Licenciatura en Ingeniería en Ciencias de la Computación

Sistemas Operativos

Sección: 10

Catedrático: Alvaro Sebastián Galindo

Proyecto#2 Simulador

Javier Chen – 22153
Gustavo Cruz -22779

4 de junio del 2025

Introducción

En este proyecto se desarrolla una aplicación con interfaz gráfica y lógica en C/C++ que permite simular de manera visual y dinámica algoritmos de planificación de procesos y mecanismos de sincronización. El objetivo es reforzar los conocimientos en concurrencia, planificación, uso de mutex y semáforos, además del manejo de archivos y programación defensiva.

Objetivos

- Implementar diferentes algoritmos de planificación de procesos.
- Visualizar de forma dinámica la ejecución de los procesos.
- Simular el acceso a recursos mediante mecanismos de sincronización.
- Ofrecer una interfaz intuitiva que permita seleccionar configuraciones, cargar archivos y ejecutar las simulaciones.

Calendarización y sincronización:

Calendarización: La calendarización (también llamada planificación de procesos) es el mecanismo que utiliza un sistema operativo para decidir qué proceso se ejecuta y en qué momento, especialmente cuando hay múltiples procesos listos para ejecutarse.

Objetivo principal:

Optimizar el uso del procesador y mejorar el rendimiento general del sistema.

Factores que considera un algoritmo de planificación:

- Tiempo de llegada de cada proceso (Arrival Time)
- Duración de ejecución (Burst Time)
- Prioridad asignada
- Quantum de tiempo (en algoritmos como Round Robin)

Sincronización: La sincronización se refiere a los mecanismos que permiten controlar el acceso concurrente a recursos compartidos entre varios procesos o hilos, evitando

conflictos como condiciones de carrera (race conditions), inconsistencia de datos, y bloqueos mutuos (deadlocks).

Situación típica:

Varios procesos desean leer o escribir un recurso, pero si lo hacen simultáneamente sin coordinación, los resultados pueden ser incorrectos o caóticos.

Estados en la sincronización:

- ACCESSED: El proceso pudo acceder exitosamente al recurso.
- WAITING: El proceso está bloqueado esperando que el recurso esté disponible.

Algoritmos:

Algoritmos de Calendarización:

First In First Out (FIFO): es uno de los algoritmos de planificación más simples. En este método, los procesos se ejecutan en el orden exacto en que llegan al sistema, sin considerar su duración ni prioridad. Una vez que un proceso comienza a ejecutarse, no puede ser interrumpido hasta que finalice. Aunque es fácil de implementar, puede generar tiempos de espera elevados, especialmente cuando un proceso largo bloquea a otros más cortos que llegaron después.

Shortest Job First (SJF): es un algoritmo que da preferencia a los procesos que tienen el menor tiempo de ejecución estimado. Este algoritmo es eficiente porque reduce el tiempo promedio de espera para todos los procesos. Sin embargo, su principal limitación es que requiere conocer previamente la duración de cada proceso, lo cual no siempre es posible. Además, puede causar starvation, ya que los procesos largos podrían quedar esperando indefinidamente si continuamente llegan procesos más cortos.

Shortest Remaining Time (SRT): es una versión preventiva del algoritmo SJF. En este caso, si un proceso nuevo llega al sistema y su tiempo de ejecución restante es menor que el del proceso en ejecución, este último se interrumpe para dar paso al nuevo. Este

enfoque mejora aún más el tiempo de respuesta promedio, pero aumenta la complejidad del sistema y también puede causar starvation en procesos largos.

Round Robin (RR): es un algoritmo diseñado para compartir el tiempo de CPU de manera equitativa entre todos los procesos. A cada uno se le asigna un intervalo fijo de tiempo, llamado quantum, durante el cual puede ejecutarse. Si el proceso no finaliza dentro de su quantum, se coloca al final de la cola y se le dará otra oportunidad más adelante. Este enfoque es ideal para sistemas interactivos, pero la eficiencia del sistema depende en gran medida del tamaño del quantum elegido.

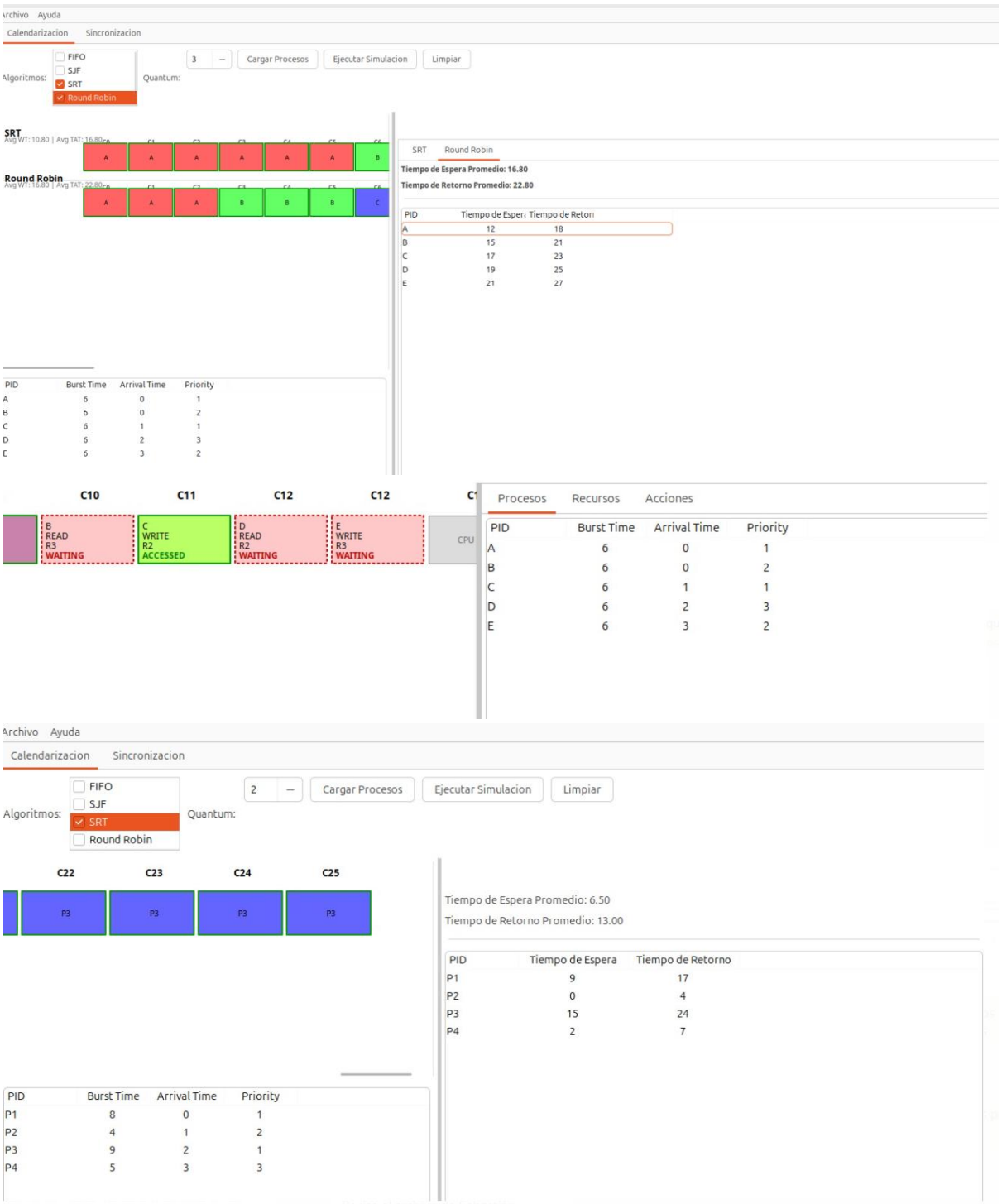
Priority Scheduling; asigna a cada proceso un nivel de prioridad, y el sistema siempre ejecuta el proceso con la prioridad más alta disponible. Este algoritmo puede ser preventivo o no, y permite atender primero tareas críticas o importantes. No obstante, presenta el riesgo de que los procesos con baja prioridad nunca lleguen a ejecutarse si constantemente llegan procesos con mayor prioridad, problema conocido como starvation, que puede resolverse aplicando técnicas como el envejecimiento (aging).

Mecanismos de Sincronización:

Mutex Locks : son un mecanismo de sincronización utilizado para evitar que múltiples procesos o hilos accedan al mismo recurso crítico de forma simultánea. Un mutex permite que solo un proceso tenga acceso al recurso en un momento dado; los demás deben esperar a que el proceso actual libere el mutex. Este mecanismo es simple y eficaz para proteger secciones críticas, pero si no se gestiona correctamente, puede provocar bloqueos o condiciones de inactividad entre procesos (deadlocks).

Semáforos: son mecanismos de sincronización más generales y versátiles que los mutex. Están basados en un contador que representa el número de unidades disponibles de un recurso. Los semáforos pueden ser binarios (como los mutex, con valores 0 o 1) o contadores (permitiendo múltiples accesos simultáneos hasta un límite). Se utilizan operaciones estándar como wait (también llamada P) para solicitar acceso y signal (o V) para liberar el recurso. Los semáforos permiten coordinar el acceso concurrente a recursos compartidos de manera más flexible, pero también pueden introducir problemas si no se implementan con cuidado, como condiciones de carrera o bloqueos mutuos.

Capturas de Pantalla



Conclusiones

- El desarrollo de este simulador permitió implementar una herramienta funcional, visual e interactiva, que facilita la comprensión profunda de los conceptos fundamentales de planificación de procesos y sincronización en sistemas operativos. Gracias a su diseño modular y flexible, la aplicación permite cargar distintos conjuntos de datos y observar en tiempo real el comportamiento de diversos algoritmos, lo que la convierte en un recurso educativo valioso.
- La interfaz gráfica intuitiva posibilita la comparación directa entre algoritmos de calendarización, analizando métricas de eficiencia como el tiempo promedio de espera. Asimismo, la simulación dinámica de mecanismos de sincronización como mutex y semáforos permite observar visualmente el acceso a recursos críticos y los efectos de la concurrencia, incluyendo situaciones de espera o bloqueo.
- Este proyecto no solo fortaleció conocimientos teóricos, sino que también brindó experiencia práctica en programación concurrente, manejo de estructuras de datos, diseño de interfaces y simulación de comportamientos complejos del sistema operativo. En conjunto, se logró crear una plataforma didáctica que apoya el aprendizaje significativo de temas clave en el área de sistemas operativos.