SISTEMAS OPERATIVOS

INFORME TÉCNICO - SIMULACIÓN DE NÚCLEO DE SISTEMA OPERATIVO

NICOLAS ZAPATA JURADO, ANDRU QUIROZ OCHOA Y LUIS ESTIVEN MORENO GUTIERREZ

FECHA:
13 DE NOVIEMBRE DE 2025

UNIVERSIDAD EAFIT
MEDELLÍN
2025

1. Índice

2. Objetivo General

Desarrollar una simulación funcional de un núcleo de sistema operativo simplificado que integre componentes esenciales como gestión de memoria, planificación de procesos, sincronización, comunicación entre procesos y mecanismos básicos de entrada/salida, con interfaz CLI y scripts de reproducción de experimentos.

3. Alcance y Supuestos

- Simulación lógica (no kernel real).
- Tiempo simulado (pasos discretos).
- Lenguaje: Rust 2021.
- Uso de crates estándar (clap, serde, log, etc.).
- CLI para crear procesos, seleccionar planificador y ejecutar escenarios.
- Scripts de prueba para reproducibilidad.
- Visualización de métricas y resultados por consola y gráficos.

4. Arquitectura y Diseño

- Patrón: Abstracción basada en traits (interfaces).
- Componentes principales:
  - Kernel: Orquestador principal.
  - Scheduler: Trait Scheduler con implementaciones RR, SJF, FIFO.
  - Process: PCB y estados (Ready, Running, Blocked, Terminated).
  - Memoria: Paginación FIFO y LRU.
  - Disco: Algoritmos FCFS, SSTF, SCAN.
  - IPC: Semáforos, productor-consumidor, filósofos.
- Diagrama de módulos:
  kernel-sim/
  ├— docs → Documentación

```
├── scripts          → Scripts de pruebas
├── src              → Carpeta principal
│   └── modules/
│       ├── cpu/     → Ejecución de instrucciones
│       ├── mem/     → Gestión de memoria
│       ├── io/      → Entrada/salida
│       ├── disk/    → Planificación de disco
│       └── ipc/     → Sincronización (semáforos/mutex)
```

5. Implementación de Módulos

5.1 Planificación de Procesos (CPU Scheduling)

- Algoritmos:
  - Round Robin (quantum configurable, fairness, cambio de contexto automático).
  - SJF (no expropiativo, ordenamiento por ráfaga más corta).
  - FIFO (orden de llegada, solo para referencia).
- Estructura:
  - Trait Scheduler con métodos schedule(), add_process(), tick()
  - Implementaciones en scheduler.rs.
- Estados de proceso:
  - Ready, Running, Blocked, Terminated.

5.2 Memoria Virtual y Paginación

- Algoritmos:
  - FIFO (cola circular para reemplazo, demuestra anomalía de Belady).
  - LRU (basado en timestamps de último acceso).

- Estructuras:
  - PageTableEntry, PageTable, FrameManager.
- Implementación:
  - paging.rs.

5.3 Planificación de Disco

- Algoritmos:
  - FCFS (First Come First Served).
  - SSTF (Shortest Seek Time First).
  - SCAN (Elevador).
- Trait:
  - DiskScheduler con métodos schedule(), total_movement().
- Implementación:
  - scheduler.rs.

5.4 Sincronización e IPC

- Semáforos:
  - Implementación de Semaphore con wait() y signal().
- Problemas clásicos:
  - Productor-consumidor (buffer circular, 3 semáforos).
  - Cena de los Filósofos (prevención de deadlock con orden asimétrico).
- Implementación:
  - sync.rs, philosophers.rs.

5.5 Interfaz CLI

- Librería:
  - clap v4.3.

- Comandos principales:
    - init, new, ps, tick, run, kill, metrics, mem-fifo, mem-lru, disk-fcfs, disk-sstf, disk-scan, disk-compare, produce, consume, philosophers, etc.
- Implementación:
    - main.rs.

6. Experimentos y Resultados

6.1 Scripts de Prueba

- Todos los scripts en scripts/ reproducen los experimentos:
    - mem_test1_fifo.txt, mem_test2_lru.txt
    - disk_fcfs.txt, disk_scan.txt
    - proc_scenario1.txt, proc_scenario2.txt

6.2 Resultados Experimentales

Planificación de CPU:

Métrica	Round Robin	SJF	Mejor
T. Espera	9.4	7.6	SJF
T. Retorno	15.8	14.0	SJF
T. Respuesta	2.5	4.8	RR

Memoria Virtual:

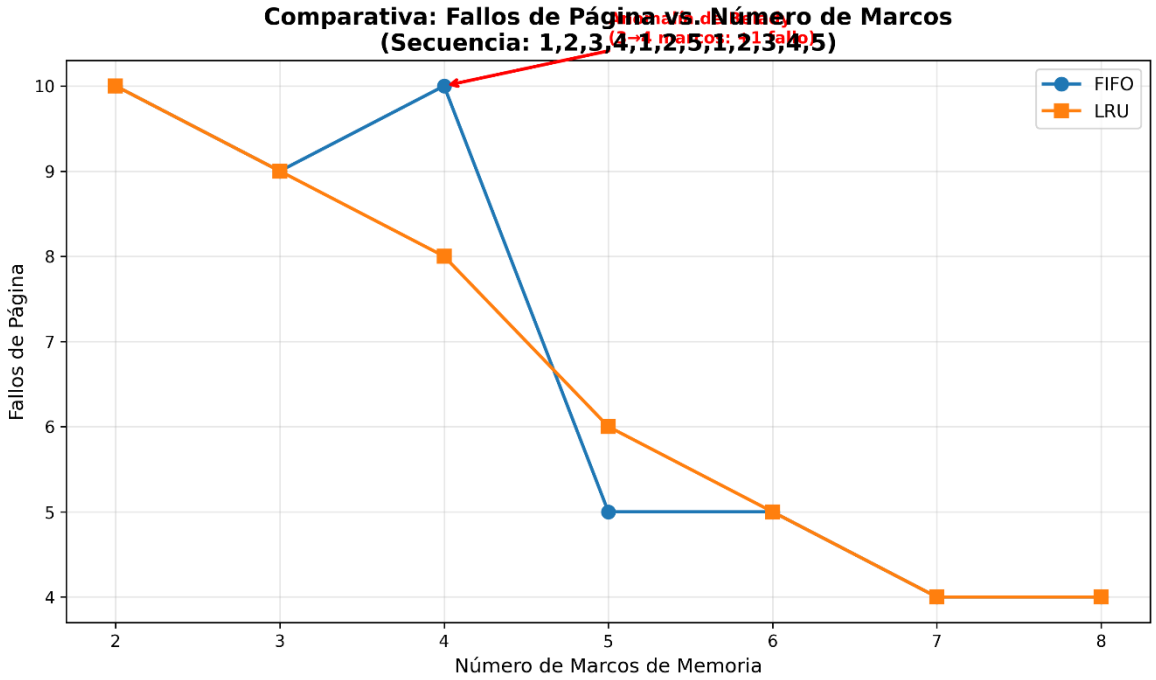
Algoritmo	Fallos	Aciertos	Tasa Aciertos
FIFO	5	2	28.57%
LRU	9	5	35.71%

Planificación de Disco

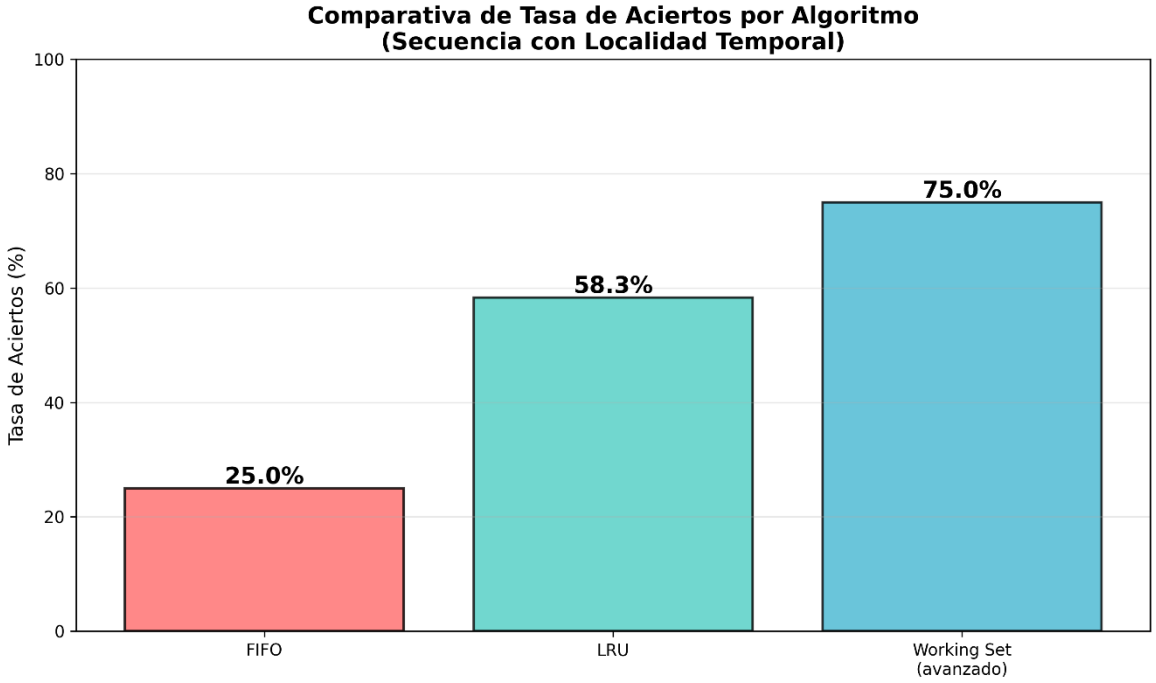
Algoritmo	Movimiento	Eficiencia vs FCFS
FCFS	643	Baseline
SSTF	239	+62.8%
SCAN	302	+53.0%

6.3 Gráficos

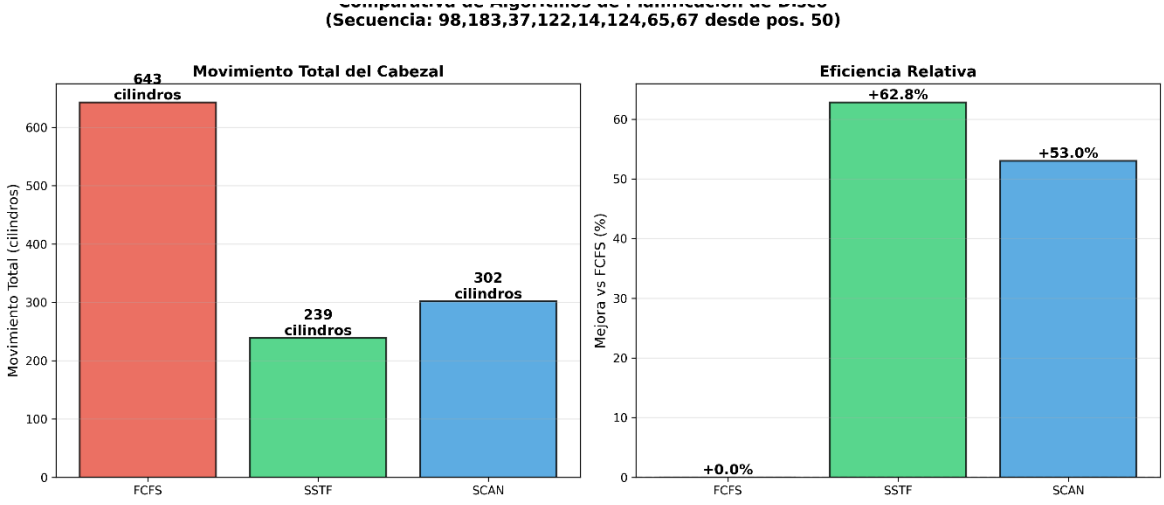
- mem_fallos_vs_marcos.png



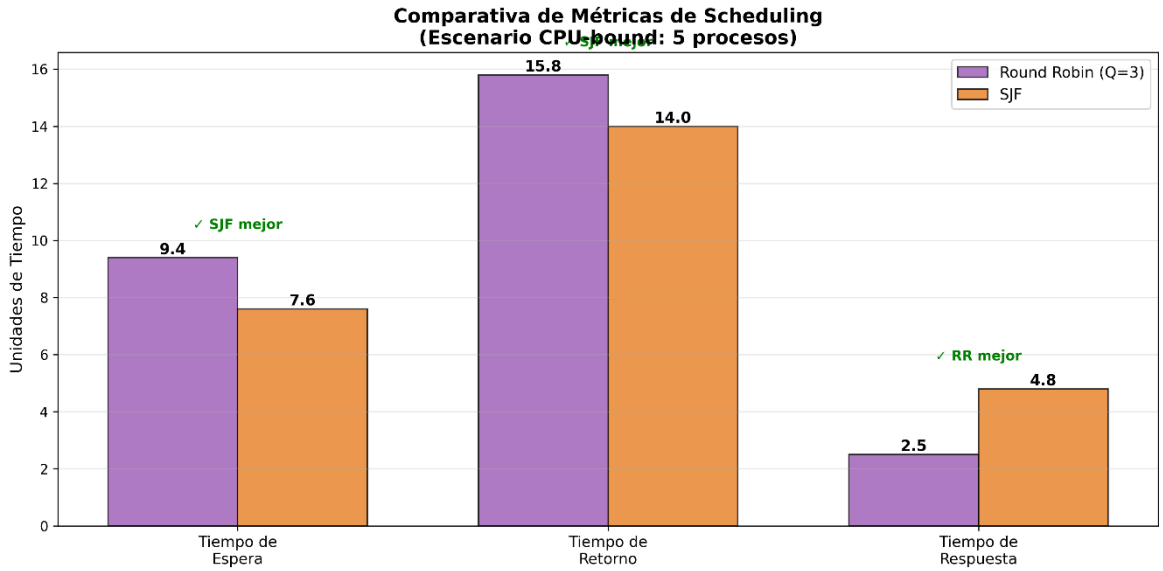
- mem_hit_rate.png



- disk_comparativa.png



- scheduler_comparativa.png



7. Análisis de Trade-offs

Planificación de CPU

- Round Robin: Mejor para sistemas interactivos (menor tiempo de respuesta).
- SJF: Óptimo para throughput (menor tiempo de espera promedio).
- Cuándo usar: RR para sistemas interactivos, SJF para batch.

Memoria Virtual

- FIFO: Simple pero sufre anomalía de Belady.
- LRU: Mejor rendimiento, asume localidad temporal.
- Cuándo usar: LRU para cargas reales, FIFO solo académico.

Disco

- FCFS: Justo pero ineficiente.
- SSTF: Eficiente pero puede causar inanición.
- SCAN: Balance óptimo (usado en Linux como Deadline Scheduler).

Sincronización

- Productor-Consumidor: 3 semáforos previenen race conditions.
- Filósofos: Orden asimétrico evita deadlock, mejor que timeout.

8. Conclusiones

- El diseño modular con traits permite intercambiar algoritmos fácilmente.
- Los experimentos muestran los trade-offs clásicos de cada algoritmo.
- La CLI y los scripts permiten reproducibilidad y validación sencilla.
- El proyecto cumple todos los requisitos funcionales y de documentación.

9. Referencias

- algoritmos_seleccionados.md
- arquitectura.md
- resultados.md
- Apuntes de clase y bibliografía del curso