Simulador de Calendarización y Sincronización

Sergio Orellana
Curso de Sistemas Operativos, UVG

31 de mayo de 2025

Resumen

Este documento describe el Proyecto 2: un simulador educativo en Python con GUI *Tkinter* para visualizar algoritmos clásicos de calendarización de procesos y mecanismos de sincronización (mutex y semáforos). Se detallan la estructura del proyecto, el funcionamiento de cada algoritmo y recomendaciones de uso.

Índice

1. Descripción del Proyecto

El objetivo es reforzar conocimientos de *scheduling*, concurrencia y sincronización. El simulador ofrece:

- Carga dinámica de archivos de texto (.txt) con la definición de procesos, recursos y acciones.
- Simulación paso a paso y visualización interactiva de:
  - Algoritmos de calendarización (diagrama de Gantt dinámico).
  - Mecanismos de sincronización (estado ACCESSED/WAITING por ciclo).
- Métricas de rendimiento: tiempo de espera promedio (WT) y turnaround (TAT).
- Interfaz intuitiva con scroll horizontal/vertical y leyenda de colores.

2. Estructura del Repositorio

```
Proyecto2-SO/
 main.py          % Punto de entrada
 gui/
    interfaz.py % Implementación de Tkinter
 scheduler/
    fifo.py      % First-In First-Out
    sjf.py       % Shortest Job First
    srt.py       % Shortest Remaining Time
    round_robin.py % Round Robin (quantum)
    priority.py  % Priority Scheduling
    utils.py     % Funciones comunes
 sync/
    mutex.py     % Simulación con mutex
    semaforo.py  % Simulación con semáforos
    sync_utils.py% Carga de recursos y acciones
 data/
     procesos_*.txt
     recursos.txt
     acciones.txt
```

3. Algoritmos de Calendarización

A continuación se presenta un resumen de cada algoritmo implementado.

3.1. First-In First-Out (FIFO)

- **Política:** Ejecuta los procesos en orden de llegada (arrival time).
- **Funcionamiento:** No hay preemptive; cada proceso corre hasta completar su CPU burst.

- **Ventaja:** Simple de implementar.
- **Desventaja:** Puede provocar *convoy effect* (procesos cortos esperan).

3.2. Shortest Job First (SJF)

- **Política:** Ejecuta primero el proceso con menor tiempo de ejecución (BT).
- **Funcionamiento:** No preemptive en esta versión.
- **Ventaja:** Minimiza tiempo de espera promedio.
- **Desventaja:** Requiere conocimiento futuro de BT; riesgo de inanición de procesos largos.

3.3. Shortest Remaining Time (SRT)

- **Política:** Versión preemptive de SJF.
- **Funcionamiento:** Si llega un proceso con menor *remaining time*, preempta al actual.
- **Ventaja:** Mejor WT promedio que SJF no preemptive.
- **Desventaja:** Mayor overhead de contexto.

3.4. Round Robin

- **Política:** Time slice (quantum) asignado de forma circular.
- **Funcionamiento:** Cada proceso obtiene hasta *quantum* ciclos; si no termina, retorna a la cola.
- **Ventaja:** Justo, adecuado para sistemas interactivos.
- **Desventaja:** Elección de quantum impacta rendimiento y overhead.

3.5. Priority scheduling

- **Política:** Procesos con prioridad más alta se ejecutan primero.
- **Funcionamiento:** Puede ser preemptive o no preemptive (según configuración).
- **Ventaja:** Permite diferenciar importancia.
- **Desventaja:** Riesgo de inanición de procesos de baja prioridad.

4. Mecanismos de sincronización

4.1. Mutex (mutual exclusion)

- **Concepto:** Bloquea un recurso compartido para un único proceso a la vez.
- **Implementación:** Cola FIFO, acceso de 1 ciclo, *auto_release* tras uso.

4.2. Semáforos

- **Concepto:** Contador que permite hasta N procesos concurrentes.
- **Operaciones:** `WAIT(P)` (decrementa, bloquea si 0), `SIGNAL(V)` (incrementa, despierta cola).
- **Ventaja:** Modela tanto exclusión mutua (N=1) como lectores simultáneos (N>1).

5. Recomendaciones y buenas prácticas

- Verificar el formato de los archivos de entrada antes de simular.
- Ajustar el *quantum* según la carga de procesos para Round Robin.
- Observar el comportamiento gráfico y comparar métricas para análisis.

6. Conclusiones

Este simulador facilita la comprensión dinámica de la planificación y sincronización en sistemas operativos, mostrando gráficamente cómo las políticas impactan los estados y tiempos de los procesos.

Referencias

[1] GeeksforGeeks. (2023, 24 marzo). *Program for Shortest Job First (or SJF) CPU Scheduling | Set 1 (Non preemptive)*. GeeksforGeeks. https://www.geeksforgeeks.org/program-for-shortest-job-first-or-sjf-cpu-scheduling-set-1-non-preemptive/

[2] GeeksforGeeks. (2024a, 21 octubre). *Mutex vs Semaphore*. GeeksforGeeks. https://www.geeksforgeeks.org/mutex-vs-semaphore/

[3] GeeksforGeeks. (2024b, 28 diciembre). *Introduction of Shortest Remaining Time First (SRTF) algorithm*. GeeksforGeeks. https://www.geeksforgeeks.org/introduction-of-shortest-remaining-time-first-srtf-algorithm/

[4] GeeksforGeeks. (2025a, 7 enero). *Semaphores in Process Synchronization*. GeeksforGeeks. https://www.geeksforgeeks.org/semaphores-in-process-synchronization/

[5] GeeksforGeeks. (2025b, 13 enero). *Program for Round Robin Scheduling for the Same Arrival Time*. GeeksforGeeks. https://www.geeksforgeeks.org/program-for-round-robin-scheduling-for-the-same-arrival-time/

[6] GeeksforGeeks. (2025c, 13 enero). *Round Robin Scheduling Algorithm with Different Arrival Time*. GeeksforGeeks. https://www.geeksforgeeks.org/round-robin-scheduling-with-different-arrival-times/

[7] GeeksforGeeks. (2025d, 3 febrero). *Shortest remaining time first (Preemptive SJF) scheduling algorithm*. GeeksforGeeks. https://www.geeksforgeeks.org/shortest-remaining-time-first-preemptive-sjf-scheduling-algorithm/

[8] GeeksforGeeks. (2025e, 22 mayo). *Priority Scheduling in Operating System*. GeeksforGeeks. https://www.geeksforgeeks.org/priority-scheduling-in-operating-system/