PROYECTO DE SISTEMAS OPERATIVOS

Sistema para el Préstamo de Libros

Integrantes:

Juan Martín Sánchez

Natalia Ramirez

Juan Sebastian Tellez

Juan Esteban Camargo

Santiago Martinez

Profesor:

John Corredor

26 de Mayo el 2025

Indice

1. Introducción:

Con este proyecto se buscó diseñar e implementar un sistema de gestión de préstamos de libros, como parte del desarrollo académico en la asignatura de Sistemas Operativos. Durante su construcción se aplicaron conceptos fundamentales de programación concurrente aprendidos en clase, como la creación y manejo de procesos e hilos mediante la biblioteca POSIX, así como mecanismos de comunicación interprocesos usando pipes y técnicas de sincronización con semáforos.

La solución propuesta consistió en simular las principales funciones de una biblioteca préstamo, renovación y devolución de libros mediante la interacción entre procesos solicitantes (PS) y un proceso receptor (PR). El receptor es responsable de coordinar y actualizar una base de datos en memoria, utilizando hilos de trabajo que permiten procesar solicitudes de forma simultánea y controlada.

El sistema fue diseñado para operar tanto de forma automatizada, a través de archivos de texto de entrada, como en modo interactivo mediante menús. Además, se incluyeron funcionalidades para generar reportes, monitorear el sistema desde una consola y guardar el estado final de la base de datos. El resultado fue un sistema estable y funcional que permite simular múltiples peticiones concurrentes de manera organizada, aplicando correctamente los principios de sincronización y comunicación vistos durante el curso.

2. Objetivo del proyecto:

El objetivo principal de este proyecto fue diseñar e implementar un sistema que simule el funcionamiento básico de una biblioteca, permitiendo gestionar operaciones de préstamo, renovación y devolución de libros, aplicando conceptos de programación concurrente y lo visto en la materia de sistemas operativos.

De forma específica, se buscó:

- Aplicar los conocimientos sobre creación y gestión de procesos e hilos usando la biblioteca POSIX.
- Implementar comunicación entre procesos a través de pipes y sincronización mediante semáforos.
- Desarrollar un sistema que soporte múltiples solicitudes concurrentes de forma segura y controlada.
- Simular la interacción entre múltiples procesos solicitantes y un proceso receptor que coordina y actualiza una base de datos compartida.
- Facilitar la interacción tanto en modo automático (archivo de solicitudes) como en modo interactivo (menú).

3. Descripción general del sistema:

El sistema desarrollado simula el funcionamiento básico de una biblioteca digital a través de la interacción entre procesos concurrentes. Está compuesto por tres módulos principales: los procesos solicitantes (PS), el proceso receptor (PR) y el módulo de base de datos.

Los procesos solicitantes representan usuarios que envían solicitudes de préstamo, devolución o renovación de libros. Estas solicitudes son comunicadas al proceso receptor mediante un pipe nominal. El receptor actúa como coordinador central, encargado de interpretar, validar y gestionar cada solicitud. Para ello, accede y modifica una base de datos que contiene el estado actual de todos los libros y sus ejemplares.

Las solicitudes de préstamo son procesadas directamente por el hilo principal del receptor, mientras que las operaciones de renovación y devolución son delegadas a un hilo auxiliar que opera en paralelo, utilizando un buffer circular sincronizado con semáforos para garantizar la exclusión mutua y el control de concurrencia.

Además, el sistema incluye una consola interactiva que permite ejecutar comandos de control como finalizar la ejecución. El receptor también ofrece la posibilidad de guardar el estado final de la base de datos en un archivo de salida.

El sistema fue diseñado para funcionar tanto en modo automático, leyendo solicitudes desde un archivo, como en modo interactivo, mediante el ingreso de datos por menú. Esto permite simular diferentes escenarios de uso, verificar el correcto funcionamiento del sistema y evaluar su robustez frente a solicitudes válidas e inválidas.

4. Implementación del sistema:

La implementación del sistema se basa en tres componentes principales: el proceso solicitante (PS), el proceso receptor (PR) y el módulo de base de datos. Estos módulos fueron desarrollados de forma modular y conectados mediante mecanismos de comunicación y sincronización como pipes, hilos y semáforos POSIX.

4.1. Proceso Solicitante (PS)

El archivo Solicitante.c implementa el comportamiento de los procesos solicitantes, encargados de generar solicitudes hacia el receptor. Estas solicitudes pueden enviarse en dos modos de operación:

- **Modo archivo:** el solicitante envía un archivo de texto (solicitudes.txt) con operaciones como P (préstamo), R (renovación), D (devolución) y Q (salir), y las envía secuencialmente al receptor.
- **Modo interactivo:** el usuario ingresa manualmente cada operación desde un menú en consola, lo que permite simular casos puntuales.

En ambos modos, las solicitudes se escriben en un pipe nominal (pipeRP), el cual es leído por el proceso receptor

4.2. Proceso Receptor (PR)

El archivo Receptor.c contiene el núcleo de la lógica del sistema. Este proceso actúa como receptor central de las solicitudes, leyendo desde el pipe y procesando cada petición según su tipo.

- **Las solicitudes de préstamo (P)** son procesadas directamente por el hilo principal, verificando disponibilidad en la base de datos y registrando el préstamo si es posible.
- **Las solicitudes de renovación (R) y devolución (D)** se envían a un hilo auxiliar, el cual trabaja en paralelo con el hilo principal. Esta delegación se realiza mediante un buffer circular sincronizado con semáforos (sem_t) y protegido con un mutex.

El receptor también posee un hilo de consola que permite controlar la ejecución del sistema desde teclado. Por ejemplo, ingresar s detiene el sistema de forma controlada, y r permite mostrar un mensaje de simulación de reporte.

Al finalizar, si se especificó un archivo de salida con -s, el receptor guarda la base de datos actualizada en dicho archivo (bd_final.txt), manteniendo la persistencia del estado del sistema.

4.3. Módulo de base de datos

El archivo SistemaDePrestamoDeLibros.c contiene las funciones que gestionan la base de datos, implementada como un arreglo global de estructuras Libro y Ejemplar. Este módulo ofrece funciones para:

- Cargar la base de datos inicial desde un archivo de texto (bd.txt)
- Buscar libros por ISBN
- Realizar préstamos (P), devoluciones (D) y renovaciones (R)
- Guardar el estado final en un archivo (bd_final.txt)

El archivo SistemaDePrestamoDeLibros.h contiene las declaraciones de funciones y estructuras, lo que permite usar el módulo desde otros componentes como el receptor.

4.4. Archivos auxiliares y de soporte del sistema

Además de los procesos principales y el módulo de base de datos, el sistema cuenta con un conjunto de archivos auxiliares que cumplen funciones complementarias fundamentales para su correcta ejecución, organización y validación. A continuación, se describen brevemente:

- **SistemaDePrestamoDeLibros.h**
  Archivo de cabecera que define las estructuras Libro y Ejemplar, así como los prototipos de funciones utilizadas en el módulo de base de datos. Permite la integración del módulo con el receptor de forma modular.

- **Makefile**
  Archivo de automatización de compilación. Permite compilar todos los módulos del sistema de forma eficiente mediante los comandos make y make clean. Define las dependencias entre archivos y genera los ejecutables solicitante y receptor.

- **bd.txt**
  Archivo de entrada que representa el estado inicial de la base de datos de libros, incluyendo nombre, ISBN, cantidad de ejemplares, y el estado y fecha de cada ejemplar. Es cargado por el receptor al iniciar.

- **bd_final.txt**
  Archivo de salida que contiene el estado final de la base de datos luego de procesar todas las solicitudes. Se genera automáticamente al finalizar el sistema si se especifica con el parámetro -s.

- **solicitudes.txt**
  Archivo de entrada que contiene solicitudes simuladas que el proceso solicitante puede ejecutar automáticamente. Cada línea corresponde a una operación de préstamo (P), renovación (R), devolución (D) o salida (Q).

4.5. Compilación

Para facilitar la construcción del proyecto, se incluyó un archivo Makefile que automatiza el proceso de compilación, gestionando las dependencias entre archivos fuente y generando los ejecutables correspondientes.

Comandos principales:

Para compilar el proyecto completo, se utiliza el comando make, el cual construye todos los ejecutables necesarios. En particular, genera dos archivos ejecutables: *solicitante* y *receptor*.

En caso de requerir una recompilación desde cero o para limpiar los archivos temporales generados durante compilaciones anteriores, se puede utilizar el comando make clean. Este elimina los archivos objeto (o) y los ejecutables previos, dejando el entorno listo para una nueva compilación.

Una vez compilado el proyecto, se puede ejecutar el sistema mediante los ejecutables generados: *receptor* y *solicitante*. Para su funcionamiento, es necesario crear un pipe nominal que sirva como canal de comunicación entre ambos procesos. Este pipe debe crearse manualmente antes de ejecutar el receptor.

Preparación inicial:

Antes de ejecutar los programas, se debe crear el pipe utilizando el siguiente comando en la terminal:

Crear el pipe nominal: *mkfifo pipeRP*

Ejecución del proceso receptor:

El proceso receptor debe iniciarse primero, ya que es quien se encargará de leer las solicitudes provenientes del solicitante. Este proceso recibe como argumentos el nombre del pipe, el archivo de base de datos inicial, y opcionalmente un archivo donde se guardará la base de datos actualizada al finalizar la ejecución. También puede activarse el modo *verbose* para mostrar información detallada durante la ejecución.

Ejemplo de ejecución: *./receptor -p pipeRP -f bd.txt -v -s bd_final.txt*

Donde:

- -p indica el nombre del pipe.
- -f especifica el archivo de base de datos de entrada.
- -v activa el modo detallado (*verbose*).
- -s permite definir el archivo de salida donde se guardará la base de datos final.

Ejecución del proceso solicitante:

Una vez que el receptor está activo, se puede iniciar el proceso solicitante. Este puede ejecutarse en dos modos distintos:

- **Modo interactivo:** solicita al usuario cada dato por consola.
  - **Ejecución:** *./solicitante -p pipeRP*

- **Modo automático por archivo:** lee un archivo de texto con múltiples solicitudes ya definidas.
  - **Ejecución:** *./solicitante -i solicitudes.txt -p pipeRP*

Donde:

- -i indica el archivo que contiene las solicitudes en formato texto.

- -p especifica el pipe por el cual se enviarán las solicitudes al receptor.

Es importante asegurarse de que el pipe haya sido creado antes de iniciar los procesos, y que el receptor esté corriendo antes de ejecutar el solicitante. Al finalizar la ejecución, si se utilizó el parámetro -s, el receptor generará un archivo *bd_final.txt* con el estado actualizado de la base de datos tras procesar todas las solicitudes.

4.6. Funciones desarrolladas y su propósito en el sistema

Solicitante.c:

enviar_peticion(int fd, char operacion, char libro, int isbn)

Esta función construye y envía una solicitud al proceso receptor a través de un pipe. Recibe como parámetros un descriptor de archivo fd asociado al pipe, un carácter operación que indica el tipo de solicitud (puede ser ‘P’, ‘R’, ‘D’ o ‘Q’), un puntero libro que contiene el nombre del libro, y un entero isbn correspondiente al identificador del libro.

Internamente, la función utiliza snprintf() para construir un mensaje de texto con el formato <operacion>,<libro>,<isbn>\n. Luego, emplea la llamada al sistema write() para escribir el mensaje en el pipe y enviarlo al proceso receptor.

menu_interactivo(int fd)

Esta función permite al usuario generar solicitudes de manera manual mediante un menú interactivo. Recibe como argumento un descriptor de archivo fd vinculado al pipe de salida.
Internamente, entra en un ciclo while que solicita al usuario tres entradas consecutivas:

- Un carácter op correspondiente al tipo de operación.
- Una cadena nombre que contiene el nombre del libro (acepta espacios mediante scanf(" %[^\n]")).
- Un número entero isbn correspondiente al código del libro.

Si la operación ingresada es Q, se envía una solicitud de salida y se rompe el ciclo. Para las demás operaciones (P, R, D), se llama a enviar_peticion() con los datos recolectados para transmitir la solicitud.

****leer_archivo_y_enviar(const char archivo, int fd)***

Esta función permite al solicitante operar en modo automático, leyendo una serie de solicitudes predefinidas desde un archivo de texto.
Recibe como parámetros el nombre del archivo que contiene las solicitudes (una por línea) y el descriptor de archivo fd vinculado al pipe de salida.
Internamente, abre el archivo en modo lectura utilizando fopen(). Luego, itera con fgets() sobre cada línea del archivo, escribiéndola directamente en el pipe con write(). Finaliza cerrando el archivo con fclose(). No realiza validaciones sobre el contenido del archivo, asume que las líneas están correctamente formateadas.

****main(int argc, char argv[])***

Esta es la función principal del proceso solicitante. Su objetivo es interpretar los argumentos recibidos por línea de comandos y establecer la comunicación con el proceso receptor.
Utiliza getopt() para extraer los parámetros -i (archivo de entrada) y -p (nombre del pipe). Si no se proporciona el nombre del pipe, el programa termina con un mensaje de error. Después de obtener los parámetros, abre el pipe en modo escritura (O_WRONLY). Si el modo automático fue especificado (-i), se invoca leer_archivo_y_enviar(). De lo contrario, se ejecuta menu_interactivo() para permitir el ingreso manual. Finalmente, cierra el pipe con close() y retorna 0.

SistemaDePrestamoDeLibros.c

fecha_actual()

Esta función genera la fecha actual del sistema en formato dd-mm-aaaa y la retorna como una cadena de caracteres. Internamente, utiliza la función time() para obtener la fecha y hora actual, localtime() para convertirla en una estructura tm, y snprintf() para formatear los valores en una cadena estática de 11 caracteres. Esta función es utilizada en operaciones como préstamos y devoluciones para registrar la fecha en la que se realiza la acción.

fecha_mas_7_dias()

Devuelve la fecha correspondiente a siete días posteriores a la fecha actual del sistema, formateada como dd-mm-aaaa. Para ello, suma siete días en segundos a la marca de tiempo actual (time_t) y luego usa localtime() y snprintf() para obtener el formato deseado. Es utilizada específicamente al renovar un préstamo para actualizar la fecha de devolución del ejemplar.

****guardar_base_datos(const char archivo)***

Esta función guarda en un archivo el estado actual de la base de datos en memoria, respetando el mismo formato con el que fue leída. Recibe el nombre del archivo de salida como parámetro. Recorre el arreglo biblioteca[] e imprime en el archivo los datos de cada libro (nombre, ISBN, cantidad de ejemplares), así como los datos de cada ejemplar (número, estado, fecha). Si no hay libros cargados, emite una advertencia mediante printf(). Retorna 0 si la operación fue exitosa o -1 si no se pudo abrir el archivo.

buscar_libro(int isbn)

Busca en el arreglo global biblioteca[] un libro cuyo código ISBN coincida con el valor recibido como parámetro. Retorna un puntero al libro encontrado, o NULL si no se encontró ningún libro con ese ISBN. Durante la búsqueda, imprime mensajes de depuración con los datos de cada libro evaluado. Esta función es utilizada por todas las operaciones que necesitan acceso directo al libro correspondiente.

prestar_libro(int isbn)

Esta función intenta registrar un préstamo para el libro identificado por el ISBN recibido como parámetro. Llama a buscar_libro() para encontrar el libro correspondiente. Si se encuentra, recorre sus ejemplares en busca del primero que tenga estado 'D' (disponible). Si lo encuentra, cambia su estado a 'P' (prestado) y actualiza la fecha del préstamo usando fecha_actual(). Devuelve 1 si el préstamo fue exitoso, 0 si no hay ejemplares disponibles y -1 si el libro no existe.

devolver_libro(int isbn)

Permite registrar la devolución de un ejemplar de un libro previamente prestado. Busca el libro con el ISBN especificado y recorre sus ejemplares en busca del primero que tenga estado 'P'. Si lo encuentra, cambia su estado a 'D' (disponible) y actualiza la fecha con la función fecha_actual(). Retorna 1 si la devolución fue registrada, 0 si no había ejemplares prestados y -1 si el libro no fue encontrado.

renovar_libro(int isbn)

Renueva el préstamo de un ejemplar prestado de un libro identificado por el ISBN. Requiere que el ejemplar esté en estado 'P'. Al encontrarlo, actualiza su fecha con fecha_mas_7_dias() y retorna el índice del ejemplar renovado. Si no se encuentra un ejemplar prestado, devuelve -1. Esta función es utilizada para extender la fecha de devolución de un préstamo activo.

Receptor.c:

procesar_peticion(Peticion p)

Función encargada de procesar las solicitudes del tipo renovación (R) y devolución (D) una vez que han sido extraídas del buffer por el hilo auxiliar.
Recibe una estructura Peticion que contiene el tipo de operación, el nombre del libro y su ISBN. En caso de una devolución, se llama a devolver_libro() y se imprime el resultado de la operación. En caso de renovación, se ejecuta renovar_libro(), que actualiza la fecha de entrega del ejemplar prestado. La función muestra en consola el resultado, incluyendo la nueva fecha en caso de éxito.

**hilo_auxiliar(void arg)*

Hilo secundario encargado de procesar las solicitudes de tipo renovación (R) y devolución (D) que han sido colocadas en el buffer circular.
El hilo permanece en ejecución hasta que la bandera terminar sea activada. Internamente, utiliza sem_wait() para esperar elementos en el buffer (full), bloquea la sección crítica con un mutex, extrae la solicitud (Peticion), libera el espacio (sem_post(&empty)) y finalmente llama a procesar_peticion().

**hilo_consola(void arg)*

Hilo adicional que permite al usuario interactuar directamente con el receptor mediante la consola.
Este hilo espera comandos de teclado: si el usuario ingresa 's', se activa la bandera global terminar, lo que provoca el cierre ordenado del sistema. Si se ingresa 'r', se imprime un mensaje simulado de reporte. Este hilo facilita el cierre manual y monitoreo del sistema durante las pruebas.

publicar_en_buffer(Peticion p)

Inserta una solicitud (Peticion) en el buffer circular sincronizado.
Primero espera espacio disponible (sem_wait(&empty)), luego accede a la sección crítica protegida por pthread_mutex_lock(), inserta la solicitud en la posición in, actualiza el índice circular, libera el mutex y finalmente indica que hay un nuevo elemento disponible (sem_post(&full)).

***procesar_linea(char linea):**

Recibe una línea de texto con el formato <tipo>,<libro>,<isbn> y la convierte en una estructura Peticion.
Si la solicitud es de tipo P (préstamo), se procesa directamente mediante la función prestar_libro(). Si es de tipo R o D, se coloca en el buffer circular mediante publicar_en_buffer(). Si la solicitud es Q, se imprime un mensaje y se activa la variable terminar para finalizar el receptor. También muestra mensajes detallados si el modo verbose está activado.

***main(int argc, char argv[]):**

Función principal del proceso receptor. Su responsabilidad es inicializar el sistema, interpretar los argumentos de entrada, crear los hilos, leer las solicitudes desde el pipe, y al finalizar, guardar la base de datos actualizada y liberar los recursos del sistema.
Interpreta las opciones -p (nombre del pipe), -f (archivo de base de datos inicial), -s (archivo de salida), y -v (modo verbose). Abre el pipe en modo lectura, inicializa semáforos y mutex, y lanza los hilos hilo_auxiliar y hilo_consola. En su bucle principal, lee del pipe, descompone las líneas y llama a procesar_linea() para cada solicitud. Al final, espera la terminación de los hilos, guarda la base si se indicó, y destruye los objetos de sincronización.

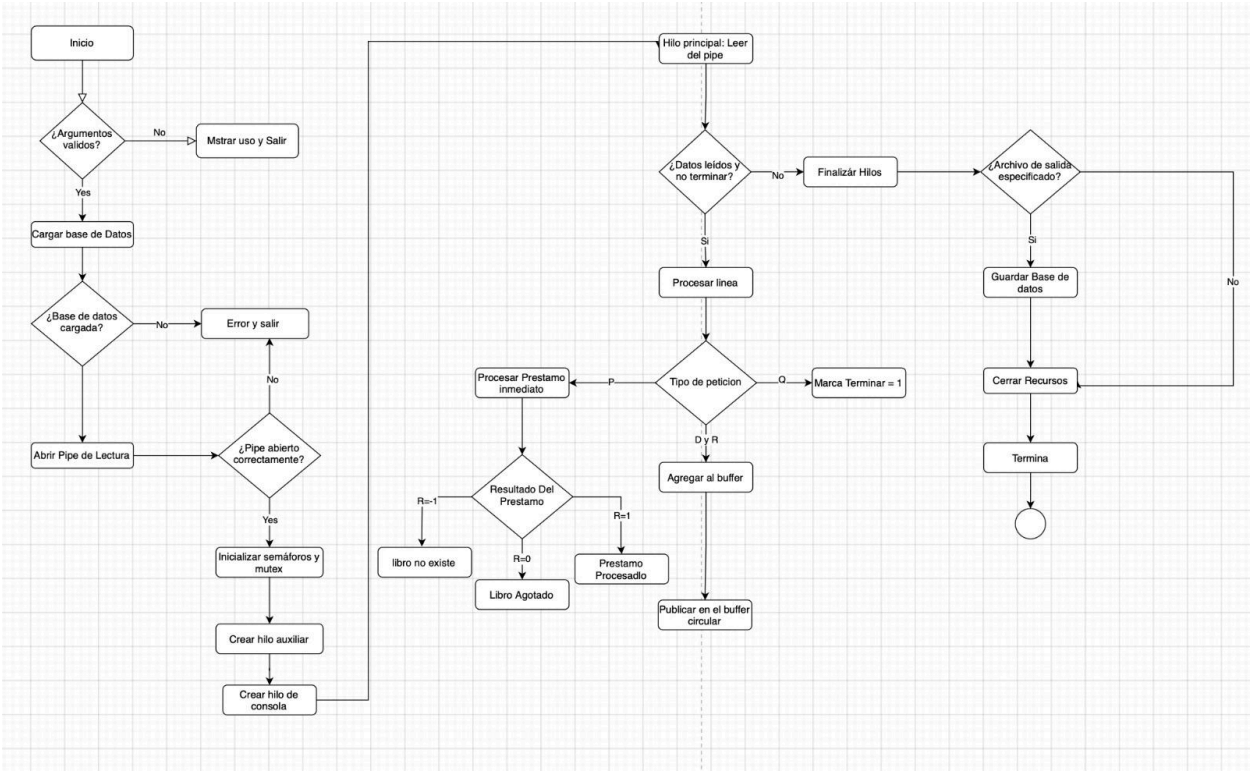
5. Diagramas de flujo del sistema

Con el objetivo de representar de forma visual y estructurada el comportamiento del sistema, a continuación se presentan los diagramas de flujo correspondientes a los principales archivos fuente del proyecto: Solicitante.c y Receptor.c.

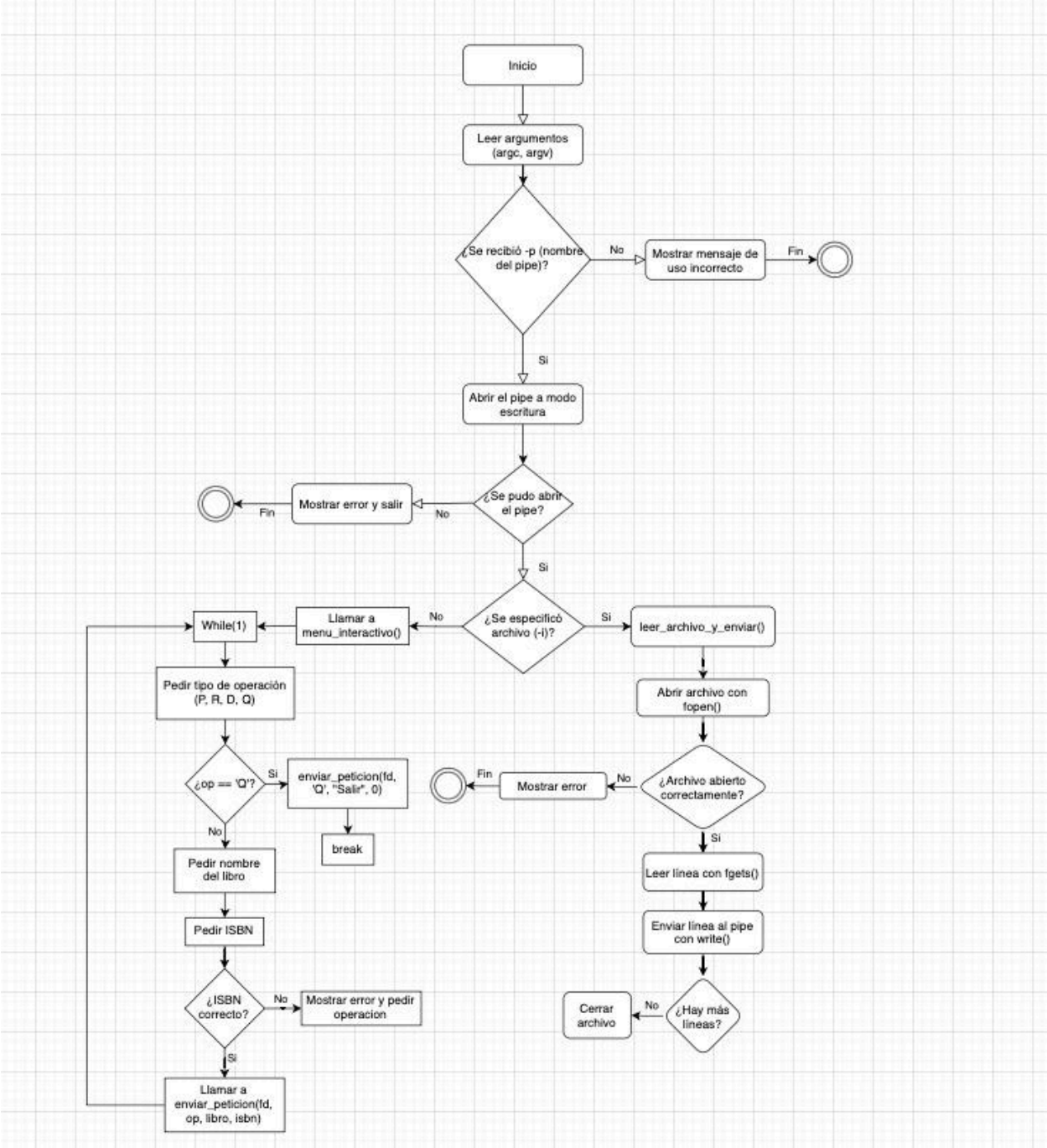
Estos diagramas permiten identificar claramente la lógica secuencial, las decisiones condicionales, las llamadas a funciones, y el flujo de comunicación entre procesos y módulos internos.

Cada diagrama refleja el recorrido lógico desde el inicio de ejecución hasta el cierre controlado del programa, diferenciando los modos de operación y el tratamiento particular que se le da a cada tipo de solicitud (P, R, D, Q).

Receptor.c:



Solicitante.c:



6. Plan de pruebas

Con el fin de validar el correcto funcionamiento del sistema, se diseñó un conjunto de pruebas que simulan distintos escenarios de uso reales e intencionalmente erróneos. Estas pruebas evalúan el comportamiento del sistema frente a solicitudes válidas (préstamos disponibles, renovaciones y devoluciones correctas) así como frente a situaciones límite o inválidas (solicitudes con ISBN inexistente o libros agotados) las pruebas se realizaron el día 26 de mayo del 2025 para tener en cuenta el resultado de la renovación válida.

Prueba	Descripción	Entrada	Salida
01	**Préstamo válido**	**P,Operating Systems,2233**	**[RP] Préstamo realizado con éxito (ISBN 2233)**
02	**Préstamo sin ejemplares disponibles**	**P,Artificial Intelligence,4444**	**[DB] No hay ejemplares disponibles para Artificial Intelligence** **[RP] El libro con ISBN está agotado**
03	**ISBN inexistente**	**P,PruebaErr,9999**	**[DB] Error: No se encontró el libro con ISBN 9999** **[RP] Error: El libro con ISBN 9999 no existe**
04	**Renovación válida**	**R,Operating Systems,2233**	**[RP] Renovación realizada (ISBN 2233), nueva fecha: 02-06-2025**
05	**Devolución válida**	**D,Databases,5555**	**[RP] Devolución realizada con éxito (ISBN 5555)**
06	**Finalizar sistema**	**Q,Salir,0**	**[RP] Recibido: Q,Salir,0** **[RP] PS indica salida**

5.1. Plan de pruebas (Imagenes)

Se evidenciará el funcionamiento del sistema con las imágenes del contenido que nos arroja la salida desde el sistema en el que se ejecutó el programa, se mostrará mediante el ID de la prueba asignado en la tabla anterior, y sus respectiva imagen.

Prueba 01: Se realizó el préstamo de un libro que resultó exitoso.

```
Operación (P: Préstamo, R: Renovación, D: Devolución, Q: Salir): P
Nombre del libro: Operating Systems
ISBN: 2233
```

```
[RP] Recibido: P,Operating Systems,2233
[DEBUG] Buscando ISBN 2233 en biblioteca...
[DEBUG] Revisando libro: Operating Systems, ISBN=2233
[RP] Procesando préstamo de Operating Systems
[RP] Préstamo realizado con éxito (ISBN 2233)
```

Prueba 02: Para probar además de que un ejemplar está agotado, se realizó un préstamo de un libro que solo tenía un ejemplar y luego volver a realizar la acción, demostrando así que el sistema también actualiza el estado del libro, el resultado fue exitoso.

```
Operación (P: Préstamo, R: Renovación, D: Devolución, Q: Salir): P
Nombre del libro: Artificial Intelligence
ISBN: 4444
Operación (P: Préstamo, R: Renovación, D: Devolución, Q: Salir): P
Nombre del libro: Artificial Intelligence
ISBN: 4444
```

```
[RP] Recibido: P,Artificial Intelligence,4444
[DEBUG] Buscando ISBN 4444 en biblioteca...
[DEBUG] Revisando libro: Operating Systems, ISBN=2233
[DEBUG] Revisando libro: Artificial Intelligence, ISBN=4444
[RP] Procesando préstamo de Artificial Intelligence
[RP] Préstamo realizado con éxito (ISBN 4444)
[RP] Recibido: P,Artificial Intelligence,4444
[DEBUG] Buscando ISBN 4444 en biblioteca...
[DEBUG] Revisando libro: Operating Systems, ISBN=2233
[DEBUG] Revisando libro: Artificial Intelligence, ISBN=4444
[DB] No hay ejemplares disponibles para Artificial Intelligence
[RP] El libro con ISBN 4444 está agotado
```

Prueba 03: Prueba para validar que al buscar un libro con ISBN inexistente, arroje un error, su resultado fue exitoso.

```
Operación (P: Préstamo, R: Renovación, D: Devolución, Q: Salir): P
Nombre del libro: PruebaErr
ISBN: 9999
```

```
[RP] Recibido: P,PruebaErr,9999
[DEBUG] Buscando ISBN 9999 en biblioteca...
[DEBUG] Revisando libro: Operating Systems, ISBN=2233
[DEBUG] Revisando libro: Artificial Intelligence, ISBN=4444
[DEBUG] Revisando libro: Databases, ISBN=5555
[DB] Error: No se encontró el libro con ISBN 9999
[RP] Error: El libro con ISBN 9999 no existe
```

Prueba 04: Validación de que una renovación se realizó con éxito, previo a la renovación se realizó el préstamo del libro, el resultado fue exitoso.

```
Operación (P: Préstamo, R: Renovación, D: Devolución, Q: Salir): P
Nombre del libro: Operating Systems
ISBN: 2233
Operación (P: Préstamo, R: Renovación, D: Devolución, Q: Salir): R
Nombre del libro: Operating Systems
ISBN: 2233
```

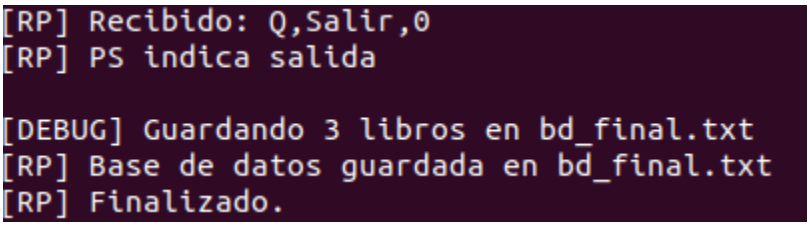
```
[RP] Recibido: P,Operating Systems,2233
[DEBUG] Buscando ISBN 2233 en biblioteca...
[DEBUG] Revisando libro: Operating Systems, ISBN=2233
[RP] Procesando préstamo de Operating Systems
[RP] Préstamo realizado con éxito (ISBN 2233)
[RP] Recibido: R,Operating Systems,2233
[RP] Aceptado R para Operating Systems
[DB] Procesando renovación - Operating Systems (2233)
[DEBUG] Revisando libro: Operating Systems, ISBN=2233
[DEBUG] Revisando libro: Operating Systems, ISBN=2233
[RP] Renovación realizada (ISBN 2233), nueva fecha: 02-06-2025
```

Prueba 05: Validación de que una devolución se haya realizado con éxito con previa solicitud de préstamo de un libro con un ejemplar en la BD, lo agotamos al principio para que al momento de devolverlo se evidencie que efectivamente se puede volver a prestar, con resultado exitoso.

```
Operación (P: Préstamo, R: Renovación, D: Devolución, Q: Salir): P
Nombre del libro: Databases
ISBN: 5555
Operación (P: Préstamo, R: Renovación, D: Devolución, Q: Salir): D
Nombre del libro: Databases
ISBN: 5555
Operación (P: Préstamo, R: Renovación, D: Devolución, Q: Salir): P
Nombre del libro: Databases
ISBN: 5555
```

```
[RP] Recibido: P,Databases,5555
[DEBUG] Buscando ISBN 5555 en biblioteca...
[DEBUG] Revisando libro: Operating Systems, ISBN=2233
[DEBUG] Revisando libro: Artificial Intelligence, ISBN=4444
[DEBUG] Revisando libro: Databases, ISBN=5555
[DB] No hay ejemplares disponibles para Databases
[RP] El libro con ISBN 5555 está agotado
[RP] Recibido: D,Databases,5555
[RP] Aceptado D para Databases
[DB] Procesando devolución - Databases (5555)
[DEBUG] Revisando libro: Operating Systems, ISBN=2233
[DEBUG] Revisando libro: Artificial Intelligence, ISBN=4444
[DEBUG] Revisando libro: Databases, ISBN=5555
[RP] Devolución realizada con éxito (ISBN 5555)
[RP] Recibido: P,Databases,5555
[DEBUG] Buscando ISBN 5555 en biblioteca...
[DEBUG] Revisando libro: Operating Systems, ISBN=2233
[DEBUG] Revisando libro: Artificial Intelligence, ISBN=4444
[DEBUG] Revisando libro: Databases, ISBN=5555
[RP] Procesando préstamo de Databases
[RP] Préstamo realizado con éxito (ISBN 5555)
```

Prueba 06: Se evidencia la salida del programa, el resultado es exitoso finalizando el programa y guardando la BD.



Nota: Durante la ejecución, en algunos segmentos aparece [DEBUG], lo usamos al momento de corregir el código ya que estábamos teniendo errores, por lo que añadimos unas validaciones de los datos y nos pareció interesante dejarlo en el código para que así se evidencie que se cargan y se buscan bien los datos.

7. Conclusiones

La implementación del sistema para la administración de la biblioteca de ejemplos de libros abiertos hizo posible la utilización y aplicación de los conceptos más importantes de la materia de sistemas operativos, lo que nutrió competencias en programación concurrente y manejo de recursos compartidos. Durante el desarrollo del proyecto, se consiguió crear un entorno razonablemente completo que simule el funcionamiento de una biblioteca, donde es posible atender a varios usuarios para préstamos, renovación y devolución de libros en forma concurrente y segura.

Uno de los logros de mayor relevancia fue el uso apropiado de los procesos e hilos POSIX y el uso de pipes como canales de comunicación y semáforos y mutexes como herramientas de control de sincronización. Estos medios eran necesarios para asegurar que las operaciones concurrentes se ejecuten sin conflictos ni pérdida de información. Esto es un correcto uso de control de concurrencia y exclusión mutua.

La estructura modular del sistema con definiciones claras de componentes tales como los procesos solicitantes, el receptor y el módulo de la base de datos, favoreció el diseño, la implementación y ensayos del sistema. Contar con modos de operación automático e interactivo proporciona flexibilidad para probar distintos resultados valorando su estabilidad y funcionalidad, Adicionalmente, la integración de un plan de acción.

8. Referencias

- Silberschatz, A., Galvin, P. B., & Gagne, G. (2018). *Operating System Concepts* (10th ed.). Wiley.
- Kerrisk, M. (2010). *The Linux Programming Interface: A Linux and UNIX System Programming Handbook*. No Starch Press.
- Robbins, K. A., & Robbins, S. (2003). *Unix Systems Programming: Communication, Concurrency and Threads* (2nd ed.). Pearson Education.
- Stevens, W. R., & Rago, S. A. (2013). *Advanced Programming in the UNIX Environment* (3rd ed.). Addison-Wesley.
- Stallings, W. (2018). *Operating Systems: Internals and Design Principles* (9th ed.). Pearson.
- GNU. (s.f.). *The GNU C Library*. GNU Project. Recuperado de https://www.gnu.org/software/libc/manual/