Comparison and Simulation of Process Scheduling Algorithms: Round Robin, Shortest Job First, and Priority Scheduling

Zhuoying Cai(2104069), Na Fang(2089103), Qingyun Zhang(2095754)

Abstract

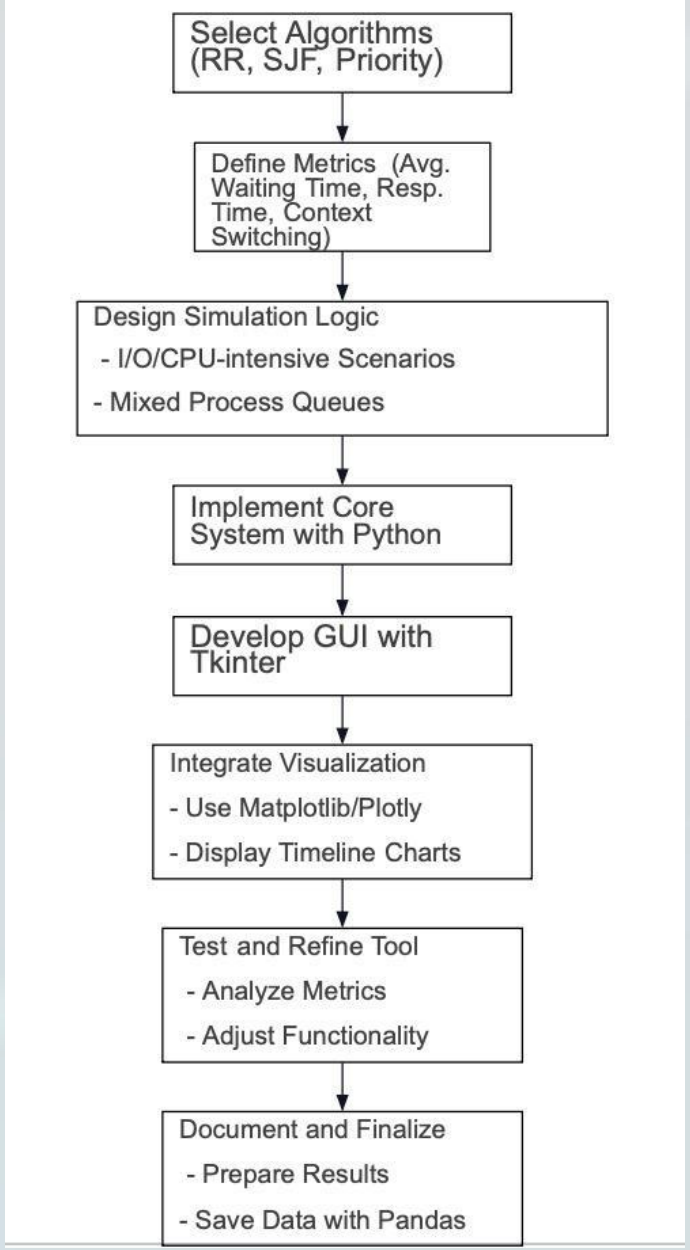
Through simulation and comparison under varying workloads, this project explores the performance of three classic process scheduling algorithms—Round Robin (RR), Shortest Job First (SJF), and Priority Scheduling.

We will design a simple process scheduling simulation system to evaluate each algorithm across multiple scenarios, including CPU-intensive, I/O-intensive, and mixed-process queues. We will comprehensively assess each algorithm's strengths and weaknesses regarding scheduling fairness, efficiency, and resource utilization by analyzing key scheduling metrics such as average waiting time, response time, and context switch frequency.

Additionally, we will identify optimal use cases for each approach. The final results, presented through detailed experimental data comparisons, will enhance understanding of the applicability and performance differences among these scheduling strategies in real-world applications, ultimately guiding practitioners in selecting the most suitable algorithm based on specific system requirements.

Requirement Definition

1. **Select Algorithms (RR, SJF, Priority)**
   - Start by selecting the type of scheduling algorithm the user wants to simulate. Options include Round Robin (RR), Shortest Job First (SJF), and Priority Scheduling.
2. **Define Metrics (Avg. Waiting Time, Resp. Time, Context Switching)**
   - Determine the metrics that will be used to evaluate the performance of the scheduling. Metrics include average waiting time, response time, and the number of context switches.
3. **Design Simulation Logic**
   - Develop the core simulation logic that considers I/O and CPU-intensive scenarios. Create different types of process queues for mixed workloads.
4. **Implement Core System with Python**
   - Implement the core logic using Python, leveraging its readability and simplicity to accurately simulate the chosen scheduling algorithms.
5. **Develop GUI with Tkinter**
   - Build a graphical user interface (GUI) using Tkinter to allow users to interact with the scheduling tool, input process data, and observe results.
6. **Integrate Visualization**
   - Integrate visualization capabilities using Matplotlib or Plotly to display timeline charts of process execution.
7. **Test and Refine Tool**
   - Test the simulation tool by analyzing performance metrics. Adjust and fine-tune functionality as necessary to ensure the tool is accurate and efficient.
8. **Document and Finalize**
   - Prepare and document results for the user, and save scheduling data with Pandas for further analysis.



Three Scheduling Algorithm Studies:

Round Robin:

- Uses time slices for fair execution
- Maintains a ready queue for waiting processes
- Rotates through processes, giving each a fixed time quantum
- Preempts process if time slice expires
- Tracks context switches when changing processes

Shortest Job First (Preemptive):

- Selects process with shortest remaining time
- Updates queue at each time unit
- Preempts current process if shorter job arrives
- Processes run until completion or preemption
- Sorts by remaining execution time

Priority Scheduling:

- Selects process with highest priority (lowest number)
- Non-preemptive implementation
- Processes run to completion once started
- Sorts by arrival time and priority
- Executes highest priority available process

Implementation

Development Language: Python 3.9

GUI Development with Tkinter

- Input Process Data
- Select Scheduling Algorithm
- Execute Scheduling
- Visualize Scheduling

Visualization of Scheduling Process

- Timeline and Process Blocks
- Conditional Judgment for Time Slice

Dynamic Process Management

- Adding Processes
- Removing Processes

Evaluation Metrics and Analysis - Key Performance Indicators

- Average Waiting Time
- Standard deviation of Waiting Time
- CPU Utilization
- Average Response Time
- Average Turnaround Time
- Throughput

Interface Output Limitations and Export to Excel Functionality

CPU Intensive

Process ID	Process Type	Arrival Time (ms)	Burst Time (ms)	Priority
P1	CPU-intensive	0	7	3
P2	CPU-intensive	1	6	2
P3	CPU-intensive	3	11	4
P4	CPU-intensive	3	8	4
P5	CPU-intensive	7	9	1
P6	CPU-intensive	8	8	5

Round Robin

Average Waiting Time: 29.50

Waiting Time Standard Deviation: 4.27

CPU Utilization: 100.00%

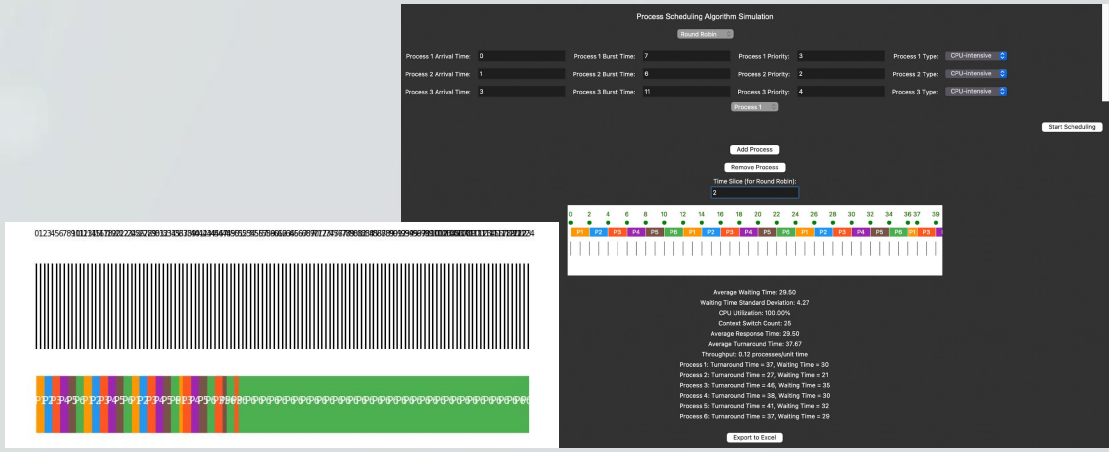
Context Switch Count: 25

Average Response Time: 1.13

Average Turnaround Time: 37.67

Throughput: 0.05 processes/unit time

PID	Arrival Time	Burst Time	Start Time	End Time	Waiting Time	Turnaround Time
1	0	7	0	37	30	37
2	1	6	2	28	21	27
3	3	11	4	51	37	48
4	3	8	6	41	30	38
5	7	9	8	48	32	41
6	8	8	10	124	33	116



Shortest Job First

Average Waiting Time: 15.00

Waiting Time Standard Deviation: 11.55

CPU Utilization: 100.00%

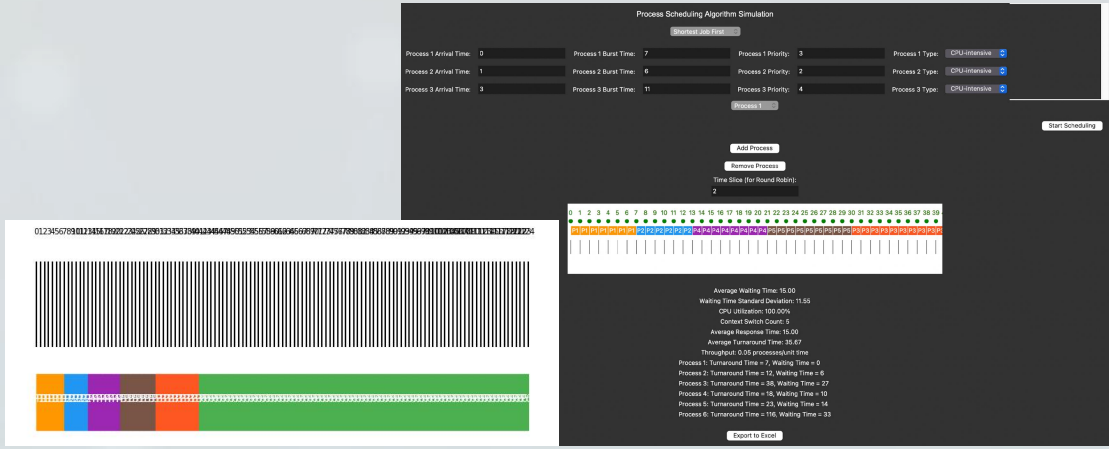
Context Switch Count: 5

Average Response Time: 15.00

Average Turnaround Time: 35.67

Throughput: 0.05 processes/unit time

PID	Arrival Time	Burst Time	Start Time	End Time	Waiting Time	Turnaround Time
1	0	7	0	7	0	7
2	1	6	7	13	6	12
4	3	11	13	21	10	18
5	3	8	21	30	14	23
3	7	9	30	41	27	38
6	8	8	41	124	33	116



Priority Scheduling

Average waiting Time: 16.17

Waiting Time Standard Deviation; 12.95

CPU Utilization: 100.00%

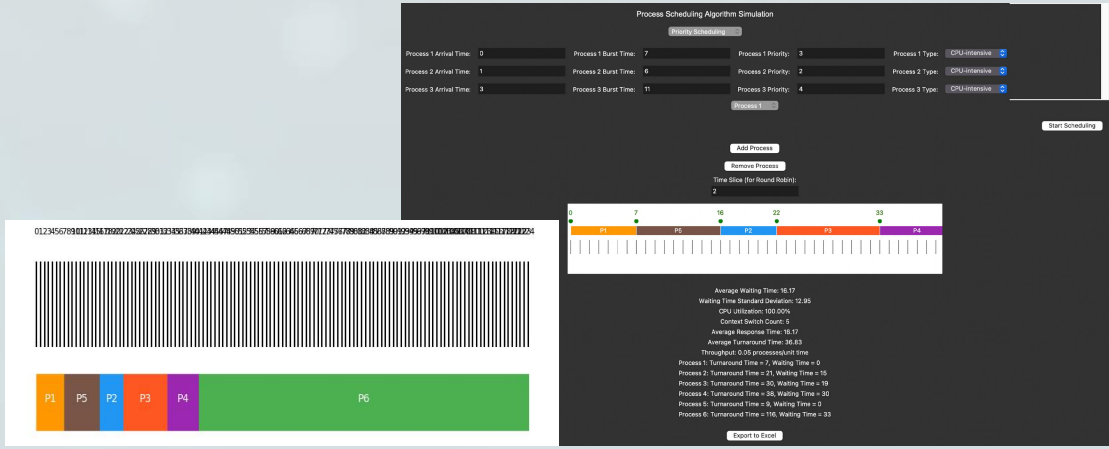
Context Switch Count: 5

Average Response Time: 16.17

Average Turnaround Time: 36.83

Throughput: 0.05 processes/unit time

PID	Arrival Time	Burst Time	Priority	Start Time	End Time	Waiting Time	Turnaround Time
1	0	7	3	0	7	0	7
5	1	6	2	7	16	0	9
2	3	11	4	16	22	15	21
3	3	8	4	22	33	19	30
4	7	9	1	33	41	30	38
6	8	8	5	41	124	33	116



I/O Intensive

Process ID	Process Type	Arrival Time (ms)	Burst Time (ms)	Priority
P1	I/O-intensive	0	2	4
P2	I/O-intensive	3	1	5
P3	I/O-intensive	6	2	5
P4	I/O-intensive	6	3	4
P5	I/O-intensive	9	1	1
P6	I/O-intensive	9	1	2

Round Robin

Average Waiting Time: 1.17

Waiting Time Standard Deviation: 1.21

CPU Utilization: 76.92%

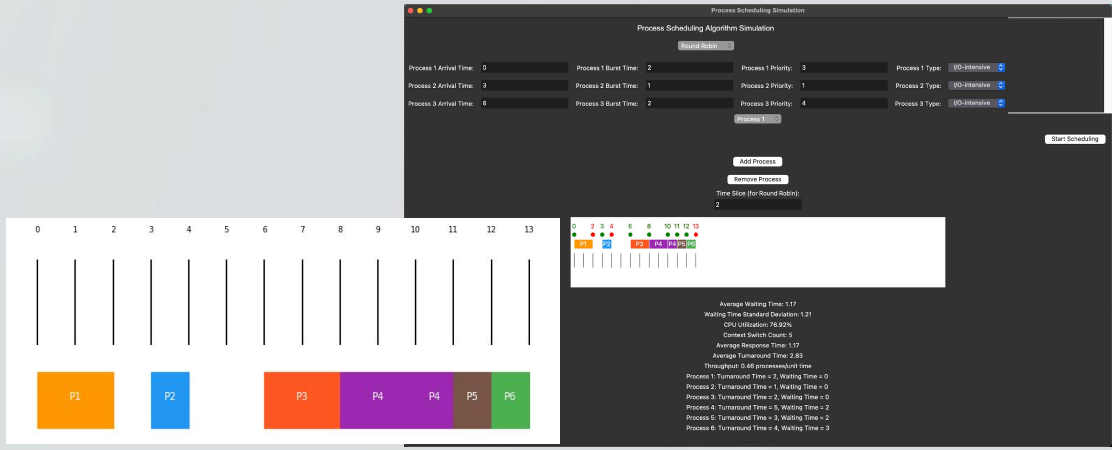
Context Switch Count: 5

Average Response Time: 1.17

Average Turnaround Time: 2.83

Throughput: 0.46 processes/unit time

PID	Arrival Time	Burst Time	Start Time	End Time	Waiting Time	Turnaround Time
1	0	2	0	2	0	2
2	3	1	3	4	0	1
3	6	2	6	8	0	2
4	6	3	8	11	2	5
5	9	1	11	12	2	3
6	9	1	12	13	3	4



Shortest Job First

Average Waiting Time: 0.83

Waiting Time Standard Deviation: 1.46

CPU Utilization: 76.92%

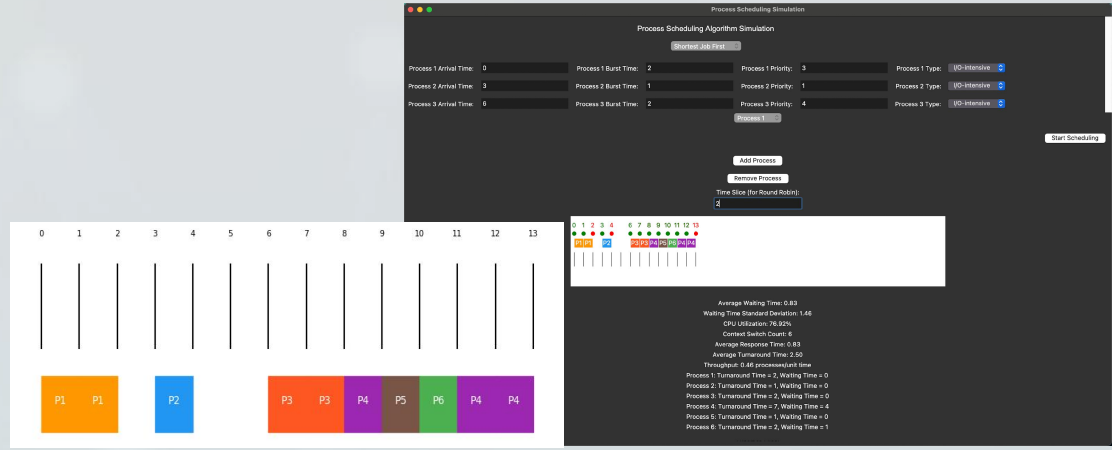
Context Switch Count: 6

Average Response Time: 0.5

Average Turnaround Time: 2.50

Throughput: 0.46 processes/unit time

PID	Arrival Time	Burst Time	Start Time	End Time	Waiting Time	Turnaround Time
1	0	2	0	2	0	2
2	3	1	3	4	0	1
3	6	2	6	8	0	2
4	6	3	8	13	4	7
5	9	1	9	10	0	1
6	9	1	10	11	1	2



Priority Scheduling

Average waiting Time: 1.00

Waiting Time Standard Deviation: 1.83

CPU Utilization: 76.92%

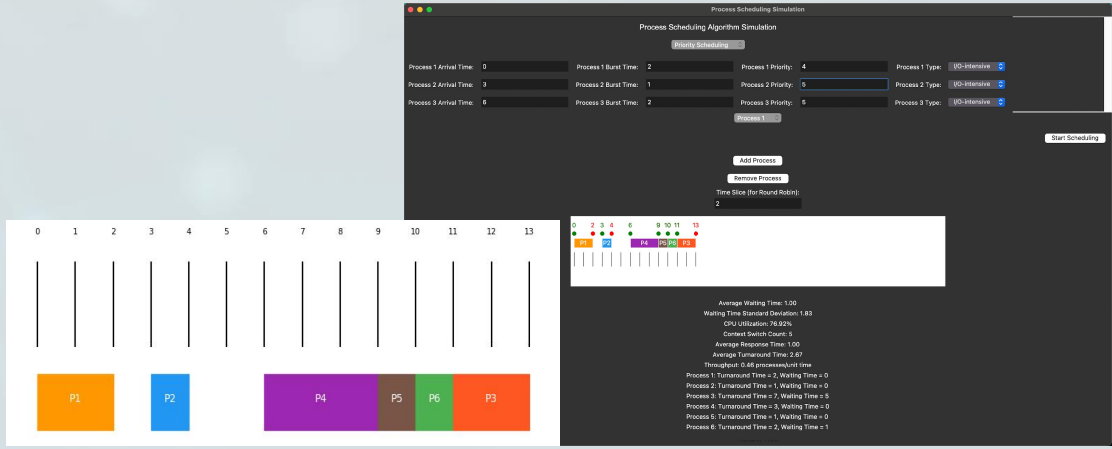
Context Switch Count: 5

Average Response Time: 1.00

Average Turnaround Time: 2.67

Throughput: 0.46 processes/unit time

PID	Arrival Time	Burst Time	Priority	Start Time	End Time	Waiting Time	Turnaround Time
1	0	2	4	0	2	0	2
2	3	1	5	3	4	0	1
4	6	2	5	6	9	0	3
5	6	3	4	9	10	0	1
6	9	1	1	10	11	1	2
3	9	1	2	11	13	5	7



Mixed Processes

Process ID	Process Type	Arrival Time (ms)	Burst Time (ms)	Priority
P1	CPU-intensive	0	8	2
P2	I/O-intensive	4	2	1
P3	CPU-intensive	4	8	5
P4	I/O-intensive	6	1	3
P5	CPU-intensive	6	9	5
P6	I/O-intensive	9	2	1

Round Robin

Average Waiting Time: 8.33

Waiting Time Standard Deviation: 5.06

CPU Utilization: 100.00%

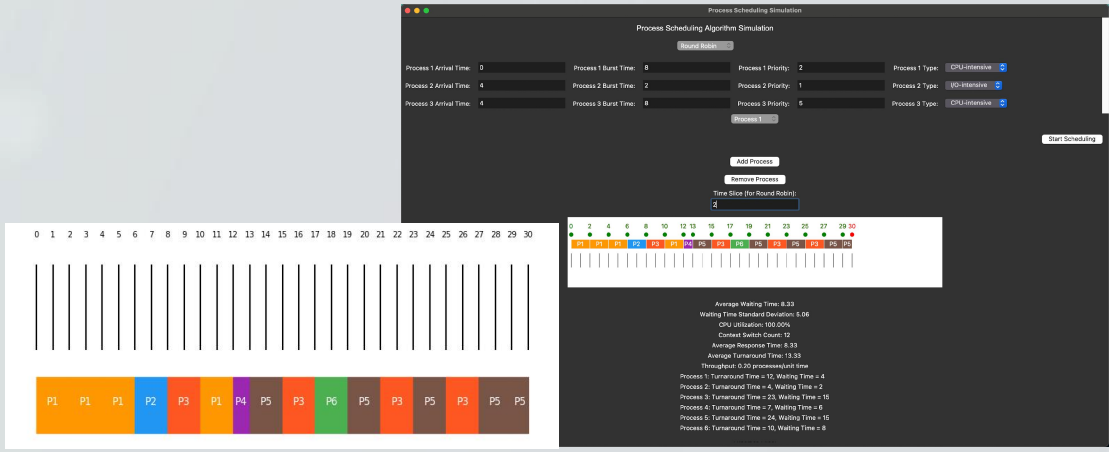
Context Switch Count: 12

Average Response Time: 4.5

Average Turnaround Time: 13.33

Throughput: 0.20 processes/unit time

PID	Arrival Time	Burst Time	Start Time	End Time	Waiting Time	Turnaround Time
1	0	8	0	12	4	12
2	4	2	6	8	2	4
3	4	8	8	27	15	23
4	6	1	12	13	6	7
5	6	9	13	30	15	24
6	9	2	17	19	8	10



Shortest Job First

Average Waiting Time: 4.83

Waiting Time Standard Deviation: 5.46

CPU Utilization: 100.00%

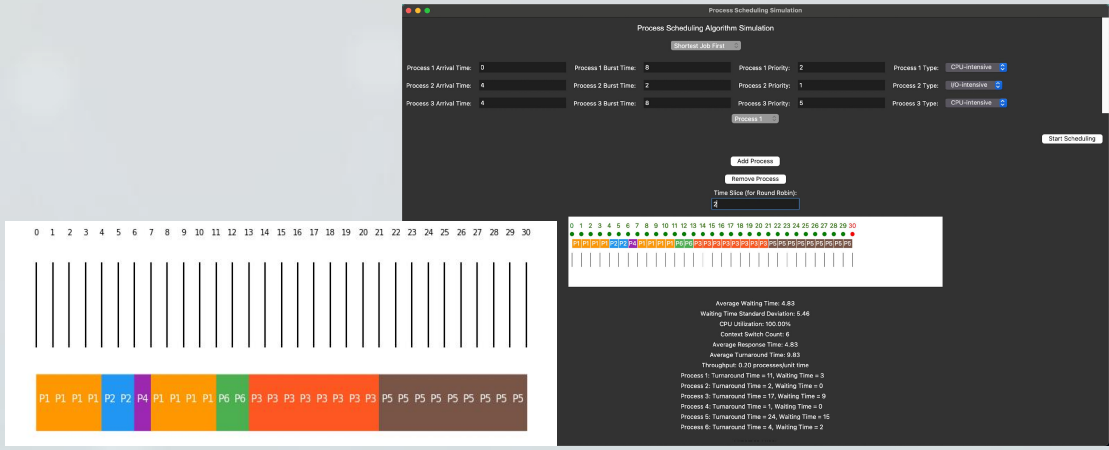
Context Switch Count: 6

Average Response Time: 4.33

Average Turnaround Time: 9.83

Throughput: 0.20 processes/unit time

PID	Arrival Time	Burst Time	Start Time	End Time	Waiting Time	Turnaround Time
1	0	8	0	11	3	11
2	4	2	4	6	0	2
4	4	8	6	7	0	1
6	6	1	11	13	2	4
3	6	9	13	21	9	17
5	9	2	21	30	15	24



Priority Scheduling

Average waiting Time: 5.83

Waiting Time Standard Deviation: 5.08

CPU Utilization: 100.00%

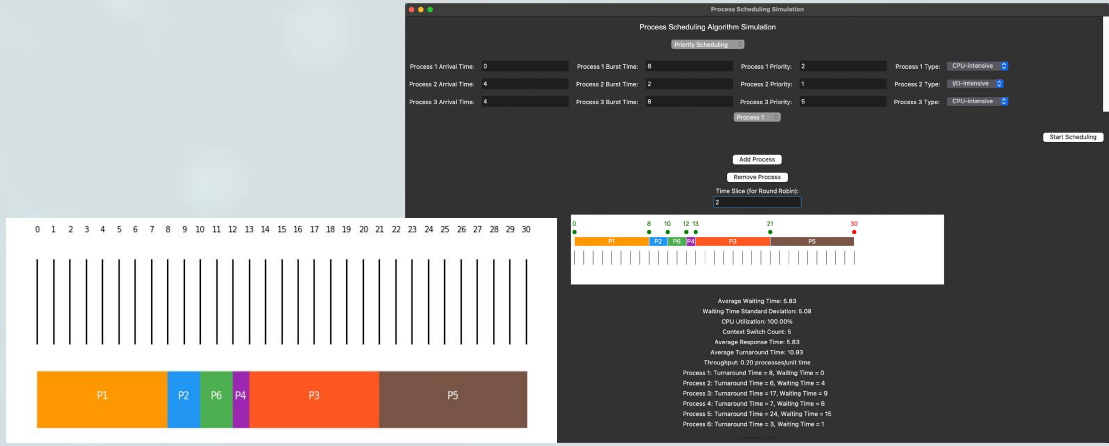
Context Switch Count: 5

Average Response Time: 5.83

Average Turnaround Time: 10.83

Throughput: 0.20 processes/unit time

PID	Arrival Time	Burst Time	Priority	Start Time	End Time	Waiting Time	Turnaround Time
1	0	8	2	0	8	0	8
2	4	2	1	8	10	4	6
6	4	8	5	10	12	1	3
4	6	1	3	12	13	6	7
3	6	9	5	13	21	9	17
5	9	2	1	21	30	15	24



Experimental Results Table

	Algorithm	Average Waiting Time	Waiting Time Standard Deviation	CPU Utilization	Context Switch Count	Average Response Time	Average Turnaround Time	Throughput (processes/unit time)
CPU Intensive	Round Robin	29.5	4.27	100.00%	25	1.13	37.67	0.05
	Shortest Job First	15	11.55	100.00%	5	15	35.67	0.05
	Priority Scheduling	16.17	12.95	100.00%	5	16.17	36.83	0.05
I/O Intensive	Round Robin	1.17	1.21	76.92%	5	1.17	2.83	0.46
	Shortest Job First	0.83	1.46	76.92%	6	0.5	2.5	0.46
	Priority Scheduling	1	1.83	76.92%	5	1	2.67	0.46
Mixed Processes	Round Robin	8.33	5.06	100.00%	12	4.5	13.33	0.2
	Shortest Job First	4.83	5.46	100.00%	6	4.33	9.83	0.2
	Priority Scheduling	5.83	5.08	100.00%	5	5.83	10.83	0.2

Findings

Performance characteristics under different process load types:

CPU Intensive Scenarios:

- Context Switching: RR switch count (25) significantly higher than other algorithms (5)
- Waiting Time: SJF (15) and Priority (16.17) perform notably better than RR (29.5)
- CPU utilization reaches 100% in all cases, indicating full CPU usage
- Low throughput (0.05), which is typical for CPU-intensive tasks

I/O Intensive Scenarios:

- CPU utilization drops to 76.92%, indicating substantial I/O waiting
- All time metrics show significant reduction (waiting time, response time, turnaround time)
- Throughput increases to 0.46, indicating more frequent process switching
- Performance differences between algorithms are less pronounced

Mixed Process Scenarios:

- Performance metrics fall between the previous two scenarios
- SJF still performs best in waiting time and turnaround time
- RR context switch count (12) remains higher than other algorithms

Findings

Algorithm Characteristics Summary:

Round Robin:
- Suitable for scenarios requiring high interactivity
- Good response time but high context switching overhead
- Higher waiting times in CPU-intensive tasks

Shortest Job First:
- Best performance in waiting time and turnaround time across all scenarios
- Moderate context switching count
- Most balanced overall performance

Priority Scheduling:
- Performance typically falls between RR and SJF
- Relatively stable performance across different scenarios
- Low context switching overhead

Practical Application Recommendations:

- Consider RR for systems requiring high interactivity
- SJF is preferred when overall performance efficiency is crucial
- Priority Scheduling is suitable when task priority is important while maintaining good overall performance
- In mixed workload situations, SJF might be the best balanced choice

Comprehensive Evaluation Across Fairness, Efficiency, and Resource Utilization

Aspects	Metrics	Round Robin	Shortest Job First	Priority Scheduling
Fairness	Process Starvation	Low Risk	High Risk	Medium Risk
	Time Distribution	Equal	Favors Short Jobs	Priority Based
	Waiting Time Variance	Low (1.21-5.06)	High (1.46-11.55)	Medium (1.83-12.95)
	Overall Fairness Rating	High	Low	Medium
Efficiency	Context Switching	High (5-25)	Low (5-6)	Low (5)
	Response Time	Good (1.13-4.5)	Variable (0.5-15.0)	Variable (1.0-16.17)
	Average Waiting Time	High in CPU-intensive	Optimal	Moderate
	Turnaround Time	Variable (2.83-37.67)	Best (2.5-35.67)	Good (2.67-36.83)
	Overall Efficiency Rating	Medium	High	Medium-High
Resource Utilization	CPU Utilization	76.92-100%	76.92-100%	76.92-100%
	System Overhead	High	Low	Low
	Resource Balance	Fair	Excellent	Good
	Overall Resource Rating	Medium	High	High
Best Suited For	Use Cases	Time-sharing systems, Interactive processes	Batch processing, Known job lengths	Systems with priority requirements
Limitations	Main Drawbacks	High context switching overhead	Potential starvation of long processes	Complex priority management
Overall Rating	System Evaluation	Good for fairness, Medium for performance	Excellent for performance, Poor for fairness	Good balance of all aspects

Summary

Features	Round Robin	Shortest Job First	Priority Scheduling
Best Scenario	Interactive Systems	Batch Systems	Real-time Systems
Core Advantages	Fair, Quick Response	Short Waiting Time	Flexible Control
Main Limitation	High Switch Overhead	Long Job Starvation	Complex Management
Typical Usage	Multi-user Systems	Data Processing	Industrial Control

Challenge

1.I/O time calculation

- **Context Switching and Interrupt Handling**：Frequent I/O operations lead to constant context switching, where the CPU suspends one process to handle another, adding computational overhead. This frequent switching and state management complicate scheduling, as the system must track and manage each process's status.
- **Scheduling Decision Complexity:**The dynamic changes in the ready queue due to I/O completions require the scheduler to continually re-evaluate process order. Balancing I/O-bound and CPU-bound tasks adds complexity, as schedulers strive to prevent device idle time while optimizing CPU usage.
- **Time Management and Event-Driven Simulation:**Accurate timing of I/O events in simulators requires handling various event types precisely and in correct sequence. An event-driven model that tracks each operation’s timing adds computational load, as simulators must manage a detailed event timeline for all processes[2].

Challenge

2.Too many process inputs

- **Impact of increased calculation amount on performance:** When too many processes are input, scheduling algorithms become more complex, causing a noticeable drop in tool responsiveness and increased memory consumption. Frequent context switches also add system overhead, further degrading performance, especially when hardware resources are limited.
- **May lead to inaccurate calculation results:** With a high volume of processes, floating-point precision limitations and algorithmic constraints can cause calculation errors and issues like starvation in lower-priority processes. Additionally, resource competition, deadlocks, and GUI response lag can disrupt accurate simulation results and degrade user experience.

Discuss & Future Study

- Future research should aim to optimize simulation models for more efficient handling of I/O operations, exploring ways to reduce context switch overhead and balance I/O- and CPU-bound processes[2].
- Setting limits on the number of processes, using multi-threading, and separating computation from the GUI can enhance both performance and user experience.
- Efficient data structures and caching can help reduce calculation time and memory usage, while improving timeline visualization can aid users in interpreting results with high process counts[3].
- These approaches could collectively enhance stability, accuracy, and usability in high-demand simulation scenarios.

Conclusion

- The scheduling simulator provides a visual and interactive approach to understanding CPU scheduling algorithms. By incorporating fairness, efficiency, and resource utilization metrics, the tool facilitates a comparative analysis of scheduling algorithms[1], enabling users to evaluate performance under various scenarios.
- Round Robin is particularly suitable for systems that require high interactivity
- Shortest Job First (SJF) is recommended for scenarios where efficiency in overall performance metrics
- Priority Scheduling is well-suited for systems where task prioritization is essential.
- Future improvements include adding more scheduling algorithms, such as Multilevel Queue and Multilevel Feedback Queue, and expanding resource utilization analysis to account for I/O operations.

Reference List

1. A. A. Alsulami, Q. A. Al-Haija, M. I. Thanoon, and Q. Mao, "Performance Evaluation of Dynamic Round Robin Algorithms for CPU Scheduling," *2019 SoutheastCon*, Huntsville, AL, USA, 2019, pp. 1-5, doi: 10.1109/SoutheastCon42311.2019.9020439.

2. O. Hajjar, E. Mekhallalati, N. Annwty, F. Alghayadh, I. Keshta, and M. Algabri, "Performance Assessment of CPU Scheduling Algorithms: A Scenario-Based Approach with FCFS, RR, and SJF," *Journal of Computer Science*, vol. 20, no. 9, pp. 972-985, 2024, doi: 10.3844/jcssp.2024.972.985.

3. L. Kishor and D. Goyal, "Comparative Analysis of Various Scheduling Algorithms," *International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)*, vol. 2, no. 4, pp. 1488, 2013.