Simulador de Algoritmos de Planificación y Sincronización

CC3064 – Sistemas Operativos

Docente: Juan Luis García

Universidad del Valle de Guatemala

Brandon Javier Reyes Morales 22992

Link repositorio:

<https://github.com/BrandonReyes0609/proyecto2-SO.git>

**Descripción General**

Este proyecto permite simular visualmente:

Algoritmos de calendarización de procesos.

Mecanismos de sincronización por semáforos y mutex.

Los usuarios pueden cargar archivos .txt con datos de entrada y observar gráficamente el comportamiento de los procesos y recursos a lo largo del tiempo (por ciclos), incluyendo estados dinámicos (WAITING / ACCESSED) y métricas como tiempo promedio de espera y turnaround Definición de Proyecto

**¿Cómo se usa el proyecto?**

1. Requisitos Previos

* Tener instalado Python 3.x
* Instalar Streamlit con pip install streamlit
* Instalar dependencias adicionales si el proyecto las requiere (como matplotlib, pandas)

2. Ejecución del Simulador

Desde la terminal, navegar al directorio raíz del proyecto y ejecutar:

3. Interacción con la App

* Seleccionar tipo de simulación: Calendarización o Sincronización.
* Cargar archivos necesarios (dependiendo del modo):
  + procesos\_sync\_var.txt
  + recursos\_sync\_var.txt
  + acciones\_mutex\_var.txt o acciones\_semaforo\_var.txt
* Seleccionar algoritmo de planificación o modo de sincronización.
* Ejecutar simulación y observar resultados.

Archivos necesarios para cada modo

Modo Calendarización:

Archivo .txt con formato:

<PID>, <BT>, <AT>, <PRIORIDAD>

Ejemplo: P1, 8, 0, 1

Se requiere para los algoritmos:

* FIFO
* SJF
* SRT
* Round Robin (configurar quantum)
* Priority

Modo Sincronización:

* Procesos: procesos\_sync\_var.txt

PID, BT, AT, Priority

* Recursos: recursos\_sync\_var.txt

R1, 1

* Acciones:
  + acciones\_mutex\_var.txt para modo Mutex
  + acciones\_semaforo\_var.txt para modo Semáforo

PID, ACCIÓN, RECURSO, CICLO

Ejemplo: P1, READ, R1, 0

**Algoritmos de Calendarización Implementados**

| Algoritmo | Descripción | Métricas observadas |
| --- | --- | --- |
| FIFO | Orden de llegada | Tiempo promedio de espera y turnaround |
| SJF | Proceso con menor Burst Time | Tiempo de espera óptimo (no preventivo) |
| SRT | Variante preventiva de SJF | Mejora tiempo de respuesta para procesos |
| Round Robin | CPU compartida con quantum fijo | Justo para entornos interactivos |
| Priority | Basado en prioridad asignada a cada proceso | Puede causar starvation sin prioridad aging |

**Mecanismos de Sincronización**

Mutex

* Permite exclusión mutua de recursos críticos.
* Visualiza el estado del recurso por ciclo.
* Puede generar espera activa si no se libera el recurso.

Semáforo

* Usa contadores para controlar acceso múltiple o secuencial.
* Acciones como WAIT, SIGNAL, READ, WRITE se representan visualmente.

Discusión de Resultados

* Los algoritmos como SRT muestran el mejor rendimiento en cuanto a tiempo de espera y turnaround promedio en cargas mixtas.
* FIFO y Priority dan resultados similares cuando los tiempos de llegada y prioridades son homogéneos.
* Round Robin penaliza el turnaround cuando el quantum es pequeño comparado con los BT.
* En sincronización:
  + Semáforos logran mejor control de acceso escalonado.
  + Mutex es más estricto, y puede causar starvation si un proceso de alta prioridad no libera un recurso compartido.
* Las simulaciones muestran animaciones por ciclo, lo cual facilita la comprensión visual de conflictos como esperas activas o recursos compartidos.

| Archivo / Carpeta | Descripción |
| --- | --- |
| app.py | Archivo principal de la app Streamlit |
| simulador/ | Módulos de calendarización, sincronización, Gantt |
| procesos\_sync\_var.txt | Lista de procesos con AT, BT y prioridad |
| recursos\_sync\_var.txt | Recursos y su contador |
| acciones\_mutex\_var.txt | Acciones de sincronización con Mutex |
| acciones\_semaforo\_var.txt | Acciones de sincronización con Semáforo |

**Bibliografía**

* Silberschatz, A., Galvin, P. B., & Gagne, G. (2018). *Operating System Concepts*. Wiley.
* Stallings, W. (2018). *Operating Systems: Internals and Design Principles*. Pearson.
* Tanenbaum, A. S., & Bos, H. (2015). *Modern Operating Systems*. Pearson.
* García Zarceño, J. L. (2025). *Temas de Sistemas Operativos*. Universidad del Valle de Guatemala.