Fundamentos de los Sistemas Operativos
Ficha de entrega de práctica

**: campo obligatorio*
***IMPORTANTE:** esta ficha no debe superar las DOS PÁGINAS de extensión*

Grupo de prácticas*: 44
Miembro 1: Kilian Armas Pérez
Miembro 2:

Número de la práctica*: 1	**Fecha de entrega*:** 22 de marzo de 2023

Descripción del trabajo realizado*
(Algoritmos que han implementado, llamadas al sistema utilizadas, estrategia que han seguido para resolver el problema, etc.)
(no copiar el enunciado del trabajo, eso ya lo conocemos los profesores)

Para la implementación de la gestión de reservas de los asientos de una sala de teatro se ha usado una estructura de datos basada en un vector, cuyo tamaño es la capacidad de la sala en cuestión y que representa tanto los asientos libres como los ocupados. El código de la práctica se ha repartido en tres ficheros .c: el primero de estos es ***main.c*** (programa principal de ejecución de pruebas), el siguiente es ***sala.c*** (funciones básicas para el tratamiento de la sala) y el último es ***test.c*** (funciones de prueba más específicas). Además de estos ficheros, podemos encontrar dos .h (***sala.h*** y ***test.h***), que se encargan de la implementación de las funciones tanto de sala.c como de test.c.

Entrando más en materia, dentro del fichero sala.c nos encontramos con ciertas funciones imprescindibles para el manejo de la sala como ***crea_sala()***, la cuál se encarga de crear una nueva sala con la capacidad que se le pasa como parámetro, reservar un espacio en memoria de tamaño igual a la capacidad indicada e inicializar la sala con todos sus valores a 0 para indicar que está vacía. En contraparte, se halla la función ***elimina_sala()*** que libera el espacio de memoria que ocupa dicha sala. Otras funciones muy importantes que se presentan son ***reserva_asiento()*** (asigna un asiento a una persona si es posible), ***libera_asiento()*** (libera un asiento si está ocupado) y ***estado_asiento()*** (devuelve el estado de un asiento si este se encuentra dentro de la capacidad de la sala). También nos podemos encontrar con las funciones ***capacidad()***, ***asientos_libres()*** y ***asientos_ocupados()***, que devuelven la capacidad, el número de asientos libres y la cantidad de asientos ocupados, respectivamente.

Horas de trabajo invertidas* Miembro 1: 12 horas **Miembro2:**
(indicar las horas de todos los integrantes)

Cómo probar el trabajo*
(qué debe hacer el profesor para utilizar el programa entregado: nombre de los programas, instrucciones de compilación, opciones de menú, datos de prueba, argumentos de invocación al programa, etc.)

Para la resolución de las pruebas se puede hacer uso de las funciones implementadas en el fichero ***test.c***, las cuáles sirven de ayuda para realizar comprobaciones más exhaustivas de los resultados de las operaciones realizadas. En primer lugar, se encuentra una función llamada ***estado_sala()***, que devuelve el estado de cada uno de los asientos haciendo llamadas iterativas al método *estado_asiento()* hasta llegar a la capacidad de la sala, e imprime por pantalla tanto el aforo máximo como la ocupación actual de la sala en cuestión. Por otro lado, se hayan las funciones ***sentarse()*** y ***levantarse()***, que se encargan de imprimir mensajes por pantalla para indicar todos los posibles resultados de las operaciones *reserva_asiento()* y *libera_asiento()*, respectivamente. Por último, el propósito de las funciones ***reserva_multiple()*** y ***libera_multiple()*** es el de reservar o liberar N asientos al mismo tiempo e imprimir mensajes descriptivos de los resultados en todos los posibles casos que se puedan dar.

Grupo de prácticas*: 44
Miembro 1: Kilian Armas Pérez
Miembro 2:

Número de la práctica*: 1	**Fecha de entrega*:** 22 de marzo de 2023

Con el propósito de compilar el código propuesto en esta práctica, se deberá generar un ejecutable con un nombre definido por el usuario y posteriormente ejecutarlo haciendo uso de las siguientes instrucciones:

- ***gcc -fcommon main.c sala.c test.c -o ejecutable***
- ***./ejecutable***

En el fichero ***main.c*** podemos encontrar tanto el test básico provisto en el Campus Virtual (***test_ReservaBásica()***) como una serie de tests que engloban todos los resultados esperados del programa ante unos datos de prueba ya definidos, así como las excepciones que puedan aparecer en ciertos casos especiales como cuando la sala está vacía y se quiere liberar uno o más asientos (***test_SalaVacía()***), o cuando está llena y se desea reservar uno o más asientos (***test_SalaLlena()***). Además de estos, también se pueden encontrar los tests: ***test_ReservaSimple()***, donde se prueban las funciones de *sentarse()*, *levantarse()* y *estado_sala()*; y ***test_ReservaMultiple()***, en el cual se experimenta con las funciones de reserva y liberación múltiple (*reserva_multiple()* y *libera_multiple()*). Por último, se han agregado dos nuevos tests: ***test_ReservaNegativa()***, en donde se prueba a crear una sala con capacidad negativa, a liberar un asiento negativo y a comprobar su estado; y ***test_ReservaTeclado()***, donde se procede a probar las diferentes funciones de la sala introduciendo los parámetros por teclado.

Incidencias
(errores no resueltos, anomalías, cualquier cosa que se salga de lo normal)

Todos los errores y anomalías que han ido apareciendo a lo largo del desarrollo de la práctica y de sus pertinentes pruebas han sido solucionados. Además, se han corregido las observaciones negativas de la entrega anterior.

Comentarios
(observaciones adicionales que quieran anotar)

Hay que destacar que en cada uno de los tests que se encuentran en el fichero main.c, se crea una nueva sala con una capacidad definida por el usuario anteriormente y se elimina al terminar todas las operaciones que se deseen realizar.

A partir de esta nueva versión, se comprueba antes de crear la sala si la capacidad es válida y no se permiten liberar asientos con posición negativa ni comprobar su estado. También se ha cambiado la implementación de las funciones *asientos_libres()* y *asientos_ocupados()*, para que no haya que recorrer todo el vector completo cada vez que sean referenciadas, almacenando ahora el valor en dos nuevas variables *libres* y *ocupados*.