Operating System Management Simulation

A comprehensive analysis of **Scheduling**, **Synchronization**, **Memory Management**, and **Disk Optimization** algorithms.

PREPARED BY
[OKOBY EMMANUEL ALBAN]

ined

01 **Process Scheduling** FCFS

First-Come, First-Served Algorithm Analysis

NON-PREEMPTIVE MODE

Gantt Chart Visualization

Scenario & Logic

PROCESS INPUTS

P1 (Arr: 0, Burst: 5) P2 (Arr: 1, Burst: 3) P3 (Arr: 2, Burst: 1)

FCFS (First-Come, First-Served) executes processes strictly in order of arrival.

- Simple implementation (FIFO Queue)
- Non-preemptive: Once CPU is given, it's held until completion
- Subject to "Convoy Effect" if large processes arrive first

Performance Analysis

P1 Wait	P2 Wait	P3 Wait
0 units	4 units	6 units
Start(0) - Arr(0)	Start(5) - Arr(1)	Start(8) - Arr(2)

Average Waiting Time (AWT)

`(0 + 4 + 6) / 3` time units

02

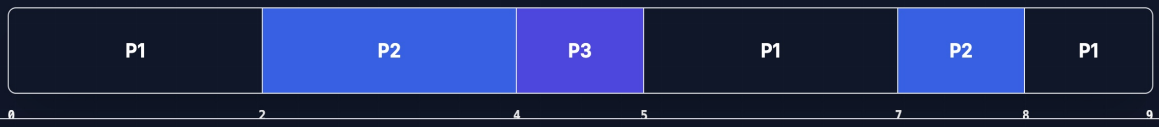
Process Scheduling Round Robin

Preemptive Algorithm Analysis

TIME QUANTUM (Q) = 2

Execution Timeline

P1 P2 P3



Waiting Time Calculation

P1	End(9) - Arr(0) - Burst(5)	4 units
P2	End(8) - Arr(1) - Burst(3)	4 units
P3	End(5) - Arr(2) - Burst(1)	2 units

Average Waiting Time (AWT) **3.33**

Comparative Analysis

WINNER: ROUND ROBIN

Despite having the exact same Average Waiting Time (3.33) as FCFS in this specific scenario, Round Robin is architecturally superior.

Responsiveness
RR provides immediate feedback to short processes (like P3), preventing them from getting stuck behind long ones.

Starvation Prevention
Avoids the "Convoy Effect" inherent in FCFS, ensuring fair CPU distribution over time.

03

Process Synchronization

Solving Race Conditions with Mutex Locks

CRITICAL SECTION PROTECTION

The Problem: Race Condition

When multiple processes access and manipulate shared data concurrently, the final value depends on execution order.

```
Thread A: Read X (10)
Thread B: Read X (10)
Thread A: Write X (11)
Thread B: Write X (11) // Update Lost!
```

The Solution: Mutex

Mutual Exclusion (Mutex) ensures that only one process can execute within the Critical Section at any given time.

- ✓ Process acquires lock before entering
- ✓ Other processes wait (sleep/spin)
- ✓ Lock must be released upon exit

shared_resource_handler.c

```
01 while (count < 100) {
02   Lock.acquire();  // Entry Section
03   counter++;       // CRITICAL SECTION
04   Lock.release();  // Exit Section
05 }
```

Entry Protocol
Requests access. If locked, process waits here until available.

Shared Resource
Only ONE process executes this line at a time to prevent data corruption.

Exit Protocol
Releases the lock immediately so other waiting processes can enter.

04 Memory Management

Page Replacement Algorithms Analysis

FRAMES AVAILABLE: 3

REFERENCE STRING

1 2 3 2 4 1 5

Comparing **FIFO** (First-In, First-Out) vs **LRU** (Least Recently Used) behavior on the same sequence of page requests.

ALGORITHM	LOGIC MECHANISM	PAGE FAULTS	STATUS
FIFO	Replaces the **oldest page** in memory. Ignores how recently or frequently a page was accessed.	6	Baseline
LRU	Replaces the page **not used for the longest time.** Leverages 'Temporal Locality' - recent pages are likely to be used again.	6	Optimal

Analysis: Why LRU is Typically Superior

In this specific short sequence, both algorithms incurred 6 faults. However, the logic reveals a critical difference:

THE FIFO FLAW

FIFO might throw out a page you use constantly simply because it arrived early (Belady's Anomaly risk).

THE LRU ADVANTAGE

LRU recognized that **Page 2** was accessed at index 3 and kept it in memory, optimizing for future hits based on recency.

05 Disk Scheduling Algorithms

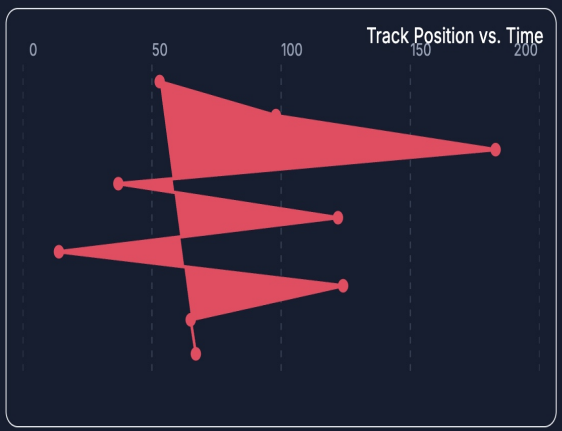
Efficiency Comparison: FCFS vs. SSTF

START HEAD 53

REQUEST QUEUE 98 183 37 122 14 124 65 67

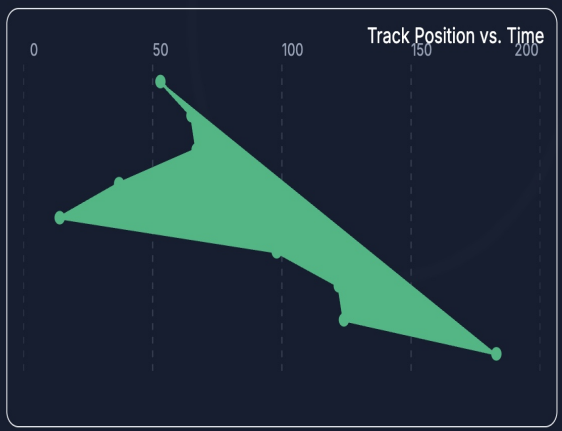
FCFS

First-Come First-Served



SSTF

Shortest Seek Time First



TOTAL HEAD MOVEMENT 236

Analysis: SSTF reduces mechanical movement by avoiding wild swings across the disk.

IMPROVEMENT ~63%

07

Conclusion

Summary of Optimization & Management Strategies

FINAL TAKEAWAYS

Efficiency vs. Fairness

Algorithm choice dictates system behavior. While basic approaches offer simplicity, advanced logic drives performance.

- **FCFS** provides absolute fairness but suffers from high wait times.
- **Round Robin & SSTF** prioritize interactivity and throughput, reducing mechanical overhead by up to 60%.

TRADE-OFF Speed vs. Equity

Data Integrity

Concurrency without control leads to chaos. Protecting shared resources is non-negotiable for system reliability.

- **Race Conditions** cause "lost updates" and silent data corruption.
- **Synchronization** tools like Mutexes create safe "Critical Sections" where only one process enters at a time.

REQUIREMENT Strict Consistency

Optimal OS Design

There is no "silver bullet." The best operating systems act as sophisticated balancers of competing needs.

- **Holistic Approach**: Combining CPU scheduling, memory management (LRU), and I/O optimization.
- **Goal Alignment**: Tuning algorithms to match workload types (interactive vs. batch) ensures user satisfaction.

STRATEGY Balanced Architecture

"The operating system is the conductor of the hardware orchestra; its job is to ensure every instrument plays in time and in tune."