Guided OS Project: Multi-Level Feedback Queue Scheduler on xv6-RISC-V

Project Overview

In this 3-week guided project, students will implement a Multi-Level Feedback Queue (MLFQ) scheduler in the xv6 operating system for RISC-V. The default xv6 scheduler is a simple round-robin; this project replaces it with a priority-based scheduler with dynamic adjustments based on process behavior. The project is designed for teams of two undergraduate students and reinforces concepts from the MIT 6.828 labs.

Learning Objectives

- Understand xv6 scheduler and process context switching.
- Implement MLFQ scheduling with priority queues and demotion/promotion.
- Learn kernel-level C programming and xv6 internals.
- Create and use new system calls (e.g., getprocinfo).
- Test OS behavior with user-level programs.

Starter Resources

- xv6 RISC-V code: `https://pdos.csail.mit.edu/6.828/2025/xv6.html`
- Read Chapter 8 of the xv6 book (Scheduling).
- "Operating Systems: Three Easy Pieces" chapter on MLFQ.

Weekly Milestones

Week 1: Setup and Design

- Set up xv6 and understand the default scheduler.
- Implement `getprocinfo` system call and test it.
- Design the MLFQ scheduler: queue count, time quanta, demotion/promotion policy.
- Deliverables: 1-2 page design document, syscall output screenshot, queue scaffolding.

Week 2: MLFQ Scheduler Core

- Implement MLFQ in `proc.c` with 4 priority queues.
- Enforce time-slice quanta and round-robin within each level.
- Test demotion and yielding behavior using test programs.
- Deliverables: MLFQ code, demo results, updated design.

Week 3: Boosting and Polishing

- Implement starvation prevention (priority boosting).
- (Optional) Add `boostproc()` syscall for testing.
- Perform tests: CPU-bound vs I/O-bound process fairness.
- Deliverables: Final xv6 code, final report, demo results.

Grading Breakdown (100 points)

- Scheduler functionality: 50 pts
- Starvation prevention: 15 pts
- System calls/tools: 10 pts
- Code quality: 10 pts
- Testing/documentation: 10 pts
- Following guidelines: 5 pts

Instructor Tips

- Stick to single-core (`CPUS := 1`) for simplicity.
- Start with queues setup and simple round-robin for one level.
- Use debug prints and `getprocinfo` for validation.
- Be incremental; test each feature before proceeding.
- Plan well and divide responsibilities among team members.