Rosita Jamalzade

ComputerScienceStudent

Email: Rositajamalzade@gmail.com
Phone : 09935256322
Location : *Tehran, Iran*
LinkedIn: linkedin.com/in/amanda-franklin-306569311/
GitHub : *github.com/diamond − johnson*

1 Professional Summary

Motivated Computer Science student at Amirkabir University of Technology with a strong foundation in programming languages and software development. Experienced in kernel modifications, system simulations, and collaborative tools like Git. Eager to apply technical skills in innovative projects while building expertise in areas like multi-threading and CPU scheduling. Seeking opportunities to contribute to real-world applications in software engineering.

2 Education

B.Sc. in Computer Science

Amirkabir University of Technology Tehran, Iran
September 2023 – June 2027

- Pursuing a rigorous curriculum focused on algorithms, operating systems, software engineering, and computer architecture.
- Relevant coursework includes Operating Systems, Data Structures, Programming Languages, and Database Systems.
- Actively involved in hands-on projects to deepen understanding of core CS concepts.

3 Work Experience

3.1 Translator

HybridAd Tehran, Iran
June 2022 – May 2023

- Provided high-quality Persian subtitles for English-language videos, ensuring accurate translation and cultural adaptation for diverse audiences.
- Collaborated with content teams to review and refine subtitles, improving accessibility and viewer engagement for multimedia projects.
- Managed multiple translation tasks under tight deadlines, demonstrating strong attention to detail and time management skills.
- Contributed to the localization process, enhancing the company's reach in Persian-speaking markets.

4 Projects

4.1 xv6 Multithreading Extension

June 2025
Technologies: C, xv6 architecture, QEMU, Git

- Extended the xv6 educational operating system kernel to support multi-threading, enabling concurrent execution of multiple threads within a single process to improve efficiency and resource utilization.
- Studied and modified core kernel components, including process management (creation, scheduling, context switching, and termination), and updated data structures like the ‘proc‘ struct to incorporate thread-specific features such as creation, joining, termination, and synchronization primitives.
- Addressed key concurrency challenges, including shared resource management, race conditions, and deadlock prevention, through careful kernel-level modifications and extensive testing.
- Built, ran, and debugged the modified kernel using QEMU in a virtual environment, ensuring stability and correctness across various test scenarios; version-controlled the project with Git for collaborative development.

4.2 CPU Scheduling Simulation

May 2025

Technologies: C, Build System

- Developed a comprehensive simulation in C to model and compare three essential CPU scheduling algorithms: First Come First Serve (FCFS), Shortest Job First (SJF, non-preemptive), and Round Robin (RR), providing insights into process execution and system performance metrics.
- Implemented process queues, time slices, and turnaround/waiting time calculations to accurately simulate real-world operating system behaviors, allowing for quantitative evaluation of scheduling efficiency.
- Utilized a modular build system to compile and test the simulation with various input scenarios, including burst times and arrival rates, to demonstrate trade-offs like throughput, response time, and fairness.
- Analyzed results to highlight algorithm strengths and weaknesses, such as FCFS's simplicity versus RR's better interactivity, preparing for deeper studies in operating systems.

4.3 Synchronization Problem: TA-Student Simulation

May 2025

Technologies: C, POSIX Threads, Standard Libraries, GCC compiler

- Adapted and simulated the classic Sleeping Barber synchronization problem to a Teaching Assistant (TA) helping students scenario, using POSIX threads (pthreads) to demonstrate thread coordination, resource sharing, and avoidance of race conditions in a multi-threaded environment.
- Implemented key synchronization primitives like mutexes, condition variables, and semaphores to manage the TA's availability, student queues, and office hours, ensuring threads wait appropriately without busy-waiting or deadlocks.
- Compiled and tested the simulation with GCC, evaluating scenarios with varying numbers of students and TA response times to illustrate concepts like producer-consumer patterns and critical sections in operating systems.

4.4 Multithreaded Sorting Application

April 2025

Technologies: C, POSIX Threads, Standard Libraries

- Designed and implemented a parallel sorting application using POSIX threads (pthreads) to divide an input array into two halves, sort each concurrently with quicksort, and merge the results for efficient large-scale data processing.
- Utilized thread creation, synchronization (via mutexes for the merge step), and joining to handle concurrent execution, reducing sorting time compared to single-threaded approaches while managing shared memory access safely.
- Tested with various array sizes and random data using standard C libraries, analyzing performance metrics like speedup and overhead to highlight the benefits and challenges of multithreading in algorithmic implementations.

5 Technical Skills

Programming Languages: Python, Java, JavaScript, C++, C, Matlab

Frameworks & Libraries: React

Tools & Technologies: Git, Docker, LaTeX, MySQL

Languages: English (Fluent), French (Beginner)