Chapter 8 Scheduling: The Multi-Level Feedback Queue

- Multi-Level Feedback Queue is one of the most well-known approaches to scheduling.
- The problem MLFQ tries to address:
  - optimizing turnaround time.
  - Make the system feel responsive to the user.

8.1 MLFQ: Basic Rules.

- MLFQ has distinct queues each assigned a different priority level.
  - The queue with the higher priority has jobs from it run first.
  - When two jobs are in the same queue MLFQ uses round robin.
- The key is in how the scheduler sets priorities.
  - MLFQ varies priority for each job based on observed behaviros.
  - If a job relinquishes CPU time while waiting for input the scheduler will keep the priority high.
  - If a job uses the CPU for long periods of intense activity the priority will be reduced.
- MLFQ will try to use history of behavior for a process to predict future behavior.

8.2 How to Change Priority.

- Consider a mix of interactive jobs:
  - short running (may frequently relinquish CPU)
  - longer running CPU bound jobs
    - response time isn't as important.
- First attempt at setting a priority adjustment algorithm:

    - R3: When a job enters the system it is placed at the highest priority.
    - R4a: If a job uses up an entire time slice while running its priority is reduced.
    - R4b: If a job gives up the CPU before the time slice ends it stays at the same priority.
- **A Single Long-Running Job**
    - When the job first enters the system it is at the highest priority.
    - As it goes through time slices it gets moved down the queue's to the lowest priority.
- **Second Shorter Job**
    - A long running job has been running a while and moved to the lowest queue.
    - A second shorter job is inserted into the highest queue.
        - The job completes before reaching the bottom queue.
        - The longer job resumes running.
- **I/O**
    - If a job is doign a lot of I/O the scheduler keeps it at the same priority level.
- **Problems with this setup**
    - Starvation:
        - If there are too many interactive jobs on the system they will combine to consume all of the CPU time.
        - Long running jobs will never get the CPU.
    - A programmer could write a program to take advantage of the scheduling to game the scheduler.
        - garnering more resources than should be

8.3 Priority Boost

- Change the rules to avoid starvation.

    - R5: After some period of time S move all jobs in the system to the highest priority queue
- Sovles two problems:
    - Processes are guaranteed not to starve.
    - If a CPU bound job becomes interactive it will be appropriately queued
- What should S be set to?
    - Getting S right requires some guessing.

8.4 Better Accounting

- Preventing gaming of the scheduler.
- Perform better accounting of the CPU time at each level of the MLFQ.
    - The scheduler should keep track of how much of a time slice a job uses.
    - R4: Once a job uses up its time allotment at a given level its priority is reduced.

8.5 Tuning MLFQ and Issues

- How do you parameterize a scheduler?
    - How many queues should there be?
    - How big should the time slices be?
    - How often should priority be boosted?
- Most MLFQ variants allow for different time-slize length across different queues
- Some MLFQ variants use a mathematical formula to adjust time-slices.