Fundamentos de los Sistemas Operativos
Ficha de entrega de práctica 2

Grupo de prácticas*: **Miembro 1: Romen Adama Caetano Ramirez**	
Número de la práctica*: 2/Revisión	**Fecha de entrega*:** 24/05/2023

Descripción del trabajo realizado*

Descripción del trabajo realizado

El presente informe describe las modificaciones realizadas en la práctica 2 previamente entregada.

En primer lugar, se solicitó la ampliación del archivo sala.c proporcionado en la entrega inicial. Dicho programa, denominado sala.c, es una aplicación que permite crear, gestionar y almacenar información sobre una sala (API). La estructura Sala almacena datos como los asientos, la capacidad, los asientos libres y los ocupados. Para esta segunda entrega, se han incorporado las siguientes funciones que permiten acceder al archivo, escribirlo y guardarlo:

1. *guarda_estado_sala: Esta función se encarga de guardar el estado actual de la sala en un archivo especificado. En primer lugar, verifica si el archivo existe y lo abre en modo lectura/escritura, creándolo en caso de que no exista. A continuación, reserva memoria para un búfer de tamaño 10. Escribe la capacidad de la sala en el archivo y luego escribe el ID de cada cliente en los asientos correspondientes.*

2. *recupera_estado_sala: Esta función recupera el estado de la sala a partir de la información almacenada en un archivo especificado. Verifica la existencia del archivo y lo abre en modo lectura. Lee la capacidad de la sala desde el archivo y crea una sala con dicha capacidad. A continuación, lee los ID de los clientes desde el archivo y los guarda en el array de asientos de la sala.*

3. *guarda_estadoparcial_sala: Esta función guarda el estado de un conjunto de asientos de la sala en un archivo preexistente. Verifica si el archivo existe y lo abre en modo lectura/escritura. Luego, reserva memoria para un búfer de tamaño 10 y salta 10 bytes en el archivo. Recorre los asientos de la sala y, si el asiento se encuentra en el conjunto de asientos especificado, escribe el ID del cliente en el archivo.*

4. *recupera_estadoparcial_sala: Esta función recupera el estado de un conjunto de asientos de la sala desde un archivo preexistente. Verifica la existencia del archivo y lo abre en modo lectura. Luego, reserva memoria para un búfer de tamaño 10 y salta 10 bytes en el archivo. Recorre los asientos de la sala y, si el asiento se encuentra en el conjunto de asientos especificado, lee el ID del cliente desde el archivo y actualiza el estado del asiento en la sala.*

En segundo lugar, se solicitó el desarrollo del programa misala.c.

El programa "misala" permite realizar diversas operaciones sobre los asientos de una sala. Se invoca con la siguiente sintaxis general: "misala orden argumentos".

El punto central del programa es la función "main()", que desempeña un papel fundamental en la ejecución y control del flujo del programa. En esta función, se declaran y se inicializan las variables necesarias para el funcionamiento del programa, como contadores, flags o estructuras de datos.

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

Número de la práctica*: 2/Revisión	**Fecha de entrega*:** 24/05/2023

Además, se utiliza la función "getopt()" para procesar los argumentos de línea de comandos proporcionados por el usuario al momento de ejecutar el programa. Esta función permite analizar y extraer las opciones y argumentos ingresados, lo que facilita el establecimiento de los modos de operación del programa y la toma de decisiones correspondientes.

El programa admite varios modos de operación, como la creación de una sala con una capacidad especificada, la reserva o anulación de asientos, y la obtención del estado parcial o completo de la sala, entre otros. Estos modos se activan mediante opciones de línea de comandos, como "-c" (crear sala), "-f" (fichero), "-n" (reserva),"-a" (anulación) y " " (estado).

Revisión y ampliación de la practica (02 de junio del 2023)

- Manejo de errores: En esta revisión, se ha agregado un manejo de errores utilizando la biblioteca de errno, donde las funciones perror() muestran por consola interna los posibles errores generados y verificando el valor de retorno de las llamadas a las funciones de E/S (como open(), read(), write(), close()). Esto permite obtener información detallada sobre los errores ocurridos. En cambio, las funciones anteriores no manejan explícitamente los errores y no proporcionan información detallada sobre los errores ocurridos.
- Comprobación de apertura de ficheros: En las funciones actualizadas, se verifica si la llamada a open() ha sido exitosa antes de continuar con las operaciones de lectura o escritura. En las funciones anteriores, no se realiza esta comprobación.
- Control de capacidad de la sala: En las funciones revisadas, se utiliza la función capacidad() para obtener la capacidad de la sala. En las funciones anteriores, no se muestra cómo se obtiene la capacidad de la sala.
- Uso de ssize_t para el retorno de funciones de E/S: En las funciones revisadas, las llamadas a read() y write() devuelven un valor de tipo ssize_t, que indica la cantidad de bytes leídos o escritos. En las funciones anteriores, no se verifica el valor de retorno de estas funciones.
- Uso de malloc() para asignar memoria: En las funciones revisadas, se utiliza malloc() para asignar memoria al buffer de lectura/escritura (buffer). En las funciones anteriores, se utiliza un arreglo de caracteres estático (char buffer[10]).
- Uso de off_t para el desplazamiento en el fichero: En las funciones revisadas, se utiliza el tipo de datos off_t para almacenar el desplazamiento en el fichero al utilizar la función lseek(). En las funciones anteriores, se utiliza un int para el desplazamiento.
-

Horas de trabajo invertidas* Miembro 1:
12 horas

Cómo probar el trabajo*

- Descomprimir él .zip y mediante comandos de terminal (cd) desplazarse hasta el directorio:
- ***…/pract2*** (seguramente sea el directorio por defecto al descomprimir)

Ejecutar el script desarrollado para facilitar la manipulación del programa:
(se encarga de compilar los archivos en caso de alguna modificación en los mismos)

- ***bash script.sh***

En el directorio /pract2/fuentes/ y ejecutar el programa misala. Existe varios modos:

- *./misala crea -c 10 -f sala.txt (crea un fichero de una sala con 10 asientos)*
- *./misala crea -c 15 -f sala.txt -o (modifica la sala previamente creada con el comando –o)*
- *./misala reserva -f sala.txt -n 3 1 2 2 (error no se puede reservar asientos con el mismo identificador)*
- *./misala reserva -f sala.txt -n 3 1 2 3 (comando de reserva correcto)*

Grupo de prácticas*: **Miembro 1: Romen Adama Caetano Ramirez**	
Número de la práctica*: 2/Revisión	**Fecha de entrega*:** 24/05/2023

- *./misala estado -f sala.txt (muestra el estado de la sala)*
- *./misala anula -f sala.txt -a 1 2 (anula la reserva de los asientos 1 y 2)*

En el directorio /pract2/fuentes/ y ejecutar el programa de funciones parciales. Este programa es para el testeo de las funciones parciales.

- ./parciales

Incidencias

- *Como primer intento y como se comentó en mi primera entrega, estaba trabajando con un fichero que trabajaba con caracteres, esto complico el desarrollo de la práctica, y por tanto migue a un "buffer de enteros".*

- *No podía aplicar un modo_parcial en misala para el testeo de las funciones parciales, no encontré el motivo, por tanto, desarrollé una clase y un programa aparte.*

- *Al cambiar del modo de caracteres al modo de enteros, para las funcines de recuperar y guardar, tanto enteras como parciales, se me olvido la parte de a;adir la comprobacion y manejo de errores, causando que tuviera que revisar la practica e incorporar lo requerido por el profesor.*

Comentarios

Las capturas, se encuentran dentro del .zip, en una carpeta llamada MEDIA, en ella se adjuntan las capturas del funcionamiento de los diferentes modos que se encuentra para misala y parciales.
Link del repositorio para ver la evolución del proyecto:
https://github.com/Romen-Adama-Dev/FSO.git