CPU Scheduler

CS630 Operating Systems Design

FALL 2024

Under the Supervision of

Professor Larry Lay

New Jersey Institute of Technology

Project Reflection and Team Contributions

Team Members

- **Mohammed Aslam**
  Email: mm2954@njit.edu
- **Kyle Kissinger**
  Email: kck6@njit.edu
- **Ahmeduddin Mohammed**
  Email: am3837@njit.edu

---

1. Mohammed Aslam

Contribution to the Project

My primary responsibility was developing the **Graphical User Interface (GUI)** for the CPU Scheduling Simulator, assisting with backend logic, and conducting **black-box testing** for overall functionality. My key contributions included:

1. **GUI Development**:
    - Designed and implemented the GUI using **Tkinter**, focusing on a user-friendly layout.
    - Created input fields for process parameters (Process Name, Burst Time, Arrival Time, and Priority).
    - Designed dropdown menus for selecting scheduling algorithms and implemented an input field for the Round Robin time quantum.
    - Integrated buttons for **Run Simulation**, **Reset**, and **Exit**, linking them to their respective functionalities.
2. **Backend Assistance**:
    - Collaborated with Kyle on the **First-Come-First-Serve (FCFS)** algorithm to ensure proper integration with the GUI.
    - Assisted Ahmeduddin in debugging and refining the **Round Robin** implementation, focusing on data flow between the GUI and algorithm logic.
3. **Visualization**:
    - Integrated **Matplotlib** into the GUI to dynamically generate Gantt charts, providing clear and intuitive visualizations of process scheduling.
4. **Black-Box Testing**:
    - Conducted comprehensive **black-box testing** of the application, verifying the system's behavior against expected outputs without inspecting the code.
    - Tested multiple scenarios, including valid inputs, invalid inputs (e.g., missing fields, negative values), and edge cases like simultaneous arrivals.
5. **System Integration**:
    - Worked on the `main.py` file to connect the GUI with the scheduling algorithms, ensuring efficient input handling and result display.

Challenges Faced

- **Integration with Backend Code**: Passing user inputs from the GUI to backend algorithms and receiving accurate outputs was initially complex.
    - **Solution**: Collaborated with teammates to debug integration issues, testing each algorithm’s functionality iteratively.
- **Gantt Chart Implementation**: Ensuring a visually appealing and accurate chart was challenging.
    - **Solution**: Utilized online resources and tutorials on **Matplotlib**, iterating through designs to achieve the desired results.

Key Takeaways

This project enhanced my ability to design interactive user interfaces and strengthened my understanding of integrating frontend and backend components. The black-box testing approach helped me gain insights into system reliability and robustness.

---

2. Kyle Kissinger

Contribution to the Project

I focused extensively on implementing and testing the **First-Come-First-Serve (FCFS)** and **Shortest Job First (SJF)** scheduling algorithms while taking the lead on **code debugging** for critical issues. My primary contributions included:

1. **Algorithm Implementation**:
    - Developed the **FCFS** algorithm in `fcfs.py`, ensuring correct computation of waiting times, turnaround times, and completion times.
    - Implemented both **preemptive** and **non-preemptive SJF** algorithms in `sjf.py`, handling process execution efficiently and accurately.
2. **Testing and Debugging**:
    - Wrote unit tests for FCFS and SJF in `test_fcfs.py` and `test_sjf.py`.
    - Conducted thorough **code debugging**, identifying and resolving issues related to scheduling logic and integration with the GUI.
    - Debugged edge-case scenarios, such as processes arriving simultaneously or with identical burst times, to ensure reliability.
3. **Black-Box Testing**:
    - Performed black-box testing for FCFS and SJF, verifying outputs against expected results without directly inspecting the code.
    - Designed test cases for scenarios such as varying process arrival times, simultaneous arrivals, and extreme burst times.
4. **Collaboration**:
    - Worked closely with Mohammed on integrating FCFS and SJF algorithms into the GUI, ensuring smooth input-output handling.
    - Supported Ahmeduddin in debugging Priority Scheduling to handle edge cases effectively.

Challenges Faced

- **Preemptive SJF Complexity**: Accounting for interruptions during process execution added complexity to the logic.
    - **Solution**: Researched algorithm implementations and broke the logic into smaller, testable components.
- **Integration Issues**: Ensuring algorithm outputs were displayed correctly in the GUI required multiple debugging sessions.
    - **Solution**: Worked iteratively with Mohammed to identify and fix integration bugs.

Key Takeaways

This project provided valuable experience in CPU scheduling and sharpened my debugging skills. Black-box testing allowed me to view the system from the user's perspective, ensuring accuracy and usability.

---

3. Ahmeduddin Mohammed

Contribution to the Project

I took responsibility for implementing the **Round Robin (RR)** and **Priority Scheduling** algorithms, while also contributing to **black-box testing** and **code debugging**. My primary contributions included:

1. **Algorithm Implementation**:
    - Developed the **Round Robin** algorithm in `rr.py`, focusing on accurate application of the time quantum and handling process queues.
    - Implemented the **Priority Scheduling** algorithm in `priority.py`, ensuring fairness by resolving ties using arrival times.
2. **Optimization and Testing**:
    - Optimized both algorithms to handle large datasets efficiently and reduce execution time.
    - Wrote unit tests for RR and Priority Scheduling in `test_rr.py` and `test_priority.py`.
    - Conducted **black-box testing**, validating system outputs against expected behaviors for edge cases, including:
        - Processes with zero burst time.
        - Processes with identical priorities or simultaneous arrivals.
3. **Code Debugging**:
    - Collaborated with Mohammed and Kyle to debug integration issues between the algorithms and the GUI, particularly for Round Robin and Priority Scheduling.
    - Resolved performance bottlenecks in the Round Robin implementation by refining the logic for handling remaining burst times.
4. **Collaboration**:
    - Assisted Mohammed with debugging Gantt chart generation for Round Robin and Priority Scheduling outputs.

    - Worked with Kyle to ensure consistency across all algorithms and resolved discrepancies in shared logic.

Challenges Faced

- **Round Robin Complexity**: Accurately tracking remaining burst times and ensuring fairness in time-sharing was challenging.
    - **Solution**: Researched algorithm patterns and collaborated with teammates to refine the implementation.
- **Handling Priority Ties**: Processes with identical priorities required additional logic to ensure fairness.
    - **Solution**: Implemented a secondary sorting mechanism based on arrival times.

Key Takeaways

This project enhanced my problem-solving skills, particularly in implementing and optimizing algorithms. Debugging and black-box testing reinforced my ability to evaluate the system holistically and ensure correctness under various conditions.

---

4. Team Management

How the Group Managed the Project

- **Task Allocation**: Tasks were divided based on individual strengths:
    - Mohammed Aslam: GUI design and integration, assistance with FCFS and Round Robin, and black-box testing.
    - Kyle Kissinger: Implementation and testing of FCFS and SJF, extensive debugging, and black-box testing.
    - Ahmeduddin Mohammed: Implementation and testing of Round Robin and Priority Scheduling, black-box testing, and code debugging.
- **Collaboration**: Regular virtual meetings were held to discuss progress, troubleshoot issues, and plan next steps. Google Meet was used for discussions, and GitHub was used for version control.
- **Version Control**: Each team member worked on separate branches, which were merged into the main branch after thorough testing.

Challenges Faced

1. **Integration Issues**: Connecting the GUI to the backend logic required significant debugging.
    - **Solution**: All team members collaborated to ensure seamless communication between components.
2. **Time Management**: Balancing the project with other academic commitments was challenging.
    - **Solution**: We adhered to a strict timeline, focusing on completing one milestone at a time.

5. Submission Details

- **Project Submitted by**: **Ahmeduddin Mohammed**
- **Repository Link**: https://github.com/MohammedAhmeduddin/CPU_Scheduler

Conclusion

This project was an excellent opportunity to apply theoretical concepts to a practical problem. Each team member made significant contributions, ensuring a high-quality final product. Turnaround Time (TAT) in CPU scheduling measures the total time from process submission to completion, impacting system efficiency. Lower TAT indicates better performance, influenced by various scheduling algorithms. The collaborative effort, emphasis on debugging, and comprehensive testing have prepared us well for future challenges.

Acknowledgment

We would like to express our sincere gratitude to **Professor Larry Lay** for providing us with the opportunity to work on this project as part of the **CS630 Operating Systems Design** course. This project allowed us to gain a deeper understanding of CPU scheduling algorithms, system design, and collaborative development practices.

The guidance and knowledge imparted during the course were instrumental in the successful completion of this project. We are thankful for the invaluable learning experience and the skills we have developed through this assignment.

Thank you once again for your support and encouragement throughout this journey.

References

1. **Silberschatz, A., Galvin, P. B., & Gagne, G. (2018).** *Operating System Concepts* (10th ed.). Wiley. Link to book
2. **Python Tkinter Documentation** https://docs.python.org/3/library/tkinter.html A comprehensive guide to using Tkinter for GUI development in Python.
3. **Matplotlib Documentation: Plotting with Matplotlib** https://matplotlib.org/stable/contents.html Official documentation for Matplotlib, used for generating Gantt chart visualizations.