Web-Based CPU Scheduling Algorithms Simulation

Operating Systems Course Project

Course

Operating Systems
First Semester 2025/2026

Instructor

Dr. Abdullah Al Zaqebah

Team Members

- Joud Kayyali (3230601030) Lead Architect
- Ibraheem Abdullah (3230605046) Systems Integrator
- Alnader Ahmad (3230601029) UI/UX Engineer

Introduction to CPU Scheduling

What is CPU Scheduling?

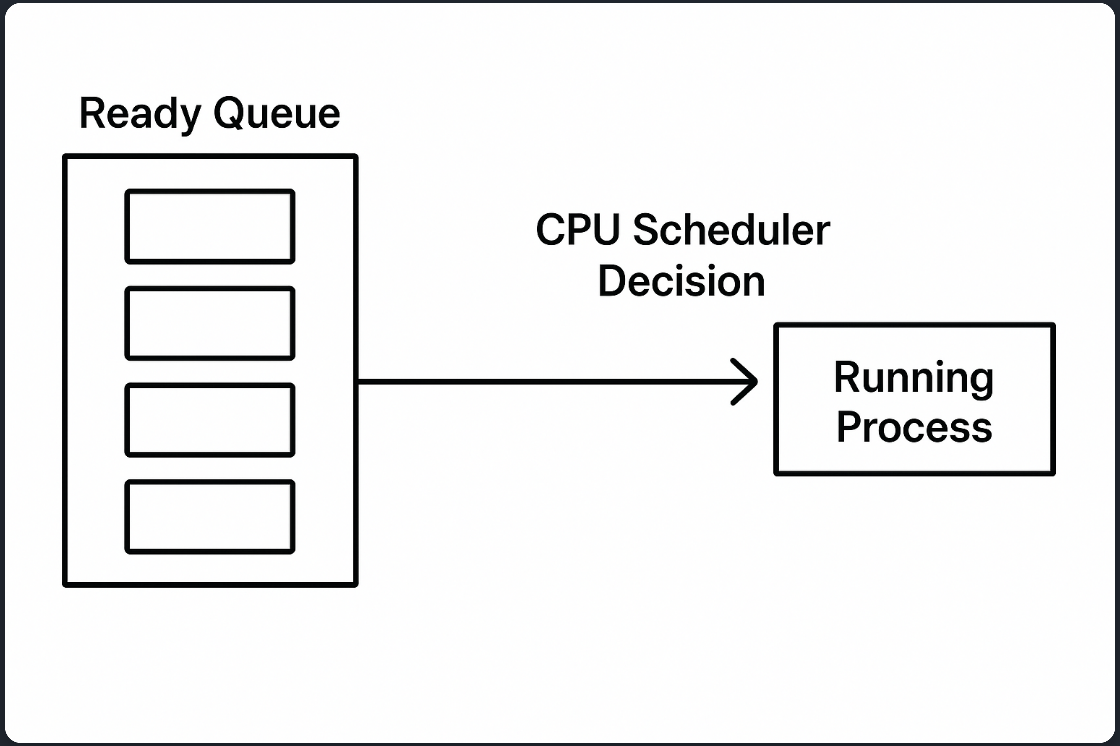
- Process of deciding which process gets **CPU time** when multiple processes compete
- Core responsibility of the **Operating System kernel**
- Balance between system efficiency and fairness

Why is it Important?

- Maximizes **CPU utilization** in multiprogramming environments
- Directly impacts system performance and user experience
- Different algorithms optimize for different metrics

Key Performance Metrics

- **Waiting Time (WT):** Time spent in ready queue
- **Turnaround Time (TAT):** Total time from submission to completion
- **Response Time (RT):** Time until first response



Problem & Solution

The Challenge

- CPU scheduling algorithms are **complex and abstract**
- Traditional learning relies on **static diagrams** and calculations
- Students struggle to visualize **dynamic process execution**
- No interactive tools to explore **algorithm trade-offs**

Our Solution

- **Web-based simulator** with interactive visualization
- **Real-time Gantt charts** showing process execution
- **Instant calculation** of performance metrics
- **Comparative analysis** of different algorithms

Project Objectives

Visualizing Execution

- Dynamic Gantt charts showing process timeline
- Real-time process state transitions
- Interactive algorithm comparison

Comparing Metrics

- Calculate WT, TAT, RT for each process
- Display average metrics for algorithm efficiency
- Side-by-side performance comparison

Mobile Accessibility

- Mobile-first design philosophy
- Responsive layout for all devices
- Touch-friendly interface controls

Educational Value

- Interactive learning tool for students
- Visualize algorithm trade-offs
- Bridge theory with practical implementation

The Algorithms

FCFS

First Come First Served

- Non-preemptive algorithm
- Processes executed in arrival order
- Simple but suffers from convoy effect

SJF

Shortest Job First

- Non-preemptive algorithm
- Selects process with shortest burst time
- Optimal for minimum waiting time

Round Robin

Time-Sharing Algorithm

- Preemptive algorithm
- Fixed time quantum for each process
- Processes rotate in circular queue

Priority

Priority-Based Scheduling

- Preemptive algorithm
- Runs highest priority process
- Preempts for higher priority arrivals

Architecture & Technologies

Modular JavaScript

- **Vanilla ES6+** with no external libraries
- Separate modules for each algorithm
- Clear separation between **logic** and **UI**

Mobile-First Design

- UI designed for **mobile first**
- Touch-friendly interface elements
- Progressive enhancement for **desktop**

CSS Architecture

- **Flexbox/Grid** for responsive layouts
- Dark theme with **#00ADB5** accent color
- CSS-only scrolling for Gantt charts

File Structure

- **index.html** - Presentation Layer
- **style.css** - Styling Layer
- **[Algorithm].js** - Algorithm Layer
- **script.js** - Controller Layer

Development Phases

Research & Proposal

1

Weeks 6-7

- Defined project scope and objectives
- Selected 4 algorithms for implementation
- Designed PCB data structure
- Planned input/output formats

Key Deliverables

- Project proposal (1-2 pages)
- Technology stack selection

Design & Logic

2

Weeks 8-9

- Implemented FCFS algorithm
- Created modular architecture
- Developed metric calculations
- Built core UI structure

Key Deliverables

- Progress report (2-3 pages)
- FCFS implementation

UI & Testing

3

Weeks 10-12

- Completed all algorithms
- Implemented Gantt charts
- Added responsive design
- Created test cases and analysis

Key Deliverables

- Final report (5+ pages)
- Live demo & source code

Live Demo

Live Demo

https://joudn2001.github.io/OS-Scheduler-Simulation-Project

Interactive web-based CPU scheduling simulator

Source Code

https://github.com/JoudN2001/OS-Scheduler-Simulation-Project

Complete project repository with documentation

Scan to Access Live Demo

Key Features

- **4 algorithms** with visual Gantt charts
- **Real-time metrics** calculation (WT, TAT, RT)
- **Mobile-first** responsive design

Test Cases & Results

Test Scenarios

1 Same Arrival Time

Process	AT	BT	Priority
P1	0	4	2
P2	0	3	1
P3	0	2	3
P4	0	1	2

2 Different Arrival Time

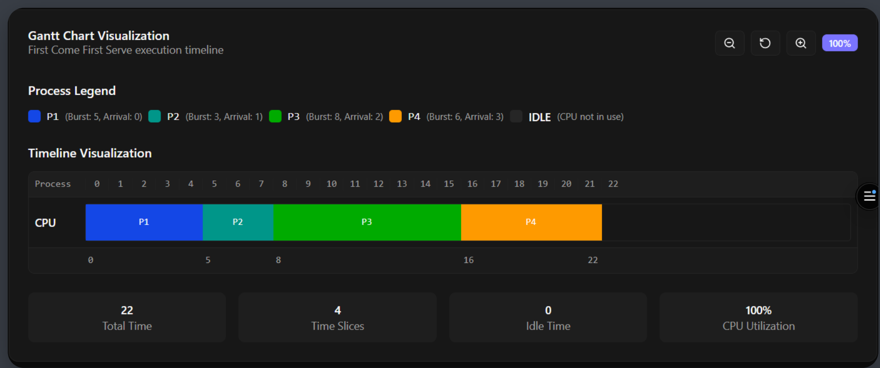
Process	AT	BT	Priority
P1	0	4	2
P2	3	3	1
P3	5	2	3
P4	9	1	2

Performance Comparison

Algorithm	Avg WT	Avg TAT	Best For
FCFS	4.5 / 3.5	7.5 / 6.5	Simplicity
SJF	**2.75 / 2.25**	**5.75 / 5.25**	**Both Scenarios**
RR (Q=2)	3.25 / 2.75	6.25 / 5.75	Fairness
Priority	3.0 / 2.5	6.0 / 5.5	Urgent Tasks

Key Findings

- **SJF** consistently outperformed others in both scenarios
- **FCFS** showed 28% higher waiting time than SJF
- **Different arrival times** improved all algorithms by ~20%
- **RR** provided best balance between fairness and efficiency



Conclusion & Future Work

Project Achievements

- Successfully implemented **4 algorithms** with accurate metric calculations
- Created **responsive, mobile-first** web interface
- Developed **modular architecture** without external dependencies
- Validated algorithm behavior through **comprehensive testing**

Lessons Learned

- **Visualization** significantly enhances understanding of abstract concepts
- **Mobile-first** design requires careful planning from the start
- **Modular code** simplifies implementation of complex algorithms

Potential Enhancements

- Implement **additional algorithms** (MLFQ, Lottery, Guaranteed)
- Add **process animation** to visualize state transitions
- Include **multi-core simulation** with CPU affinity
- Develop **export functionality** for results and charts
- Create **interactive tutorials** for each algorithm

> *"This project successfully bridges the gap between theoretical OS concepts and practical implementation, providing students with an intuitive tool to explore CPU scheduling algorithms."*