CPU Scheduling

- Basic Concepts
- Scheduling Criteria
- Scheduling Algorithms
  - FCFS, SJF, RR, Priority, Multiple-level
- Schedule Evaluation

Basic Concepts

- Long term scheduling: decide which jobs to select for the pool of active processes
- Medium term scheduling: decide to move some of the processes out of main memory
- Short term scheduling: decide which process to allocate to the CPU
- Context switch: the actions taken to make the selected process run on CPU, i.e. remove another process out of CPU

Objectives for Schedulers

- Good performance for the user
  - High throughput
  - Low turnaround time
  - Low response time
- Good performance for the system
  - CPU utilization
- Fairness
- Stability

CPU Scheduling Algorithm

- First Come First Served (FCFS)
- Shortest Job First (SJF)
- Priority
- Round Robin (RR)
- Multi-level Queues
- Multi-level with feedback
Problem 5 of midterm

- Process Burst Time Priority
  - \( P_1 \) 10 3
  - \( P_2 \) 1 1
  - \( P_3 \) 2 3
  - \( P_4 \) 1 4
  - \( P_5 \) 5 2
- The processes are assumed to have arrived in the order \( P_1, P_2, P_3, P_4, P_5 \), all at time 0.
- Draw Gantt graph, turnaround, waiting time for FCFS, RR, SJF, Priority
  - Turnaround: sum of periods spent waiting to get into memory, waiting in the ready queue, executing on CPU, and doing I/O
  - Waiting time: sum of periods spent waiting in the ready queue

Multi-level with feedback

- A new job enters queue 0 which is served FCFS. It receives 8 milliseconds. If it doesn’t finish in 8 milliseconds, it’s moved to queue 1
- At queue 1, job is also served FCFS and receives 16 additional milliseconds. If it doesn’t not complete, it’s moved to queue 2

Scheduling Evaluation

- Queuing Theory Analysis: uses well established mathematical models and techniques
- Simulation: create a model for the system and simulate its performance
- Empirical Experiments: implement and test in a real system

Midterm

- mean is 81.22.

Fork

- Causes a new process to be created with a duplicate of the parent’s code
- Fork returns a value of 0 to the child
- Fork returns the child’s PID (greater than 0) to the parent

Fork cont.

- \( \text{int value} = 5; \)
- \( \text{int main()} \)
  - \{ \)
    - \( \text{pid_t pid;} \)
    - \( \text{pid = fork();} \)
    - \( \text{if (pid == 0)} \)
      - \( \text{/* child process */} \)
      - \( \text{value++15;} \)
    - \}
    - \( \text{else if (pid > 0)} \)
      - \( \text{/* parent process */} \)
      - \( \text{wait(NULL);} \)
      - \( \text{printf(PARENT: value = %d, value);} \)
    - \( \text{exit(0);} \)
  - \}
  - \}