MLFQ Scheduler - Week 2

Muhammad Hussain - 29004
Sarfaraz Ahmed - 24520

December 4, 2025

Overview

Week 2 implements the core Multi-Level Feedback Queue (MLFQ) scheduler in xv6–RISC-V. The round-robin scheduler is replaced with strict priority-based queue selection, time quantum enforcement, demotion/promotion, and a starvation-prevention boost. The design uses four queues with quanta $\{2,4,8,16\}$ ticks.

1 Queue Operations

Six key queue functions were implemented:

get_quantum(level)

Returns the quantum for each queue:

```
int quanta[] = {2,4,8,16};
int get_quantum(int level){ return quanta[level]; }
```

enqueue(p, level) / dequeue(level)

Simple linked-list queue: enqueue adds to tail, dequeue takes from head.

dequeue_specific(p)

Removes a process from any queue (O(n)). Used by sleep, exit, boost, and demotion.

demote_process(p)

If a process exhausts its quantum:

```
p->queue_level = min(p->queue_level+1, 3);
p->time_in_queue = 0;
```

priority_boost()

Every 100 ticks all RUNNABLE processes return to Q0, preventing starvation.

2 MLFQ Scheduler

The scheduler enforces strict priority: Q0 always runs before Q1, etc.

```
for(;;){
  if(cycle >= 100) priority_boost();
  cycle++;

  for(int lvl=0; lvl<4; lvl++){
    if(queue_heads[lvl]==0) continue;
    p = dequeue(lvl);

    if(p->state == RUNNABLE){
      p->state = RUNNING;
      swtch(&c->context, &p->context);
    }
    if(p->state == RUNNABLE)
      enqueue(p, p->queue_level);

    lvl = -1; // restart at Q0
  }
}
```

Key Properties

- **Strict priority:** Higher queues always preempt lower queues.
- **I/O fairness:** If a process yields early, it stays at its current level.
- **Starvation prevention:** Priority boost every 100 ticks.

3 Time Quantum Enforcement

The clock interrupt increments `time_in_queue` and triggers demotion on quantum expiry:

```
p->time_in_queue++;
if(p->time_in_queue >= get_quantum(p->queue_level)){
  demote_process(p);
  p->yielded = 1;
}
```

4 Process Lifecycle Integration

allocproc: New processes start in Q0. **wakeup:** I/O-bound processes keep their priority, improving responsiveness. **sleep/exit:** Processes are removed from queues cleanly via dequeue_specific().

5 Behavior Examples

CPU-bound

Uses entire quantum, demotes from Q0→Q1→Q2→Q3, stays at Q3 until next boost.

I/O-bound

Yields early and remains in Q0. Never demotes.

6 Starvation Prevention

Every 100 ticks:

- All RUNNABLE processes move to Q0.
- `time_in_queue` resets to 0.
- Ensures even Q3 processes eventually run.

7 Complexity

- enqueue: O(n) (tail pointer optimization possible)
- dequeue: O(1)
- dequeue_specific: O(n)
- boost: O(p) processes
- scheduler cycle: O(1) avg

Conclusion

Week 2 delivers a complete working MLFQ scheduler with strict priority queues, dynamic quantum-based demotion, I/O-sensitive behavior, starvation prevention, and full lifecycle integration. The system is stable, fair, and ready for performance testing in Week 3.