```
./server <puerto> <salas.csv> <peliculas.csv> <funciones.csv>
```

Todos los parámetros son obligatorios.

Códigos de Retorno

El proceso servidor retornará:

- **0,** en caso de una ejecución sin errores.
- **1,** en caso de un error en los parámetros de entrada, que pueden ser:
    - Algún archivo inexistente.
    - Cantidad incorrecta de parámetros.
- **2,** en caso de un error dentro de los archivos de entrada:
    - Una función hace referencia a una sala que no existe.
    - Una función hace referencia a una película que no existe.

Salida Estándar de Errores

En caso de los errores antes descritos, se debe mostrar un mensaje de error pertinente:

- Si un archivo no existe, se debe mostrar la cadena:

```
El archivo <nombre> no existe.
```

- Si es errónea la cantidad de parámetros, se debe mostrar el modo de uso:

```
Uso: ./server <puerto> <salas.csv> <peliculas.csv> <funciones.csv>
```

- Si se hace referencia a una sala que no existe:

```
La sala <Identificador incorrecto> no existe en el sistema.
```

- Si se hace referencia a una película que no existe:

```
La película <Nombre incorrecto> no existe en el sistema.
```

Formato de los Archivos de Entrada

Archivo de Salas

Los archivos de salas tienen formato CSV, y contienen los siguientes campos:

```
<ID de la Sala>,<Tipo de Pantalla>,<Capacidad>
```

- El **ID de la Sala** es una letra mayúscula que la representa.
- El **Tipo de Pantalla** puede ser:
  - 2D
  - 3D
  - 4D
- La **Capacidad** de la sala puede ser:
  - chica (tienen 5 filas y 6 columnas)
  - mediana (tienen 10 filas y 11 columnas)
  - grande (tienen 15 filas y 16 columnas)

Un ejemplo de archivo de salas:

```
A,2D,grande
B,3D,chica
C,2D,mediana
D,4D,chica
```

Archivo de Películas

Los archivos de películas tienen formato CSV, y contienen los siguientes campos:

```
<Título>,<Idioma>,<Restricción de Edad>,<Género>
```

- El **Título** es el de la película, y no habrá dos películas con el mismo título.
- El **Idioma** podrá ser:
  - ESP (Español)
  - SUB (Idioma original con subtítulos en español)
- La **Restricción de Edad** puede ser:
  - ATP (Apta para todo público)
  - +13 (Sólo apta para mayores de 13 años)
  - +18 (Sólo apta para mayores de 18 años)
- El **Género** es uno de los siguientes:
  - Drama
  - Accion
  - Comedia
  - Animacion
  - Terror
  - Suspenso

Un ejemplo de archivo de películas:

```
Megalodon (Castellano),ESP,+13,Suspenso
Megalodon (Subtitulada),SUB,+13,Suspenso
12 Horas para Sobrevivir,SUB,+13,Terror
Hotel Transylvania 14,ESP,ATP,Animacion
```

Archivo de Funciones

Los archivos de funciones tienen formato CSV, y contienen los siguientes campos:

```
<ID de la Sala>,<Título>,<Fecha>,<Hora de Inicio>
```

- El **Título** es el de la película.
- La **Fecha** estará en formato DD/MM/AAAA.

- La **Hora de Inicio** estará en formato hh:mm.

Un ejemplo de archivo de funciones:

```
A,Megalodon (Castellano),13/10/2018,12:00
B,Megalodon (Subtitulada),13/10/2018,12:00
A,12 Horas para Sobrevivir,13/10/2018,17:00
B,Hotel Transylvania 14,13/10/2018,17:00
```

NOTA: Si bien los datos pueden ser incorrectos, se puede asumir que los archivos son CSV válidos, y que no tendrán una cantidad incorrecta de columnas.

Cliente

Formato de Línea de Comandos

La aplicación cliente recibirá por línea de comandos la dirección IP del servidor, y el puerto al cual conectarse:

```
./client <ip-servidor> <puerto-servidor>
```

Códigos de Retorno

El cliente retornará:

- **0:** si hubo una ejecución sin errores.
- **1:** si hay un error en los parámetros.

Salida Estándar de Errores

El único uso contemplado en el alcance del ejercicio para la salida estándar de errores es ante una cantidad errónea de parámetros. En tal caso se debe mostrar el modo de uso:

```
Uso: ./client <ip-servidor> <puerto-servidor>
```

Comandos

Los comandos que debe soportar el servidor son los siguientes:

Listar Películas por Idioma

Si se quieren mostrar las películas filtradas por idioma, se debe ingresar por entrada estándar la siguiente línea:

```
IDIOMA <Idioma>
```

Por ejemplo, si se quiere consultar las películas que se proyectan en español, se debe ingresar:

```
IDIOMA ESP
```

Los idiomas que se deben soportar son los que aparecen en la descripción de los archivos de entrada, y se usarán las mismas abreviaturas.

La salida de este comando será una lista de títulos de películas (sólo los títulos). Por ejemplo, para el ejemplo de archivo de películas de la sección anterior, la salida esperada es:

```
Megalodon (Castellano)
Hotel Transylvania 14
```

Listar Películas por Edad

El formato de línea de entrada es muy similar al anterior:

```
EDAD <Restriccion De Edad>
```

Nuevamente, las opciones de restricción de edad se pueden encontrar en la sección que describe los archivos de entrada. Por ejemplo, para pedir las películas aptas para todo público, tenemos que ingresar:

```
EDAD ATP
```

En este caso, esperaríamos ver como salida:

```
Hotel Transylvania 14
```

Listar Películas por Género

Otro filtro que usaremos será por el género de la película. La línea que deberemos ingresar será:

```
GENERO <Género>
```

A modo de ejemplo, podríamos pedir las películas de terror:

```
GENERO Terror
```

Y la salida esperada debería ser:

```
12 Horas para Sobrevivir
```

Listar Funciones Disponibles para un Día

Este comando lista las funciones e una determinada fecha. Se deben indicar los siguientes parámetros:

```
FECHA <DD/MM/AAAA>
```

Por ejemplo, para listar las funciones del 13 de octubre de este año:

```
FECHA 13/10/2018
```

La salida esperada tiene el siguiente formato:

```
(Id Funcion): <Funcion para "(Título)" en la sala (Id Sala) con fecha (DD/MM/AAAA) - (hh:mm)> [AGOTADA]
```

En el caso de nuestro ejemplo:

```
1: <Funcion para "Megalodon (Castellano)" en la sala A con fecha 13/10/2018 - 12:00>
2: <Funcion para "Megalodon (Subtitulada)" en la sala B con fecha 13/10/2018 - 12:00>
3: <Funcion para "12 Horas para Sobrevivir" en la sala A con fecha 13/10/2018 - 17:00>
4: <Funcion para "Hotel Transylvania 14" en la sala B con fecha 13/10/2018 - 17:00>
```

En caso de que la función 1 esté agotada:

```
1: <Funcion para "Megalodon (Castellano)" en la sala A con fecha 13/10/2018 - 12:00> AGOTADA
2: <Funcion para "Megalodon (Subtitulada)" en la sala B con fecha 13/10/2018 - 12:00>
3: <Funcion para "12 Horas para Sobrevivir" en la sala A con fecha 13/10/2018 - 17:00>
4: <Funcion para "Hotel Transylvania 14" en la sala B con fecha 13/10/2018 - 17:00>
```

Ver Asientos de una Función

También se proveerá un comando para ver cuáles asientos están reservados y cuáles no en una función. El formato deberá ser:

```
ASIENTOS <Id Funcion>
```

La salida esperada es una matriz de caracteres ‘O’ o bien ‘X’ representando si un asiento está disponible o reservado, respectivamente. La cantidad de filas y columnas de la matriz se decide según el tamaño de la sala en la que se proyecta la película según esa función. Por ejemplo, para la entrada:

```
ASIENTOS 2
```

Asumiendo que hay algunos asientos reservados, la salida esperada sería:

```
2: <Funcion para "Megalodon (Subtitulada)" en la sala B con fecha 13/10/2018 - 12:00>
      1     2     3     4     5     6
A     O     O     O     O     O     O
B     O     O     O     X     X     O
C     O     X     X     X     O     O
D     X     X     O     O     O     O
E     O     O     O     O     O     O
```

Notar que cada columna empieza después de un **tab**. Mostrando los tabs y saltos de línea como ‘`\t`‘ y ‘`\n`‘ respectivamente, la salida sería:

```
2: <Funcion para "Megalodon (Subtitulada)" en la sala B con fecha 13/10/2018 - 12:00>\n
\t1\t2\t3\t4\t5\t6\n
A\tO\tO\tO\tO\tO\tO\n
B\tO\tO\tO\tX\tX\tO\n
C\tO\tX\tX\tX\tO\tO\n
D\tX\tX\tO\tO\tO\tO\n
E\tO\tO\tO\tO\tO\tO\n
```

Reservar Asiento

Por último, se necesita un comando para reservar un asiento de una determinada función. Para ello deberemos indicar la siguiente línea:

```
RESERVA <Id Función> <Letra de Fila> <Número de Columna>
```

Por ejemplo:

```
RESERVA 2 A 1
```

En este caso podría haber dos salidas posibles:

1) Si el asiento (A,1) de la función 2 ya estaba reservado previamente, la salida debe ser un error:

```
ERROR: El asiento ya esta reservado
```

2) Si el asiento estaba disponible, se debe reservar el asiento y la salida será:

```
OK
```

Entrada y Salida Estándar

Los comandos se reciben por la entrada estándar del cliente, y su salida se muestra en salida estándar de dicha aplicación.

La entrada estándar del servidor será utilizada para cerrar la aplicación, en caso de recibir un caracter ‘q’. No hay un uso esperado para la salida estándar del servidor.

Ejemplos de Ejecución

Usaremos como archivos de entrada los siguientes:

- Archivo ‘**peliculas.csv**’:

```
1945,SUB,+13,Drama
Historias de Ultratumba (Castellano),ESP,+13,Terror
Historias de Ultratumba (Subtitulada),SUB,+13,Terror
Soledad,ESP,+18,Drama
Megalodon (Castellano),ESP,+13,Comedia
Hotel Transylvania 14,ESP,ATP,Animacion
```

- Archivo ‘**funciones.csv**’:

```
A,Megalodon (Castellano),25/09/2018,18:00
C,Megalodon (Castellano),25/09/2018,18:00
B,1945,25/09/2018,18:00
C,1945,26/09/2018,18:00
A,Historias de Ultratumba (Castellano),26/09/2018,19:00
B,Soledad,26/09/2018,20:00
A,Soledad,26/09/2018,23:00
A,Hotel Transylvania 14,26/09/2018,16:00
C,Hotel Transylvania 14,25/09/2018,14:00
```

- Archivo '**salas.csv**':

```
A,2D,grande
B,3D,mediana
C,4D,chica
```

Ejemplo de Consultas

Si consultamos las películas de Terror, escribiendo en el stdin del cliente:

```
GENERO Terror
```

Obtendremos la salida:

```
Historias de Ultratumba (Castellano)
Historias de Ultratumba (Subtitulada)
```

Pero si pedimos las películas de un género que no existe:

```
GENERO Inexistente
```

Obtendremos la salida:

```
Genero no reconocido
```

Para pedir las películas por los otros criterios:

```
EDAD +13
```

Devuelve:

```
1945
Historias de Ultratumba (Castellano)
Historias de Ultratumba (Subtitulada)
Megalodon (Castellano)
```

Pero una inexistente:

```
EDAD +4
```

Devuelve:

```
Edad no reconocida
```

Ejemplo de Funciones

Para consultar las funciones para una determinada fecha:

```
FECHA 25/09/2018
```

Debería devolver:

```
1: <Funcion para "Megalodon (Castellano)" en la sala A con fecha 25/09/2018 - 18:00>
2: <Funcion para "Megalodon (Castellano)" en la sala C con fecha 25/09/2018 - 18:00>
3: <Funcion para "1945" en la sala B con fecha 25/09/2018 - 18:00>
9: <Funcion para "Hotel Transylvania 14" en la sala C con fecha 25/09/2018 - 14:00>
```

Notar que el número de función está dado por el orden que ocupaba la misma en el archivo de entrada.

Si queremos ver los asientos disponibles para la función 9:

```
ASIENTOS 9
```

Obtendremos:

```
9: <Funcion para "Hotel Transylvania 14" en la sala C con fecha 25/09/2018 - 14:00>
Asientos:
      1     2     3     4     5     6
A     O     O     O     O     O     O
B     O     O     O     O     O     O
C     O     O     O     O     O     O
D     O     O     O     O     O     O
E     O     O     O     O     O     O
```

Notar que la sala es chica, y por eso tiene esa cantidad de filas y columnas.

Ejemplo de Reservas

Si reservamos el asiento C-3 de la función 9:

```
RESERVA 9 C 3
```

Obtendremos:

```
OK
```

Y si consultamos nuevamente los asientos:

```
ASIENTOS 9
```

Obtendremos:

```
9: <Funcion para "Hotel Transylvania 14" en la sala C con fecha 25/09/2018 - 14:00>
Asientos:
      1     2     3     4     5     6
A     O     O     O     O     O     O
B     O     O     O     O     O     O
C     O     O     X     O     O     O
D     O     O     O     O     O     O
E     O     O     O     O     O     O
```

Si intentamos reservar nuevamente el asiento C-3 de la función 9:

```
RESERVA 9 C 3
```

Obtendremos:

```
ERROR: El asiento ya está reservado
```

Recomendaciones

- Tomarse un tiempo para Investigar sobre los distintos contenedores de la STL pensando en el trabajo práctico. El problema se simplifica mucho eligiendo los adecuados.
- Encarar el problema de manera incremental: Por ejemplo, programando primero las partes lógicas y de cómo guardar los datos en memoria; luego implementando la lectura de archivos y agregando un modelo cliente-servidor con un sólo hilo; y por último agregando múltiples hilos para manejar varios clientes al mismo tiempo. No es necesario que el orden sea este, pero es muy recomendable avanzar por partes.
- Incluir en el informe uno o varios diagramas explicativos de cuáles son los hilos que intervienen en el sistema, y cuáles son los recursos a los que acceden de manera concurrente. Como siempre, no hace falta que este diagrama sea UML, sino que lo más importante es que sea fácil de leer. Estos diagramas no sólo son útiles para el corrector, sino que aportan claridad sobre el sistema a quien lo desarrolla.

Restricciones

La siguiente es una lista de restricciones técnicas exigidas por el cliente:

1. El sistema debe desarrollarse en ISO C++11.
2. Está prohibido el uso de variables y funciones globales.
3. Para la comunicación se deben utilizar sockets TCP, y los mismos debe ser bloqueantes.
4. El sistema no puede tener condiciones de carrera sin proteger.
5. El servidor debe ser capaz de aceptar varios clientes de manera concurrente.
6. El uso de memoria dinámica (new/delete) debe estar correctamente fundamentado en el informe, dado que en la mayoría de los casos recomendamos usar el Stack para aprovechar el idiom RAII.
7. El manejo de errores debe hacer uso de excepciones, y está prohibido “arrojar” tipos que no hereden de **std::exception** (por ejemplo: **throw -1;** o **throw “Error”**; son líneas que no pueden aparecer en el código).

Referencias

[1] Comma Separated Values: https://en.wikipedia.org/wiki/Comma-separated_values
[2] Página de manual de endian(3): http://man7.org/linux/man-pages/man3/endian.3.html
[3] Contenedores de la Standard Template Library: http://www.cplusplus.com/reference/stl/