Simulación de Algoritmos de Planificación de CPU

Informe Técnico

Proyecto Final – Sistemas Operativos

Universidad Distrital Francisco José de Caldas

Xiomara Arias
Código: 20222020028

Alejandro Mora
Código: 20222020005

11 de diciembre de 2025

Índice general

Capítulo 1

Introducción

La planificación de CPU es un componente fundamental de los sistemas operativos modernos, responsable de decidir qué proceso debe utilizar el procesador en cada instante de tiempo. Debido a que la CPU es un recurso limitado y altamente demandado, es necesario implementar estrategias que permitan maximizar su uso, reducir tiempos de espera y garantizar un comportamiento eficiente y predecible del sistema.

El presente informe describe el funcionamiento teórico y conceptual de los algoritmos de planificación más utilizados: FCFS, SJF, SRTF y Round Robin. Además, se presenta el diseño general de un simulador desarrollado para observar su comportamiento, analizar métricas relevantes y visualizar cronogramas de ejecución (Gantt).

El documento se dirige a estudiantes y desarrolladores que deseen comprender en profundidad cómo operan estos algoritmos y cómo se comportan los procesos dentro del sistema.

Capítulo 2

Marco Teórico

2.1. Procesos y Ciclo de Vida

Un proceso es un programa en ejecución que posee un estado propio, un contador de programa y un conjunto de recursos asignados (memoria, archivos abiertos, registros, entre otros). El sistema operativo administra estos procesos y controla su transición entre estados según su comportamiento y necesidades.
El ciclo de vida clásico de un proceso incluye los siguientes estados:

- **Nuevo**: el proceso está siendo creado e inicializado.
- **Listo**: preparado para ejecutarse, esperando a que el planificador le asigne la CPU.
- **Ejecutando**: actualmente utilizando el procesador para avanzar su ráfaga de CPU.
- **Bloqueado**: esperando la finalización de una operación externa (generalmente E/S).
- **Terminado**: el proceso finalizó su ejecución o fue terminado por el sistema.

Cada transición entre estados es gestionada por el sistema operativo, lo que permite mantener múltiples procesos avanzando de manera ordenada y eficiente.

2.2. Planificadores en un Sistema Operativo

Para administrar múltiples procesos, los sistemas operativos implementan diferentes niveles de planificación:

- **Planificador a largo plazo**: controla el grado de multiprogramación. Decide qué procesos pasan del almacenamiento secundario a la memoria principal, afectando la carga general del sistema.

- **Planificador a mediano plazo**: gestiona la *swapping*, es decir, la suspensión y reanudación de procesos para liberar memoria cuando sea necesario.
- **Planificador a corto plazo (CPU scheduler)**: es el encargado de elegir, entre los procesos listos, cuál obtiene la CPU en el siguiente instante. Opera de manera muy frecuente y determina el comportamiento observable del sistema.

Este proyecto se enfoca específicamente en el planificador a corto plazo, ya que es el componente que define cómo se simula el funcionamiento de los algoritmos de planificación.

2.3. Concurrencia, Multiprogramación y Expropiación

2.3.1. Multiprogramación

La multiprogramación busca mantener múltiples procesos cargados en memoria para aumentar el uso eficiente del procesador. Mientras un proceso está bloqueado por una operación de E/S, otro puede ejecutarse, reduciendo los tiempos muertos.

2.3.2. Concurrencia

La concurrencia describe la ilusión de que varios procesos se ejecutan de forma simultánea. En sistemas con una sola CPU, esto se logra mediante cambios rápidos de contexto. En sistemas con múltiples núcleos, parte de esta concurrencia puede ser real, pero los principios siguen siendo los mismos.

2.3.3. Expropiación (Preemption)

Un algoritmo de planificación es **expropiativo** si puede interrumpir un proceso en ejecución antes de que termine su ráfaga de CPU. Esto permite responder mejor a procesos interactivos y mejorar tiempos de respuesta.

- **Ejemplos de algoritmos expropiativos**: Round Robin, SRTF, Prioridad expropiativa.
- **Ejemplos de algoritmos no expropiativos**: FCFS, SJF no expropiativo, Prioridad no expropiativa.

La expropiación introduce un costo adicional: el cambio de contexto. Sin embargo, mejora significativamente la equidad y la capacidad de respuesta del sistema.

2.4. Bloqueos por Operaciones de E/S

Muchos procesos no ejecutan únicamente instrucciones de CPU. En algún punto de su ejecución pueden requerir operaciones de entrada/salida, como lectura de disco, acceso a red o interacción con periféricos. Durante estas operaciones el proceso no puede continuar ejecutándose y pasa a un estado de bloqueo.
Por esta razón, un algoritmo de planificación debe tener en cuenta:

- **Momento de inicio del bloqueo:** el instante en que el proceso deja de usar la CPU para realizar la operación de E/S.
- **Duración del bloqueo:** el tiempo que la operación tardará en completarse.
- **Reincorporación a la cola listo:** una vez finalizada la E/S, el proceso debe volver al conjunto de procesos aptos para recibir nuevamente CPU.

Durante el período en que un proceso está bloqueado, el planificador continúa asignando CPU a otros procesos sin interrupción. Esto permite mantener un uso eficiente del procesador y evita tiempos muertos. La correcta administración de estos bloqueos es esencial para simular adecuadamente el comportamiento real de un sistema operativo, donde las operaciones de E/S suelen representar una parte significativa del tiempo total de ejecución de un proceso.

Capítulo 3

Algoritmos de Planificación

Los algoritmos de planificación determinan el orden en que los procesos reciben tiempo de CPU. Su propósito principal es optimizar el rendimiento del sistema bajo criterios como tiempo de espera, tiempo de respuesta, utilización del procesador y equidad entre procesos. Esta sección describe en profundidad los algoritmos implementados en el simulador: FCFS, SJF, SRTF y Round Robin, junto con sus fundamentos, ventajas, desventajas y comportamiento general.

3.1. First Come, First Served (FCFS)

3.1.1. Descripción General

El algoritmo **First Come, First Served (FCFS)** es el más simple y directo: los procesos son atendidos en el mismo orden en que llegan a la cola de listos. Se basa en una estructura FIFO (First In, First Out), lo que garantiza que el primer proceso que solicite CPU será también el primero en ejecutarse.
FCFS es un algoritmo **no expropiativo**: una vez un proceso obtiene la CPU, la mantiene hasta completar su ráfaga o entrar en espera por E/S.

3.1.2. Comportamiento y Características

- Tiende a favorecer procesos con ráfagas largas que llegan temprano.
- El tiempo de respuesta para procesos cortos puede ser muy alto si llegan después de un proceso extenso.
- Se caracteriza por el fenómeno llamado **efecto convoy**: un proceso largo al inicio puede retrasar a todos los demás.

3.1.3. Ventajas

- Fácil de implementar.
- Justo en términos de orden de llegada.
- No requiere información adicional como estimaciones de ráfagas.

3.1.4. Desventajas

- Puede generar tiempos de espera muy altos.
- Afectado severamente por el efecto convoy.
- No es adecuado para sistemas interactivos donde importa la respuesta rápida.

3.2. Shortest Job First (SJF)

3.2.1. Descripción General

El algoritmo **Shortest Job First (SJF)** selecciona el proceso con la ráfaga de CPU más corta entre los disponibles. Es un algoritmo óptimo en términos de tiempo de espera promedio: ningún otro algoritmo no expropiativo puede superar ese rendimiento bajo condiciones ideales.

3.2.2. Requerimientos y Limitaciones

SJF requiere conocer la duración de la siguiente ráfaga de CPU, lo cual no es posible en la práctica. En sistemas reales, se utilizan estimaciones basadas en comportamiento previo mediante:

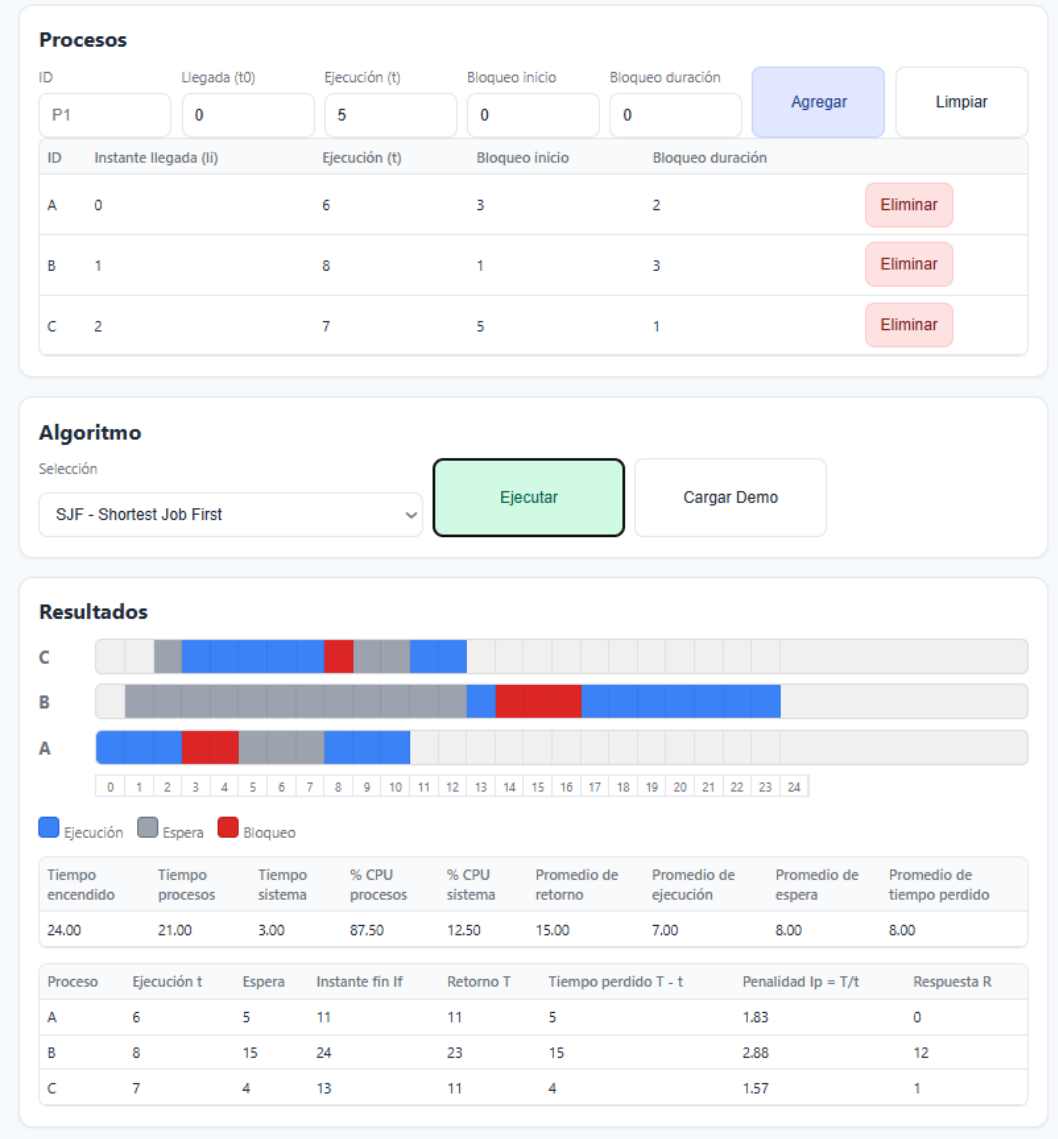
- Media exponencialmente ponderada.
- Historial de ráfagas anteriores.

3.2.3. Ventajas

- Minimiza el tiempo de espera promedio.
- Reduce significativamente los tiempos de retorno.

3.2.4. Desventajas

- Dificultad para estimar correctamente las ráfagas.
- Puede generar **inanición** para procesos largos.
- No es expropiativo (su variante expropiativa es SRTF).



3.3. Shortest Remaining Time First (SRTF)

3.3.1. Descripción General

El algoritmo **Shortest Remaining Time First (SRTF)** es la versión expropiativa de SJF. En este caso, el proceso que se selecciona es el que tiene la menor ráfaga restante de CPU.

Cada vez que llega un nuevo proceso, el planificador compara su duración con el proceso en ejecución. Si la ráfaga del nuevo proceso es menor, el proceso actual es interrumpido.

3.3.2. Características Clave

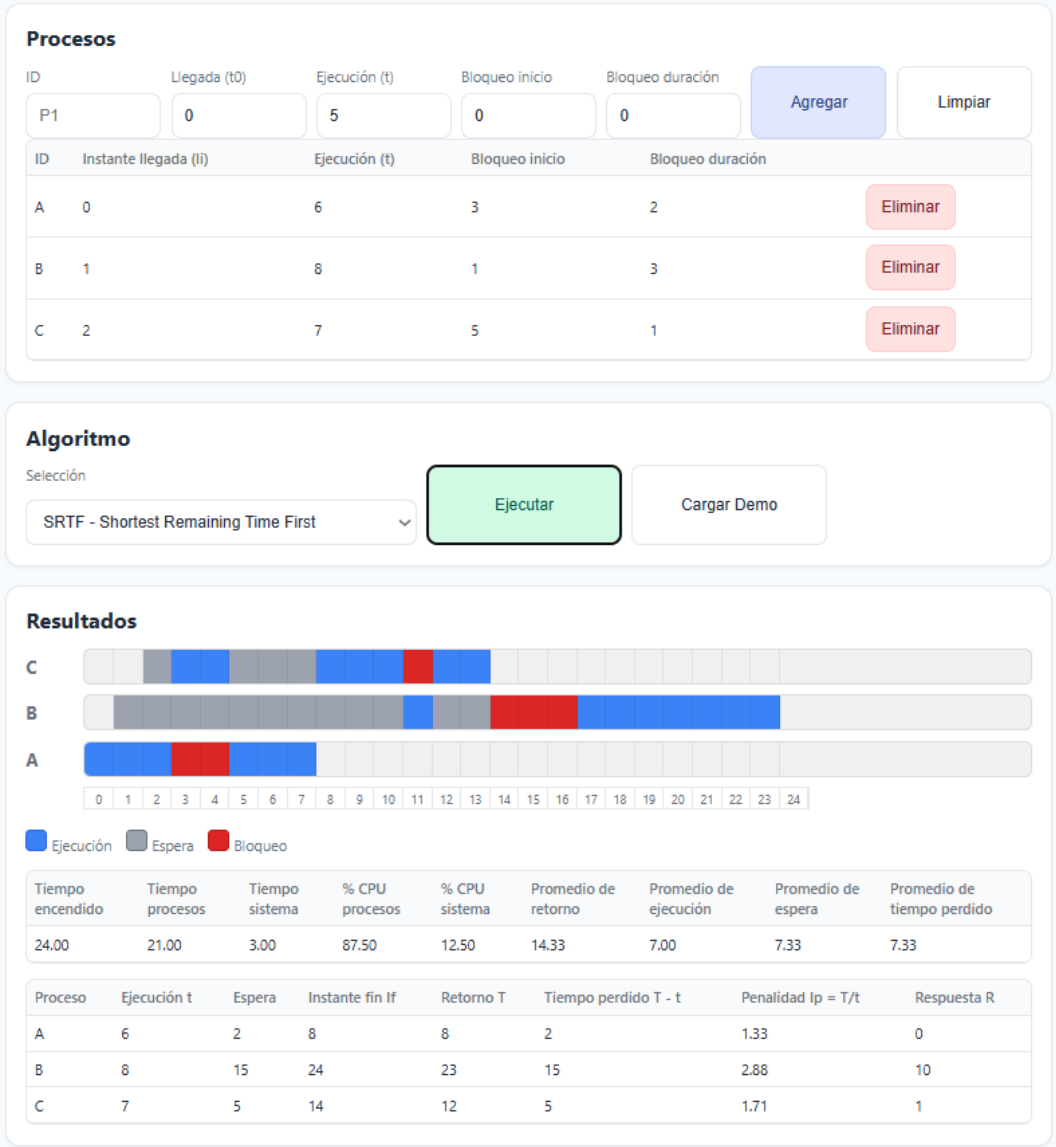
- Es un algoritmo altamente reactivo.
- Minimiza el tiempo de respuesta para procesos cortos.
- Beneficia fuertemente a procesos pequeños e interactivos.

3.3.3. Ventajas

- Mejor tiempo de respuesta entre todos los algoritmos evaluados.
- Excelente rendimiento en multiprogramación.

3.3.4. Desventajas

- Alta probabilidad de inanición para procesos largos.
- Mayor complejidad de implementación.
- Requiere estimaciones precisas o información confiable sobre ráfagas.



3.4. Round Robin (RR)

3.4.1. Descripción General

El algoritmo **Round Robin (RR)** fue diseñado específicamente para sistemas interactivos y de tiempo compartido. Asigna un tiempo fijo de ejecución a cada proceso, conocido como

quantum.
Una vez el proceso consume su quantum, es expropiado y enviado al final de la cola de listos.

3.4.2. Funcionamiento

Este algoritmo mantiene un ciclo continuo:

1. Seleccionar el primer proceso de la cola.
2. Ejecutarlo durante un quantum.
3. Si finaliza, se retira.
4. Si no, regresa al final de la cola.

3.4.3. Selección del Quantum

La elección del quantum es crítica:

- **Quantum pequeño:** Alta interactividad, pero provoca mayor overhead por cambio de contexto.
- **Quantum grande:** Reduce overhead, pero se comporta como FCFS.

3.4.4. Ventajas

- Altamente justo: todos reciben CPU regularmente.
- Ideal para sistemas multitarea e interactivos.
- Evita inanición.

3.4.5. Desventajas

- Puede aumentar el tiempo de retorno.
- El overhead puede ser significativo si el quantum es muy pequeño.
- El rendimiento depende fuertemente del valor del quantum.

Procesos

ID	Llegada (t0)	Ejecución (t)	Bloqueo inicio	Bloqueo duración		
P1	0	5	0	0	Agregar	Limpiar

ID	Instante llegada (Ii)	Ejecución (t)	Bloqueo inicio	Bloqueo duración	
A	0	6	3	2	Eliminar
B	1	8	1	3	Eliminar
C	2	7	5	1	Eliminar

Q = 4

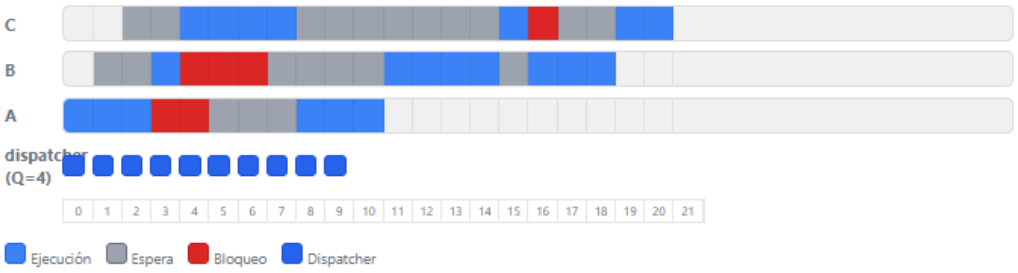
Algoritmo

Selección: RR - Round Robin

Quantum: 4

Ejecutar | Cargar Demo

Resultados



Tiempo encendido	Tiempo procesos	Tiempo sistema	% CPU procesos	% CPU sistema	Promedio de retorno	Promedio de ejecución	Promedio de espera	Promedio de tiempo perdido
21.00	21.00	0.00	100.00	0.00	16.00	7.00	9.00	9.00

Proceso	Ejecución t	Espera	Instante fin If	Retorno T	Tiempo perdido T - t	Penalidad Ip = T/t	Respuesta R
A	6	5	11	11	5	1.83	0
B	8	10	19	18	10	2.25	2
C	7	12	21	19	12	2.71	2

Capítulo 4

Diseño del Simulador

4.1. Modelo del Proceso

Cada proceso se modela con información esencial:

- Identificador
- Tiempo de llegada
- Duración de ráfaga
- Bloqueo programado (inicio y duración)
- Tiempos calculados: inicio, fin, respuesta, espera, retorno

Diagramas UML

Diagrama de Casos de Uso

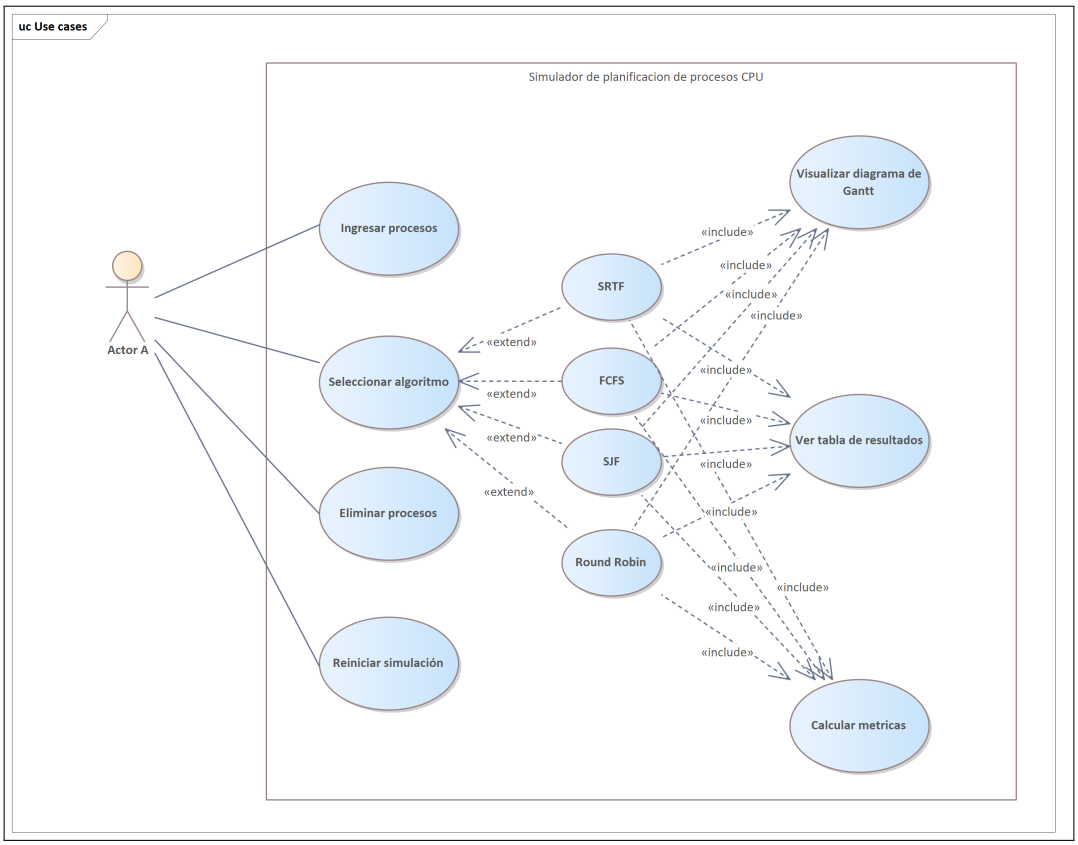


Figura 4.1: Diagrama de Casos de Uso del Sistema de Simulación

Diagrama de Clases

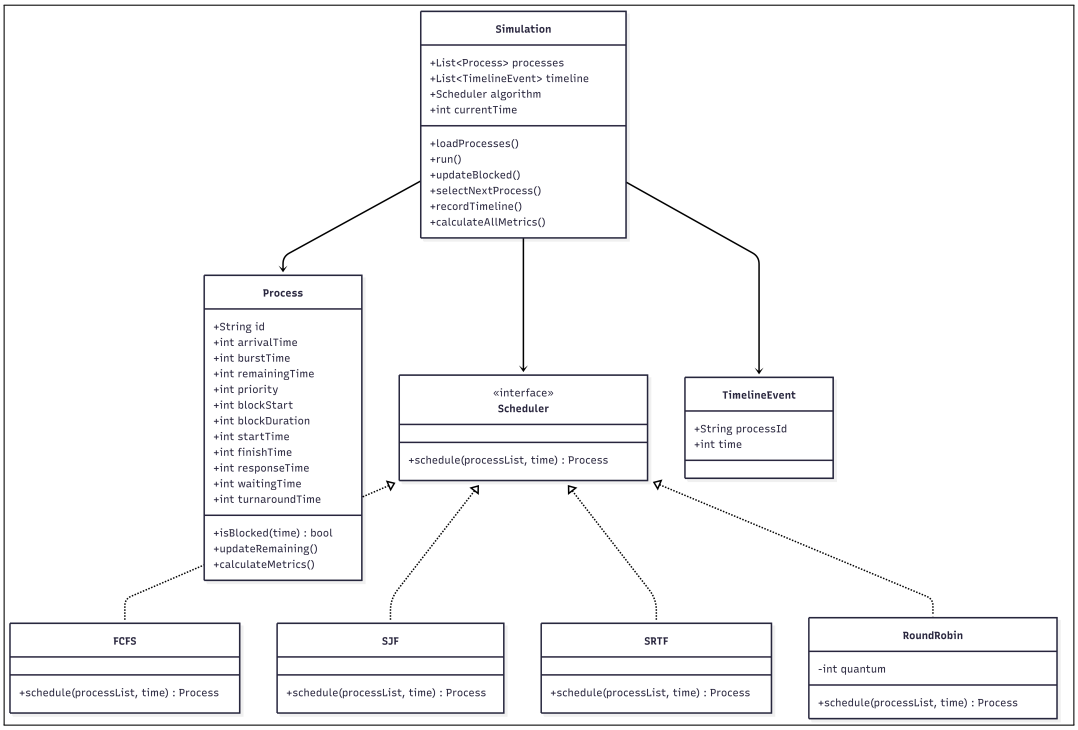


Figura 4.2: Diagrama de Clases

Diagrama de Secuencia

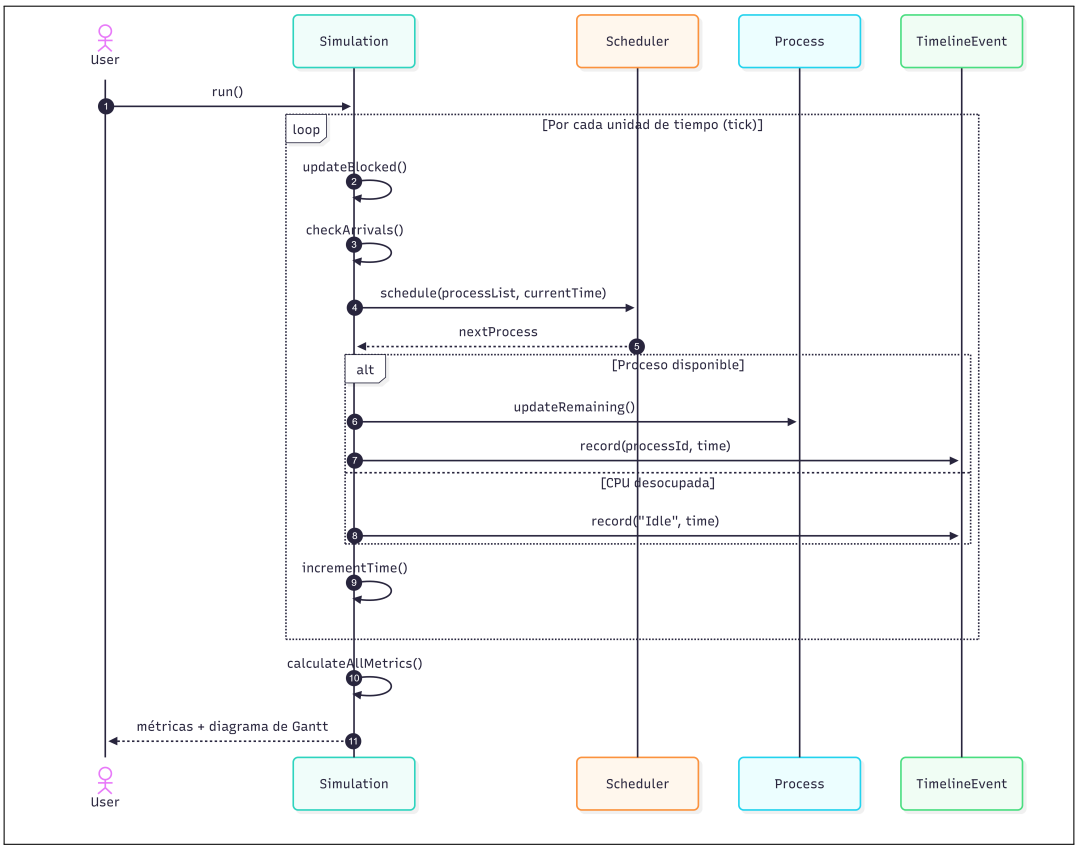


Figura 4.3: Diagrama de Secuencia del Ciclo de Simulación

4.2. Motor de Simulación

El simulador avanza tiempo en unidades discretas (ticks). En cada tick se realizan:

1. Llegada de nuevos procesos.
2. Desbloqueo de procesos finalizados en E/S.
3. Selección del proceso a ejecutar.
4. Registro del estado en la línea de tiempo.

Capítulo 5

Conclusiones

La planificación de CPU es un pilar clave en el rendimiento de los sistemas operativos. Los distintos algoritmos evaluados muestran comportamientos muy diferentes bajo las mismas condiciones de carga.
FCFS resulta simple pero ineficiente ante ráfagas largas; SJF y SRTF ofrecen tiempos óptimos bajo conocimiento adecuado; Round Robin proporciona equidad y buena interacción, aunque exige una correcta elección del quantum.
El simulador implementado permite observar estas diferencias de forma clara y educativa, siendo una herramienta útil para comprender la dinámica interna de la planificación de procesos.