Proyecto Sistemas Operativos 2025_03

Juan Diego Rojas Osorio

Juan José Ballesteros Suarez

Nicolas Pinilla

Link del repositorio:

https://github.com/Jr-412/Sistemas-Operativos/tree/main/Proyecto_SO

PROFESOR:

John Corredor, PhD.

Pontificia Universidad Javeriana

Facultad Ingeniería

Ingeniería de Sistemas

Sistemas Operativos

Bogotá D.C.

1. Introducción

En el ámbito de los sistemas operativos, la gestión eficiente de recursos y la coordinación entre procesos constituyen pilares fundamentales para el desarrollo de aplicaciones concurrentes y distribuidas. Este proyecto busca diseñar e implementar un sistema de reservas para un parque recreativo que simule condiciones reales de operación, utilizando mecanismos propios de los sistemas POSIX.

El sistema desarrollado integra una arquitectura cliente-servidor en la que un proceso controlador gestiona las solicitudes de reserva enviadas por múltiples agentes, los cuales actúan como clientes independientes. La comunicación se realiza mediante el uso de pipes nominales (FIFOs), permitiendo un intercambio estructurado y confiable de mensajes. Además de esto, se incorpora un reloj simulado mediante hilos POSIX, el cual actualiza el estado del parque en intervalos regulares y libera los recursos una vez finalizada la estancia de cada familia.

Este documento presenta de manera detallada el diseño, la implementación y los resultados obtenidos durante la ejecución del proyecto. Se incluye desde el marco teórico, hasta el análisis de las pruebas realizadas bajo distintos escenarios de carga y concurrencia. Asimismo, se explican las estructuras de datos utilizadas, los protocolos de comunicación establecidos y las estrategias de sincronización aplicadas para evitar condiciones de carrera y garantizar la consistencia de la información.

2. Objetivo

El objetivo principal de este proyecto es implementar un sistema de comunicación y coordinación entre procesos que simula la gestión de reservas en un parque recreativo, utilizando pipes nominales, procesos independientes e hilos para el manejo del tiempo en el parque y la concurrencia, garantizando la correcta administración de recursos y evitando condiciones de carrera mediante mecanismos de sincronización.

Objetivos Específicos:

- Implementar la arquitectura cliente-servidor del sistema, desarrollando un proceso controlador encargado de gestionar reservas, validar horarios, administrar el aforo del parque y coordinar la comunicación con múltiples agentes concurrentes mediante pipes nominales.

- Integrar un reloj simulado mediante hilos, empleando pthreads para avanzar la hora del sistema, liberar recursos ocupados por reservas previas y generar eventos internos sin afectar la comunicación con los agentes.

- Aplicar mecanismos de sincronización, como mutex, para proteger estructuras de datos compartidas y evitar condiciones de carrera cunado múltiples solicitudes llegan al controlador de forma simultánea.

- Diseñar y programar agentes capaces de leer solicitudes desde archivos CSV, enviarlas al controlador y procesar las respuestas recibidas.

- Documentar el diseño, la arquitectura del sistema, los protocolos de comunicación, las estructuras de datos empleadas, y las decisiones tomadas para garantizar la correcta gestión de tiempo, concurrencia y reserva de recursos.

3. Marco Teórico

Procesos en Sistemas Operativos POSIX

En los sistemas operativos compatibles con POSIX, un proceso corresponde a la ejecución activa de un programa, contando con su propio espacio de memoria, variables, estado de CPU y recursos asignados. Cada proceso se identifica con un PID (Process ID) único y es controlado a través de diversas llamadas:

-fork(): crea nuevos procesos

-exec(): reemplaza un proceso con uno nuevo

-wait(): sincroniza la ejecución de procesos padre e hijo.

El estándar POSIX garantiza que estas funciones se comporten de una manera consistente en todos los sistemas tipo UNIX. Debido a ello, es posible crear aplicaciones que crean múltiples procesos independientes (En este caso los agentes) que se comuniquen con un proceso central (El controlador).

Sincronización y Exclusión mutua

En los entornos concurrentes, la sincronización es indispensable si se desea evitar errores cuando múltiples tareas acceden de manera simultánea a recursos comunes. Uno de los riesgos más frecuentes es la condición de carrera, la cual surge cuando el resultado de un programa depende del orden en el que se ejecutan hilos o procesos, lo que puede generar datos incoherentes.

Para evitar estas situaciones se emplea la exclusión mutua, un principio que garantiza que únicamente un hilo o proceso pueda entrar en una sección crítica del código en un determinado momento. En POSIX, este control se implementa mediante herramientas como:

-Mutex

-Semáforos

-Variables de condición.

En el contexto del proyecto, la exclusión mutua es clave para resguardar estructuras como la matriz de ocupación por horas y la lista de reservas, ya que múltiples agentes pueden realizar solicitudes simultáneas y el controlador debe gestionar información compartida de manera segura.

Hilos POSIX (pthread)

Los Hilos POSIX, manejados a través de la biblioteca pthread, representan unidades de ejecución ligeras que operan dentro de un mismo proceso y comparten su espacio de direcciones. A diferencia de los procesos originales, su creación es más rápida y requieren menos recursos, lo que los hace ideales para implementar paralelismo interno en una aplicación.

En el contexto del proyecto, los hilos se utilizan para simular el paso del tiempo mediante un reloj que actualiza la hora del parque cada cierto intervalo. Paralelamente, otros hilos continúan enviando solicitudes, lo cual exige mecanismos de coordinación adecuados para mantener la consistencia de los datos.

Comunicación entre procesos con Pipes Nominales

Los pipes nominales, también conocidos como FIFOS, constituyen un método de comunicación entre procesos que permite transferir información mediante un archivo especial ubicado en el sistema de archivos. A diferencia de pipes anónimos que se limitan a procesos con una relación de parentesco, los FIFOS pueden ser utilizados por cualquier proceso que conozca su nombre y cuente con permisos adecuados.

La creación de un pipe nominal se realiza mediante:

mkfifo(“nombre_fifo”, permisos).

Una vez creado, los procesos pueden abrir el FIFO y utilizar las funciones habituales para escribir y leer datos:

-open()

-read()

-write()

-close()

En el contexto del proyecto, se emplea un FIFO general a través del cual todos los agentes envían solicitudes al controlador. Adicionalmente, cada agente genera su propio pipe para recibir respuestas individuales, lo que posibilita una comunicación organizada y asíncrona. Este enfoque resulta apropiado para arquitecturas cliente-servidor sencillas que no requiren mecanismos más complejos como sockets o memoria compartida.

4. Desarrollo del Programa

El programa implementa un sistema distribuido de reservas para un parque, compuesto por dos tipos de procesos: agentes y un controlador. Ambos se comunican mediante pipes nominales (FIFOs) y utilizan una estructura común definida en protocolo.h que estandariza los mensajes intercambiados. A continuación, se describe detalladamente el funcionamiento interno del sistema, la organización de los archivos, la lógica de comunicación y el flujo completo desde el envío de una solicitud hasta la respuesta final.

a. Arquitectura General:

El sistema está divido en dos ejecutables principales:

- **Controlador:** Gestiona el estado global del parque, procesa solicitudes de reserva, asigna horas disponibles y centraliza la información.
- **Agentes:** Lee solicitudes desde un archivo CSV y actúa como cliente, enviando peticiones al controlador y recibiendo respuestas personalizadas.

La comunicación entre ambos se realiza mediante:

1. Un pipe FIFO principal, conocido por todos los agentes y utilizado para enviar solicitudes al controlador.

2. Un pipe FIFO individual por agente, creado dinámicamente a partir del nombre y el PID del agente, utilizado exclusivamente para recibir respuestas.

Cada módulo del programa contribuye a esta arquitectura de la siguiente manera:

- protocolo.h: Define estructuras como Mensaje, Solicitud y EstadoParque.
- pipes.c/h: Implementa funciones de creación, apertura y lectura/escritura de pipes.
- log.c/h: Centraliza mensajes de depuración para controlador y agentes.
- controlador.h: Contiene la lógica de procesamiento y control.
- agente.c: Implementa un proceso agente completamente funcional.

b. Estructuras y Protocolo de comunicación:

Toda la información intercambiada entre agentes y controlador utiliza la estructura Mensaje definida protocolo.h. Esta contiene:

- Datos básicos: tipo de mensaje, agente, familia, número de personas.
- Campos de solicitud: hora_solicitada.
- Repuesta del sistema: hora_asignada, respuesta.
- Información del entorno: pipe exclusivo del agente.

Este diseño permite enviar una unidad de información completa usando una sola operación write() por el pipe.

Además, la estructura EstadoParque mantiene:

- Capacidad por hora (personas_por_hora).
- Cantidad de familias por hora.
- Registro histórico de solicitudes.
- Total de solicitudes procesadas.

El controlador utiliza esta estructura para tomar decisiones coherentes y evitar sobrecupos.

c. Funcionamiento del Agente:

El archivo agente.c implementa todo el flujo de un agente desde su arranque hasta el cierre. Su comportamiento se divide en varias fases:

1. Lectura y validación de argumentos:

El agente recibe parámetros mediante flags:

-s nombre_agente

-s nombre_agente -a archivo_de_solicitudes.csv

-p pipe_principal_controlador

Si los argumentos no coinciden con el número esperado, el programa termina.

2. Creación del pipe de respuesta:

Cada agente crea un pipe único con el patrón:

- /tmp/pipe_respuesta_<agente>_<pid>

Esto evita colisiones entre agentes con el mismo nombre o múltiples ejecuciones simultáneas.

La función crear_pipe() está implementada en pipes.c y usa mkfifo().

3. **Registro ante el controlador:**

El agente crea un mensaje de tipo “registro” con:

- Su nombre de agente
- Su pipe de respuesta

Luego:

1. Abre el pipe principal.

2. Escribe el mensaje de registro.

3. Cierra el pipe.

4. Abre su propio pipe de respuesta para leer.

El agente espera hasta 10 intentos, con pausas de a 1 segundo, a que el controlador confirme el registro enviando un mensaje con la hora actual del sistema.

4. Procesamiento de solicitudes:

El agente abre el archivo CSV, que contiene líneas con el formato:

NombreFamilia,HoraSolicitada,NumPersonas

Para cada línea válida:

1. Verifica si la hora solicitada es anterior a la hora del sistema
   Si lo es → la solicitud se descarta como extemporánea.
2. Crea un mensaje de tipo "solicitud"
   Incluye:
   - Nombre de la familia.
   - Número de personas.
   - Hora solicitada.
   - Nombre del agente.
   - Pipe de respuesta.
3. Escribe el mensaje en el pipe principal
   Utilizando escribir_mensaje() de pipes.c
4. Espera la respuesta en su pipe exclusivo
   El agente escucha hasta recibir una respuesta correspondiente a la misma familia.
   Esto evita mezclar respuestas destinadas a otras solicitudes.
5. Imprime el resultado
   Puede ser: "aprobada", "reprogramada" y "rechazada".
6. Espera un retardo configurable
   Controlado por TIEMPO_ESPERA_AGENTE.

5. Finalización:

Una vez procesado todo el archivo:

1. Se envía un mensaje de tipo **"fin"** al controlador.
2. Se cierra el pipe de respuesta.
3. Se elimina su FIFO exclusivo.
4. El agente muestra un mensaje de salida indicado que terminó su ejecución.

d. Módulo de Pipes: Comunicación entre Procesos:

El archivo pipes.c encapsula todas las operaciones con FIFOs nominales:

- crear_pipe() = crea un FIFO con mkfifo()
- abrir_pipe_lectura() = abre un pipe en modo lectura no bloqueante
- abrir_pipe_escritura() = abre un pipe en modo escritura.
- escribir_mensaje() = escribe una estructura completa mensaje.
- leer_mensaje() = lectura de un bloque del tamaño exacto del mensaje.

Este diseño permite que:

- El controlador procese múltiples agentes sin bloqueo innecesario.
- El agente pueda probar repetidamente si ya llegó la respuesta.
- Se asegure que cada mensaje mantiene una estructura uniforme.

e. **Bitácora y Trazabilidad:**
El módulo log.c proporciona funciones simples para imprimir información de depuración:

- log_controlador("mensaje")
- log_agente("nombre", "mensaje")

Estas funciones se usan para mostrar en consola:

- registros de conexión
- procesamiento de solicitudes
- errores de comunicación
- eventos importantes del sistema

Esto facilita la verificación del funcionamiento en tiempo real.

f. **Archivo Makefile:**
El Makefile automatiza la compilación de:

- controlador
- agente

Utiliza:

- gcc con el flag –Wall para advertencias
- -pthread para hilos del controlador

El comando principal: make

Genera ambos ejecutables, y:

make clean

Elimina los archivos generados.

g. **Flujo Completo del Sistema:**
   El funcionamiento general puede resumirse así:
   1. El controlador crea el pipe principal y queda escuchando.
   2. Cada agente:
   3. El controlador:
   4. El agente envía solicitudes una por una.
   5. El controlador analiza la disponibilidad por hora, actualiza el estado y responde.
   6. El agente imprime los resultados y continúa.
   7. Al finalizar, envía “fin”.
   8. El controlador registra el cierre del agente y continúa con otros.
   9. Cuando todos los agentes terminan, se libera la memoria y se cierra el sistema.

5. Archivos de prueba y resultados

Se realizo un plan de pruebas con 4 diferentes archivos que presentan los 4 posibles casos que puede presentar el sistema de reservas del parque.

a. **Prueba con archivo que contiene solicitudes aprobadas en las horas solicitadas:**
   Esta prueba valida el flujo exitoso del sistema cuando las reservas pueden ser asignadas en las horas específicas solicitadas por las familias. Se verifica la comunicación correcta entre el agente y el controlador, el procesamiento adecuado de las solicitudes, la actualización del aforo en ambas horas de la estadía, y la confirmación de reserva hacia el agente.

   Entrada:

```
Perez,8,5
Gomez,9,4
Ramirez,10,6
Lopez,13,3
Rojas,15,8
```

   Salida:

```
estudiante@NGEN42:~/Proyecto$ ./agente -s A1 -a solicitudes1.txt -p /tmp/pipe_controlador
[AGENTE A1] Agente de Reservas iniciado
Registro exitoso. Hora actual del sistema: 7
Respuesta para familia Perez: Reserva OK
Respuesta para familia Gomez: Reserva OK
Respuesta para familia Ramirez: Reserva OK
Respuesta para familia Lopez: Reserva OK
Respuesta para familia Rojas: Reserva OK
Agente A1 termina.
```

```
estudiante@NGEN42:~/Proyecto$ ./controlador -i 7 -f 19 -s 2 -t 100 -p /tmp/pipe_controlador
[CONTROLADOR] Controlador de Reservas iniciado
Hora inicial: 7, Hora final: 19
Segundos por hora: 2, Aforo máximo: 100
Esperando solicitudes de agentes...

=== HORA ACTUAL: 7 ===
Personas en el parque a las 7: 0
Agente registrado: A1
Solicitud recibida - Agente: A1, Familia: Perez, Hora: 8, Personas: 5
Resultado del procesamiento para familia Perez: 1
Personas en el parque a las 7: 0

=== HORA ACTUAL: 8 ===
Personas en el parque a las 8: 5
Solicitud recibida - Agente: A1, Familia: Gomez, Hora: 9, Personas: 4
Resultado del procesamiento para familia Gomez: 1
Personas en el parque a las 8: 5

=== HORA ACTUAL: 9 ===
Personas en el parque a las 9: 9
Procesando cambio de hora a 9

=== HORA ACTUAL: 10 ===
Familia Perez sale del parque (5 personas)
Personas en el parque a las 10: 4
Procesando cambio de hora a 10
Solicitud recibida - Agente: A1, Familia: Ramirez, Hora: 10, Personas: 6
Resultado del procesamiento para familia Ramirez: 1
Personas en el parque a las 10: 10

=== HORA ACTUAL: 11 ===
Familia Gomez sale del parque (4 personas)
Personas en el parque a las 11: 6
Procesando cambio de hora a 11
Solicitud recibida - Agente: A1, Familia: Lopez, Hora: 13, Personas: 3
Resultado del procesamiento para familia Lopez: 1
Personas en el parque a las 11: 6

=== HORA ACTUAL: 12 ===
Familia Ramirez sale del parque (6 personas)
Personas en el parque a las 12: 0
Procesando cambio de hora a 12

=== HORA ACTUAL: 13 ===
Personas en el parque a las 13: 3
Procesando cambio de hora a 13
Solicitud recibida - Agente: A1, Familia: Rojas, Hora: 15, Personas: 8
Resultado del procesamiento para familia Rojas: 1
Personas en el parque a las 13: 3
```

```
=== HORA ACTUAL: 14 ===
Personas en el parque a las 14: 3
Procesando cambio de hora a 14
Agente A1 terminó

=== HORA ACTUAL: 15 ===
Familia Lopez sale del parque (3 personas)
Personas en el parque a las 15: 8
Procesando cambio de hora a 15

=== HORA ACTUAL: 16 ===
Personas en el parque a las 16: 8
Procesando cambio de hora a 16
Procesando cambio de hora a 16

=== HORA ACTUAL: 17 ===
Familia Rojas sale del parque (8 personas)
Personas en el parque a las 17: 0

=== HORA ACTUAL: 18 ===
Personas en el parque a las 18: 0
Procesando cambio de hora a 18
Procesando cambio de hora a 18

=== HORA ACTUAL: 19 ===
Personas en el parque a las 19: 0

=== REPORTE FINAL ===
Horas pico (10 personas): 10
Horas bajas (0 personas): 7 12 17 18 19
Solicitudes aprobadas: 5
Solicitudes reprogramadas: 0
Solicitudes negadas: 0
Total solicitudes procesadas: 5
```

b. Prueba para cuando no hay bloques disponibles.

Esta prueba valida que el sistema correctamente identifica cuando no existen bloques de 2 horas consecutivas disponibles dentro del período de simulación.

Entrada:

```
Gonzalez,8,8
Ramirez,9,8
Diaz,10,8
Herrera,8,5
```

Salida:

```
estudiante@NGEN42:~/Proyecto$ ./agente -s A1 -a solicitudes1.txt -p /tmp/pipe_controlador
[AGENTE A1] Agente de Reservas iniciado
Registro exitoso. Hora actual del sistema: 7
Respuesta para familia Gonzalez: Reserva OK
Respuesta para familia Ramirez: Reserva OK
Respuesta para familia Diaz: Reserva OK
Respuesta para familia Herrera: Reserva reprogramada para la hora 12
Agente A1 termina.
```

```
estudiante@NGEN42:~/Proyecto$ ./controlador -i 7 -f 19 -s 2 -t 100 -p /tmp/pipe_controlador
[CONTROLADOR] Controlador de Reservas iniciado
Hora inicial: 7, Hora final: 19
Segundos por hora: 2, Aforo máximo: 100
Esperando solicitudes de agentes...

=== HORA ACTUAL: 7 ===
Personas en el parque a las 7: 0
Agente registrado: A1
Solicitud recibida - Agente: A1, Familia: Gonzalez, Hora: 8, Personas: 8
Resultado del procesamiento para familia Gonzalez: 1
Personas en el parque a las 7: 0

=== HORA ACTUAL: 8 ===
Personas en el parque a las 8: 8

=== HORA ACTUAL: 9 ===
Personas en el parque a las 9: 8
Procesando cambio de hora a 9
Solicitud recibida - Agente: A1, Familia: Ramirez, Hora: 9, Personas: 8
Resultado del procesamiento para familia Ramirez: 1
Personas en el parque a las 9: 16

=== HORA ACTUAL: 10 ===
Familia Gonzalez sale del parque (8 personas)
Personas en el parque a las 10: 8
Procesando cambio de hora a 10
Solicitud recibida - Agente: A1, Familia: Diaz, Hora: 10, Personas: 8
Resultado del procesamiento para familia Diaz: 1
Personas en el parque a las 10: 16

=== HORA ACTUAL: 11 ===
Familia Ramirez sale del parque (8 personas)
Personas en el parque a las 11: 8
Procesando cambio de hora a 11
Procesando cambio de hora a 11

=== HORA ACTUAL: 12 ===
Familia Diaz sale del parque (8 personas)
Personas en el parque a las 12: 0
Solicitud recibida - Agente: A1, Familia: Herrera, Hora: 8, Personas: 5
>>> Reserva REASIGNADA para familia Herrera: hora 12 (solicitó hora 8)
Resultado del procesamiento para familia Herrera: 2
Personas en el parque a las 12: 0

=== HORA ACTUAL: 13 ===
Personas en el parque a las 13: 5
Procesando cambio de hora a 13
Agente A1 terminó
```

```
=== HORA ACTUAL: 14 ===
Personas en el parque a las 14: 0

=== HORA ACTUAL: 15 ===
Personas en el parque a las 15: 0
Procesando cambio de hora a 15
Procesando cambio de hora a 15

=== HORA ACTUAL: 16 ===
Personas en el parque a las 16: 0

=== HORA ACTUAL: 17 ===
Personas en el parque a las 17: 0
Procesando cambio de hora a 17

=== HORA ACTUAL: 18 ===
Personas en el parque a las 18: 0
Procesando cambio de hora a 18

=== HORA ACTUAL: 19 ===
Personas en el parque a las 19: 0
Procesando cambio de hora a 19

=== REPORTE FINAL ===
Horas pico (16 personas): 9 10
Horas bajas (0 personas): 7 12 14 15 16 17 18 19
Solicitudes aprobadas: 3
Solicitudes reprogramadas: 1
Solicitudes negadas: 0
Total solicitudes procesadas: 4
```

c. **Prueba 3 cuando el aforo del parque se supera.**

Entrada:

```
Jimenez,10,500
```

Salida:

```
estudiante@NGEN42:~/Proyecto$ ./agente -s A1 -a solicitudes1.txt -p /tmp/pipe_controlador
[AGENTE A1] Agente de Reservas iniciado
Registro exitoso. Hora actual del sistema: 7
Respuesta para familia Jimenez: Reserva negada: número de personas excede el aforo máximo
Agente A1 termina.
```

```
estudiante@NGEN42:~/Proyecto$ ./controlador -i 7 -f 19 -s 2 -t 20 -p /tmp/pipe_controlador
[CONTROLADOR] Controlador de Reservas iniciado
Hora inicial: 7, Hora final: 19
Segundos por hora: 2, Aforo máximo: 20
Esperando solicitudes de agentes...

=== HORA ACTUAL: 7 ===
Personas en el parque a las 7: 0
Agente registrado: A1
Solicitud recibida - Agente: A1, Familia: Jimenez, Hora: 10, Personas: 500
Resultado del procesamiento para familia Jimenez: 3

=== HORA ACTUAL: 8 ===
Personas en el parque a las 8: 0
Agente A1 terminó

=== HORA ACTUAL: 9 ===
Personas en el parque a las 9: 0

=== HORA ACTUAL: 10 ===
Personas en el parque a las 10: 0

=== HORA ACTUAL: 11 ===
Personas en el parque a las 11: 0

=== HORA ACTUAL: 12 ===
Personas en el parque a las 12: 0

=== HORA ACTUAL: 13 ===
Personas en el parque a las 13: 0

=== HORA ACTUAL: 14 ===
Personas en el parque a las 14: 0

=== HORA ACTUAL: 15 ===
Personas en el parque a las 15: 0

=== HORA ACTUAL: 16 ===
Personas en el parque a las 16: 0
```

```
=== HORA ACTUAL: 17 ===
Personas en el parque a las 17: 0

=== HORA ACTUAL: 18 ===
Personas en el parque a las 18: 0

=== HORA ACTUAL: 19 ===
Personas en el parque a las 19: 0

=== REPORTE FINAL ===
Horas pico (0 personas): 7 8 9 10 11 12 13 14 15 16 17 18 19
Horas bajas (0 personas): 7 8 9 10 11 12 13 14 15 16 17 18 19
Solicitudes aprobadas: 0
Solicitudes reprogramadas: 0
Solicitudes negadas: 1
Total solicitudes procesadas: 1
estudiante@NGEN42:~/Proyecto$
```

d. Prueba 4 cuando hay dos reservas a la misma hora.

Entrada:

```
Gonzalez,8,8
Herrera,8,5
```

Salida:

```
estudiante@NGEN42:~/Proyecto$ ./agente -s A1 -a solicitudes1.txt -p /tmp/pipe_controlador
[AGENTE A1] Agente de Reservas iniciado
Registro exitoso. Hora actual del sistema: 7
Respuesta para familia Gonzalez: Reserva OK
Respuesta para familia Herrera: Reserva reprogramada para la hora 9
```

```
estudiante@NGEN42:~/Proyecto$ ./controlador -i 7 -f 19 -s 2 -t 100 -p /tmp/pipe_controlador
[CONTROLADOR] Controlador de Reservas iniciado
Hora inicial: 7, Hora final: 19
Segundos por hora: 2, Aforo máximo: 100
Esperando solicitudes de agentes...

=== HORA ACTUAL: 7 ===
Personas en el parque a las 7: 0
Agente registrado: A1
Solicitud recibida - Agente: A1, Familia: Gonzalez, Hora: 8, Personas: 8
Resultado del procesamiento para familia Gonzalez: 1
Personas en el parque a las 7: 0

=== HORA ACTUAL: 8 ===
Personas en el parque a las 8: 8

=== HORA ACTUAL: 9 ===
Personas en el parque a las 9: 8
Procesando cambio de hora a 9
Solicitud recibida - Agente: A1, Familia: Herrera, Hora: 8, Personas: 5
>>> Reserva REASIGNADA para familia Herrera: hora 9 (solicitó hora 8)
Resultado del procesamiento para familia Herrera: 2
Personas en el parque a las 9: 8
Procesando cambio de hora a 9

=== HORA ACTUAL: 10 ===
Familia Gonzalez sale del parque (8 personas)
Personas en el parque a las 10: 5
Agente A1 terminó

=== HORA ACTUAL: 11 ===
Personas en el parque a las 11: 0
Procesando cambio de hora a 11
Procesando cambio de hora a 11

=== HORA ACTUAL: 12 ===
Personas en el parque a las 12: 0

=== HORA ACTUAL: 13 ===
Personas en el parque a las 13: 0
Procesando cambio de hora a 13
Procesando cambio de hora a 13

=== HORA ACTUAL: 14 ===
Personas en el parque a las 14: 0

=== HORA ACTUAL: 15 ===
Personas en el parque a las 15: 0
Procesando cambio de hora a 15
```

```
=== HORA ACTUAL: 16 ===
Personas en el parque a las 16: 0
Procesando cambio de hora a 16

=== HORA ACTUAL: 17 ===
Personas en el parque a las 17: 0
Procesando cambio de hora a 17

=== HORA ACTUAL: 18 ===
Personas en el parque a las 18: 0
Procesando cambio de hora a 18

=== HORA ACTUAL: 19 ===
Personas en el parque a las 19: 0
Procesando cambio de hora a 19

=== REPORTE FINAL ===
Horas pico (8 personas): 8 9
Horas bajas (0 personas): 7 11 12 13 14 15 16 17 18 19
Solicitudes aprobadas: 1
Solicitudes reprogramadas: 1
Solicitudes negadas: 0
Total solicitudes procesadas: 2
```

e. Prueba 5 cuando la reserva esta fuera de las horas de disponibilidad del parque.

Entrada:

```
Torres,19,4
Flores,20,3
```

Salida:

```
estudiante@NGEN42:~/Proyecto$ ./agente -s A1 -a solicitudes1.txt -p /tmp/pipe_controlador
[AGENTE A1] Agente de Reservas iniciado
Registro exitoso. Hora actual del sistema: 7
Respuesta para familia Torres: Reserva negada: el parque cierra a las 19:00
Respuesta para familia Flores: Reserva negada: hora fuera del rango de simulación
Agente A1 termina.
```

```
estudiante@NGEN42:~/Proyecto$ ./controlador -i 7 -f 19 -s 2 -t 100 -p /tmp/pipe_controlador
[CONTROLADOR] Controlador de Reservas iniciado
Hora inicial: 7, Hora final: 19
Segundos por hora: 2, Aforo máximo: 100
Esperando solicitudes de agentes...

=== HORA ACTUAL: 7 ===
Personas en el parque a las 7: 0
Agente registrado: A1
Solicitud recibida - Agente: A1, Familia: Torres, Hora: 19, Personas: 4
Resultado del procesamiento para familia Torres: 3
Personas en el parque a las 7: 0

=== HORA ACTUAL: 8 ===
Personas en el parque a las 8: 0
Procesando cambio de hora a 8

=== HORA ACTUAL: 9 ===
Personas en el parque a las 9: 0
Solicitud recibida - Agente: A1, Familia: Flores, Hora: 20, Personas: 3
Resultado del procesamiento para familia Flores: 3
Personas en el parque a las 9: 0

=== HORA ACTUAL: 10 ===
Personas en el parque a las 10: 0
Procesando cambio de hora a 10
Agente A1 terminó

=== HORA ACTUAL: 11 ===
Personas en el parque a las 11: 0
Procesando cambio de hora a 11

=== HORA ACTUAL: 12 ===
Personas en el parque a las 12: 0
Procesando cambio de hora a 12

=== HORA ACTUAL: 13 ===
Personas en el parque a las 13: 0
Procesando cambio de hora a 13

=== HORA ACTUAL: 14 ===
Personas en el parque a las 14: 0
Procesando cambio de hora a 14

=== HORA ACTUAL: 15 ===
Personas en el parque a las 15: 0
Procesando cambio de hora a 15

=== HORA ACTUAL: 16 ===
Personas en el parque a las 16: 0
Procesando cambio de hora a 16
```

```
=== HORA ACTUAL: 17 ===
Personas en el parque a las 17: 0

=== HORA ACTUAL: 18 ===
Personas en el parque a las 18: 0
Procesando cambio de hora a 18

=== HORA ACTUAL: 19 ===
Personas en el parque a las 19: 0
Procesando cambio de hora a 19

=== REPORTE FINAL ===
Horas pico (0 personas): 7 8 9 10 11 12 13 14 15 16 17 18 19
Horas bajas (0 personas): 7 8 9 10 11 12 13 14 15 16 17 18 19
Solicitudes aprobadas: 0
Solicitudes reprogramadas: 0
Solicitudes negadas: 2
Total solicitudes procesadas: 2
```

f. Prueba 6 completa con todo el horario lleno.

Entrada:

```
Gonzalez,8,8
Ramirez,9,8
Diaz,10,8
Herrera,11,7
Pinilla,12,4
Rojas,13,5
Ballesteros,14,2
Corredor,15,7
Villaroel,16,8
Pardo,17,6
Gomez,18,10
```

Salida:

```
estudiante@NGEN42:~/Proyecto$ ./agente -s A1 -a solicitudes1.txt -p /tmp/pipe_controlador
[AGENTE A1] Agente de Reservas iniciado
Registro exitoso. Hora actual del sistema: 7
Respuesta para familia Gonzalez: Reserva OK
Respuesta para familia Ramirez: Reserva OK
Respuesta para familia Diaz: Reserva OK
Respuesta para familia Herrera: Reserva reprogramada para la hora 12
Respuesta para familia Pinilla: Reserva reprogramada para la hora 13
Respuesta para familia Rojas: Reserva reprogramada para la hora 15
Respuesta para familia Ballesteros: Reserva reprogramada para la hora 16
Respuesta para familia Corredor: Reserva reprogramada para la hora 18
Respuesta para familia Villaroel: Reserva negada, debe volver otro día
Respuesta para familia Pardo: Reserva negada, debe volver otro día
Respuesta para familia Gomez: Reserva negada: el parque cierra a las 19:00
Agente A1 termina.
```

```
estudiante@NGEN42:~/Proyecto$ ./controlador -i 7 -f 19 -s 2 -t 100 -p /tmp/pipe_controlador
[CONTROLADOR] Controlador de Reservas iniciado
Hora inicial: 7, Hora final: 19
Segundos por hora: 2, Aforo máximo: 100
Esperando solicitudes de agentes...

=== HORA ACTUAL: 7 ===
Personas en el parque a las 7: 0
Agente registrado: A1
Solicitud recibida - Agente: A1, Familia: Gonzalez, Hora: 8, Personas: 8
Resultado del procesamiento para familia Gonzalez: 1
Personas en el parque a las 7: 0

=== HORA ACTUAL: 8 ===
Personas en el parque a las 8: 8

=== HORA ACTUAL: 9 ===
Personas en el parque a las 9: 8
Solicitud recibida - Agente: A1, Familia: Ramirez, Hora: 9, Personas: 8
Resultado del procesamiento para familia Ramirez: 1
Personas en el parque a las 9: 16

=== HORA ACTUAL: 10 ===
Familia Gonzalez sale del parque (8 personas)
Personas en el parque a las 10: 8
Solicitud recibida - Agente: A1, Familia: Diaz, Hora: 10, Personas: 8
Resultado del procesamiento para familia Diaz: 1
Personas en el parque a las 10: 16

=== HORA ACTUAL: 11 ===
Familia Ramirez sale del parque (8 personas)
Personas en el parque a las 11: 8

=== HORA ACTUAL: 12 ===
Familia Diaz sale del parque (8 personas)
Personas en el parque a las 12: 0
Solicitud recibida - Agente: A1, Familia: Herrera, Hora: 11, Personas: 7
Resultado del procesamiento para familia Herrera: 2
>> Reserva de Herrera reprogramada para la hora 12
Personas en el parque a las 12: 0

=== HORA ACTUAL: 13 ===
Personas en el parque a las 13: 7
Solicitud recibida - Agente: A1, Familia: Pinilla, Hora: 12, Personas: 4
Resultado del procesamiento para familia Pinilla: 2
>> Reserva de Pinilla reprogramada para la hora 13
Personas en el parque a las 13: 0

=== HORA ACTUAL: 14 ===
Familia Herrera sale del parque (7 personas)
Personas en el parque a las 14: 4
```

```
=== HORA ACTUAL: 15 ===
Familia Pinilla sale del parque (4 personas)
Personas en el parque a las 15: 0
Solicitud recibida - Agente: A1, Familia: Rojas, Hora: 13, Personas: 5
Resultado del procesamiento para familia Rojas: 2
>> Reserva de Rojas reprogramada para la hora 15
Personas en el parque a las 15: 0

=== HORA ACTUAL: 16 ===
Personas en el parque a las 16: 5
Solicitud recibida - Agente: A1, Familia: Ballesteros, Hora: 14, Personas: 2
Resultado del procesamiento para familia Ballesteros: 2
>> Reserva de Ballesteros reprogramada para la hora 16
Personas en el parque a las 16: 0

=== HORA ACTUAL: 17 ===
Familia Rojas sale del parque (5 personas)
Personas en el parque a las 17: 2

=== HORA ACTUAL: 18 ===
Familia Ballesteros sale del parque (2 personas)
Personas en el parque a las 18: 0
Solicitud recibida - Agente: A1, Familia: Corredor, Hora: 15, Personas: 7
Resultado del procesamiento para familia Corredor: 2
>> Reserva de Corredor reprogramada para la hora 18
Personas en el parque a las 18: 0

=== HORA ACTUAL: 19 ===
Personas en el parque a las 19: 7
Solicitud recibida - Agente: A1, Familia: Villaroel, Hora: 16, Personas: 8
Resultado del procesamiento para familia Villaroel: 3
Personas en el parque a las 19: 0
Día terminado. Esperando que agentes finalicen...
Solicitud recibida - Agente: A1, Familia: Pardo, Hora: 17, Personas: 6
Resultado del procesamiento para familia Pardo: 3
Personas en el parque a las 20: 0
Solicitud recibida - Agente: A1, Familia: Gomez, Hora: 18, Personas: 10
Resultado del procesamiento para familia Gomez: 3
Personas en el parque a las 20: 0
Agente A1 terminó

=== REPORTE FINAL ===
Horas pico (16 personas): 9 10
Horas bajas (0 personas): 7 12 13 15 16 18 19
Solicitudes aprobadas: 3
Solicitudes reprogramadas: 5
Solicitudes negadas: 1
Total solicitudes procesadas: 9
```

g. Prueba 7 con dos agentes al mismo tiempo.

Esta prueba busca verificar el funcionamiento del sistema con múltiples agentes procesando solicitudes de forma concurrente durante un día completo de operación.

Se crean dos archivos de solicitudes (solicitudes1.txt, solicitudes2.txt)

```
Piñeros,8,8
Corredor,9,8
Barreto,10,8
Martinez,8,5
Rojas,17,8
Pinilla,15,8
Ballesteros,14,8
Paez,13,5
Avila,20,8
Rios,18,8
```

```
Gonzalez,8,8
Ramirez,9,8
Diaz,10,8
Herrera,8,5
Martinez,7,20
Suarez,9,2
Benitez,8,3
```

Agente 1

```
estudiante@NGEN42:~/Proyecto$ ./agente -s A1 -a solicitudes1.txt -p /tmp/pipe_controlador
[AGENTE A1] Agente de Reservas iniciado
Registro exitoso. Hora actual del sistema: 7
Respuesta para familia Piñeros: Reserva OK
Respuesta para familia Corredor: Reserva OK
Respuesta para familia Barreto: Reserva reprogramada para la hora 11
Respuesta para familia Martinez: Reserva reprogramada para la hora 12
Respuesta para familia Rojas: Reserva OK
Respuesta para familia Pinilla: Reserva OK
Respuesta para familia Ballesteros: Reserva reprogramada para la hora 16
Respuesta para familia Paez: Reserva reprogramada para la hora 18
Respuesta para familia Avila: Reserva negada: el parque cierra a las 19:00
Respuesta para familia Rios: Reserva negada: el parque cierra a las 19:00
Agente A1 termina.
```

Agente 2

```
estudiante@NGEN42:~/Proyecto$ ./agente -s A2 -a solicitudes2.txt -p /tmp/pipe_controlador
[AGENTE A2] Agente de Reservas iniciado
Registro exitoso. Hora actual del sistema: 7
Respuesta para familia Gonzalez: Reserva OK
Respuesta para familia Ramirez: Reserva OK
Respuesta para familia Diaz: Reserva OK
Respuesta para familia Herrera: Reserva reprogramada para la hora 11
Respuesta para familia Martinez: Reserva reprogramada para la hora 12
Respuesta para familia Suarez: Reserva reprogramada para la hora 13
Respuesta para familia Benitez: Reserva reprogramada para la hora 15
Agente A2 termina.
```

Controlador

```
estudiante@NGEN42:~/Proyecto$ ./controlador -i 7 -f 19 -s 2 -t 100 -p /tmp/pipe_controlador
[CONTROLADOR] Controlador de Reservas iniciado
Hora inicial: 7, Hora final: 19
Segundos por hora: 2, Aforo máximo: 100
Esperando solicitudes de agentes...

=== HORA ACTUAL: 7 ===
Personas en el parque a las 7: 0
Agente registrado: A2
Agente registrado: A1
Solicitud recibida - Agente: A2, Familia: Gonzalez, Hora: 8, Personas: 8
Resultado del procesamiento para familia Gonzalez: 1
Personas en el parque a las 7: 0

=== HORA ACTUAL: 8 ===
Personas en el parque a las 8: 8
Solicitud recibida - Agente: A1, Familia: Piñeros, Hora: 8, Personas: 8
Resultado del procesamiento para familia Piñeros: 1
Personas en el parque a las 8: 16
Solicitud recibida - Agente: A2, Familia: Ramirez, Hora: 9, Personas: 8
Resultado del procesamiento para familia Ramirez: 1
Personas en el parque a las 8: 16

=== HORA ACTUAL: 9 ===
Personas en el parque a las 9: 24
Solicitud recibida - Agente: A1, Familia: Corredor, Hora: 9, Personas: 8
Resultado del procesamiento para familia Corredor: 1
Personas en el parque a las 9: 32

=== HORA ACTUAL: 10 ===
Familia Gonzalez sale del parque (8 personas)
Familia Piñeros sale del parque (8 personas)
Personas en el parque a las 10: 16
Solicitud recibida - Agente: A2, Familia: Diaz, Hora: 10, Personas: 8
Resultado del procesamiento para familia Diaz: 1
Personas en el parque a las 10: 24

=== HORA ACTUAL: 11 ===
Familia Ramirez sale del parque (8 personas)
Familia Corredor sale del parque (8 personas)
Personas en el parque a las 11: 8
Solicitud recibida - Agente: A1, Familia: Barreto, Hora: 10, Personas: 8
Resultado del procesamiento para familia Barreto: 2
>> Reserva de Barreto reprogramada para la hora 11
Personas en el parque a las 11: 8
Solicitud recibida - Agente: A2, Familia: Herrera, Hora: 8, Personas: 5
Resultado del procesamiento para familia Herrera: 2
>> Reserva de Herrera reprogramada para la hora 11
Personas en el parque a las 11: 8
```

```
=== HORA ACTUAL: 12 ===
Familia Diaz sale del parque (8 personas)
Personas en el parque a las 12: 13
Solicitud recibida - Agente: A1, Familia: Martinez, Hora: 8, Personas: 5
Resultado del procesamiento para familia Martinez: 2
>> Reserva de Martinez reprogramada para la hora 12
Personas en el parque a las 12: 0
Solicitud recibida - Agente: A2, Familia: Martinez, Hora: 7, Personas: 20
Resultado del procesamiento para familia Martinez: 2
>> Reserva de Martinez reprogramada para la hora 12
Personas en el parque a las 12: 0

=== HORA ACTUAL: 13 ===
Familia Barreto sale del parque (8 personas)
Familia Herrera sale del parque (5 personas)
Personas en el parque a las 13: 25
Solicitud recibida - Agente: A1, Familia: Rojas, Hora: 17, Personas: 8
Resultado del procesamiento para familia Rojas: 1
Personas en el parque a las 13: 0
Solicitud recibida - Agente: A2, Familia: Suarez, Hora: 9, Personas: 2
Resultado del procesamiento para familia Suarez: 2
>> Reserva de Suarez reprogramada para la hora 13
Personas en el parque a las 13: 0

=== HORA ACTUAL: 14 ===
Familia Martinez sale del parque (5 personas)
Familia Martinez sale del parque (20 personas)
Personas en el parque a las 14: 2

=== HORA ACTUAL: 15 ===
Familia Suarez sale del parque (2 personas)
Personas en el parque a las 15: 0
Solicitud recibida - Agente: A1, Familia: Pinilla, Hora: 15, Personas: 8
Resultado del procesamiento para familia Pinilla: 1
Personas en el parque a las 15: 8
Solicitud recibida - Agente: A2, Familia: Benitez, Hora: 8, Personas: 3
Resultado del procesamiento para familia Benitez: 2
>> Reserva de Benitez reprogramada para la hora 15
Personas en el parque a las 15: 8

=== HORA ACTUAL: 16 ===
Personas en el parque a las 16: 11
Solicitud recibida - Agente: A1, Familia: Ballesteros, Hora: 14, Personas: 8
Resultado del procesamiento para familia Ballesteros: 2
>> Reserva de Ballesteros reprogramada para la hora 16
Personas en el parque a las 16: 8
Agente A2 terminó
```

```
=== HORA ACTUAL: 17 ===
Familia Pinilla sale del parque (8 personas)
Familia Benitez sale del parque (3 personas)
Personas en el parque a las 17: 16

=== HORA ACTUAL: 18 ===
Familia Ballesteros sale del parque (8 personas)
Personas en el parque a las 18: 8
Solicitud recibida - Agente: A1, Familia: Paez, Hora: 13, Personas: 5
Resultado del procesamiento para familia Paez: 2
>> Reserva de Paez reprogramada para la hora 18
Personas en el parque a las 18: 8

=== HORA ACTUAL: 19 ===
Familia Rojas sale del parque (8 personas)
Personas en el parque a las 19: 5
Solicitud recibida - Agente: A1, Familia: Avila, Hora: 20, Personas: 8
Resultado del procesamiento para familia Avila: 3
Personas en el parque a las 19: 0
Día terminado. Esperando que agentes finalicen...
Solicitud recibida - Agente: A1, Familia: Rios, Hora: 18, Personas: 8
Resultado del procesamiento para familia Rios: 3
Personas en el parque a las 20: 0
Agente A1 terminó

=== REPORTE FINAL ===
Horas pico (32 personas): 9
Horas bajas (0 personas): 7 12 13 19
Solicitudes aprobadas: 7
Solicitudes reprogramadas: 8
Solicitudes negadas: 2
Total solicitudes procesadas: 17
```

6. Análisis

Tras haber realizado la ejecución del plan de pruebas propuesto para el proyecto, se observan resultados consistentes con los objetivos planteados para este. A continuación, pasaremos a realizar un análisis a estos resultados obtenidos.

En primer lugar, se evidencio que la arquitectura controlador-agentes funciono de manera correcta y acertada, logrando establecer un canal de comunicación unidireccional donde el agente le envía solicitudes a un controlador, estas solicitudes son las reservas realizadas por los clientes del parque, y un canal independiente para las respuestas. También, se evidencio un registro exitoso de los agentes en el sistema para lograr establecer la comunicación con el controlador, es decir, el sistema de reservas. Este comportamiento confirma que la implementación mediante el uso de pipes FIFO fue exitosa y que los mensajes estructurados se transmitieron de manera íntegra, es decir que no se perdió información.

En relación con la administración del parque, el controlador logro validar horarios, se gestionó de manera óptima el aforo y se logró asignar reservas de manera satisfactoria de acuerdo con la disponibilidad, esto se evidencio en los resultados en la medida que:

- Las solicitudes validas dentro del aforo fueron aprobadas.
- Las solicitudes que excedían el aforo fueron reprogramadas a la hora disponible más cercana.
- Las solicitudes fuera del rango del tiempo operativo del parque fueron rechazadas.
- Cuando el aforo estaba saturado, el sistema evito que se sobrepasara la capacidad máxima, ya sea rechazando o reprogramando las solicitudes.

El uso de hilos POSIX, usados para simular el reloj interno del sistema del parque, también produjo resultados satisfactorios y coherentes. El hilo encargado del tiempo avanzo la hora en los intervalos establecidos.

Por otra parte, las pruebas de concurrencia revelaron que el uso de mutex para proteger el acceso a estructuras compartidas evito que se generaran condiciones de carrera, que de haberse generado hubieran sido perjudiciales para el buen funcionamiento, prácticamente dejándolo sin funcionar. Esto debido a que como se evidencio, se envían varias solicitudes de manera simultánea por diferentes actores, por lo tanto, se necesitaba de una correcta aplicación de mecanismo de sincronización, lo cual se realizó de manera satisfactoria.

7. Conclusiones

Luego de realizar el análisis de los resultados obtenidos, podemos concluir que el desarrollo del sistema de reservas permitió integrar de manera exitosa múltiples conceptos fundamentales de los Sistemas Operativos vistos a lo largo del semestre, demostrando la importancia de la comunicación entre procesos, la concurrencia controlada y la correcta administración de recursos compartidos. A lo largo de la realización del proyecto, se implementó una arquitectura del tipo cliente-servidor robusta, perfecta para la resolución del problema planteado, la cual coordina agentes haciendo uso de pipes, mientras que el controlador hace uso de hilos para el manejo interno del reloj. Además, el uso adecuado de mecanismos de sincronización se mostró sumamente necesario para evitar condiciones de carrera que pueden amenazar con el buen funcionamiento del proyecto. Las pruebas realizadas evidenciaron que el sistema es capaz de manejar diferentes tipos de situaciones y número de solicitudes realizadas, resolviendo el problema planteado, cumpliendo con los objetivos planteados.

8. Referencias

- The Open Group. (2018). *POSIX.1-2017: Base Specifications, Issue 7*. https://pubs.opengroup.org/onlinepubs/9699919799/

- GeeksforGeeks. (2025, 12 de julio). *Mutual Exclusion in Synchronization*. GeeksforGeeks. https://www.geeksforgeeks.org/operating-systems/mutual-exclusion-in-synchronization/

- Carnegie Mellon University. (s. f.). *POSIX Threads (Pthreads) — Linux Tutorial*. https://www.cs.cmu.edu/afs/cs/academic/class/15492-f07/www/pthreads.html

- Microsoft. (2024). *How to use named pipes for network interprocess communication*. Microsoft Learn. https://learn.microsoft.com/es-es/dotnet/standard/io/how-to-use-named-pipes-for-network-interprocess-communication