Fundamentos de los Sistemas Operativos
Ficha de entrega de práctica

Grupo de prácticas*: **Miembro 1: Romen Adama Caetano Ramirez**	
Número de la práctica*: 1	**Fecha de entrega*:** 31/03/2023

Descripción del trabajo realizado*

Este código implementa una estructura de datos para una sala de cine.
Formado por sala.h (cabecera), sala.c (API) y test_sala.c (bateria de pruebas).
La biblioteca sala.h contiene las constantes usadas por las funciones en este código.
Este código es un archivo de encabezado o cabecera (header file) llamado "sala.h" que define una serie de funciones y constantes para el manejo de una sala de cine.

La API proporcionada (sala.c) es un conjunto de funciones en lenguaje C para administrar una sala de asientos. La estructura de datos principal de la sala se define como un puntero a una estructura Sala, que a su vez contiene los atributos necesarios para el manejo de los asientos, como su capacidad, el número de asientos libres y ocupados y un puntero a un array que representa los asientos.

- *La función crea_sala() se encarga de crear una nueva sala con la capacidad especificada. Primero comprueba si la capacidad es válida, es decir, si es mayor a 1. Luego, asigna memoria para la estructura Sala y para el array de asientos utilizando la función malloc(). Si no se puede asignar memoria, la función devuelve un código de error. Finalmente, inicializa la capacidad, el número de asientos libres y ocupados, y todos los asientos como libres.*
- *La función elimina_sala() se encarga de liberar la memoria de la sala y los asientos. Primero comprueba si la sala existe. Luego, libera todos los asientos ocupados utilizando la función libera_asiento(). Finalmente, libera la memoria de la sala y asigna el puntero a NULL.*
- *La función reserva_asiento() se encarga de reservar un asiento para un cliente con el ID especificado. Primero comprueba si hay asientos libres en la sala. Luego, busca el primer asiento libre disponible y comprueba si está ocupado por otro cliente y si el cliente ya tiene un asiento reservado. Si todo está correcto, reserva el asiento para el cliente y actualiza el número de asientos libres y ocupados.*
- *La función libera_asiento() se encarga de liberar un asiento reservado. Primero comprueba si la sala existe, si el asiento es válido y si está ocupado. Si todo está correcto, libera el asiento, actualiza el número de asientos libres y ocupados, y devuelve el ID del cliente que había reservado el asiento.*
- *La función estado_asiento toma como entrada un número de asiento y devuelve el identificador del cliente que ha reservado ese asiento o 0 si el asiento está libre. Si el número de asiento no es válido, devuelve un error.*
- *asientos_libres() devuelve el número de asientos que todavía no han sido reservados.*
- *asientos_ocupados() devuelve el número de asientos que ya están reservados.*
- *Capacidad() devuelve el número total de asientos que hay en la sala.*

La API también define una serie de códigos de error para los distintos posibles fallos que pueden ocurrir durante la ejecución del programa, como una capacidad de sala inválida, un asiento ya ocupado o un asiento no válido. Estos códigos se definen como una enumeración llamada sala_error.

El código de test_sala.c es una batería de pruebas para una biblioteca "sala.h/sala.c".
Las pruebas son una serie de funciones que se encargan de verificar que la implementación de la sala es correcta.
El programa contiene varias funciones que realizan diferentes pruebas sobre la implementación de la sala. Cada función de prueba comienza imprimiendo el nombre de la prueba en la consola. Luego se crea una sala con una capacidad determinada, se realizan algunas operaciones (como reservar asientos o liberarlos) y se verifican algunas propiedades de la sala. Al final, se elimina la sala creada. Cada función de prueba utiliza macros "DebeSerCierto" y "DebeSerFalso" para verificar que una determinada condición es verdadera o falsa, respectivamente. Si la condición es falsa, se genera una excepción.

Grupo de prácticas*:
Miembro 1: Romen Adama Caetano Ramirez

Número de la práctica*: 1	**Fecha de entrega*:** 31/03/2023

Ademas como complemento se desarrolló estos tres tests unitarios para una función de gestión de reservas de asientos en una sala.
El primer test, "Estado sala", comprueba si se puede crear una sala con una capacidad determinada y si se puede reservar y liberar asientos. Para ello, reserva tres asientos en la sala y comprueba que estos están efectivamente ocupados, mediante la función "estado_asiento". Luego, libera uno de los asientos y comprueba que está libre, nuevamente utilizando "estado_asiento".
El segundo test, "Sentarse y levantarse", prueba la capacidad de la función "sentarse" para reservar asientos para tres personas distintas y de "levantarse" para liberar los asientos correspondientes a dos de ellas. El tercer test, "Reserva múltiple" intenta reservar cuatro asientos a la vez para cuatro personas distintas, para luego intentar reservar siete asientos, y comprueba si ambas solicitudes son procesadas correctamente.
Cada test es ejecutado dentro de un macro que lo inicia y finaliza, y que imprime en la consola el resultado de la ejecución. En caso de que algún test falle, el resultado será indicado mediante un mensaje de error.

Horas de trabajo invertidas* Miembro 1: 30 horas

Cómo probar el trabajo*

Se ha desarrollado un script que facilita el trabajo.
Únicamente navegar hasta el directorio pract1 una vez descomprimido él .zip y ejecutar bash script.sh
En mi caso:

```
romen_adama@MacBook-Air-de-Romen pract1 % pwd
/Users/romen_adama/Documents/FSO/Practicas/pract1
romen_adama@MacBook-Air-de-Romen pract1 % ls
EjercicioLaboratorio    MEDIA                README.md            fuentes              lib                  script.sh
romen_adama@MacBook-Air-de-Romen pract1 % bash script.sh
Iniciando tests...
********** batería de pruebas para Reserva básica: ********** hecho
********** batería de pruebas para Reserva consecutiva: ********** hecho
********** batería de pruebas para Sala vacía: ********** hecho
********** batería de pruebas para Reserva asiento: ********** hecho
********** batería de pruebas para Libera asiento: ********** hecho
********** batería de pruebas para Estado asiento: ********** hecho
********** batería de pruebas para Reserva, libera y estado asiento: ********** hecho
********** batería de pruebas para Estado sala: ********** hecho
********** batería de pruebas para Sentarse y levantarse: ********** hecho
********** batería de pruebas para Reserva múltiple: ********** hecho
********** batería de pruebas para Reserva imposible: ********** hecho
Batería de test completa.
romen_adama@MacBook-Air-de-Romen pract1 %
```

Para el desarrollo de esta estructura se guio y se codifico la estructura de los directorios bajo la guia del documento: Creación de biliotecas y librerías en C (curso 22-23) Archivo:
Siendo los pasos del script:
gcc -c lib/salas/sala.c -o lib/salas/sala.o
ar -crs lib/libsala.a lib/salas/sala.o
gcc fuentes/test_sala.c -lsala -Llib -o fuentes/test_sala
fuentes/test_sala
Ademas como se comenta en el documento anterior, la estructura del directorio queda de tal forma que:
Directorio Padre (pract1): fuentes/ (output/test_sala/test_sala.c) & lib/ (cabecera-{sala.h}/salas-{sala.c/sala.o}/libsalas.a)

Incidencias
Esta es la segunda versión del código ya que la primera malinterprete el código y el formato estricto que se solicitaba.

Comentarios
Comentar que el trabajo fue realizado en el IDE Visual Studio Code, y ejecutado mediante la terminal de MacOS, versión del SO Catalina.
Además, si se quiere ver la evolución del código visitar el Repositorio