Lecture 5: finish sched / synchronization

- SRTF/Adaptive scheduling
- Real time scheduling
- Mutual exclusion
Round-robin

- like FCFS, periodically preempt processes
- how long should quantum be?
  - short quantum: lots of context switches
  - long quantum: potentially long delays but responsiveness
    - high avg. waiting time
    - (infinite quantum: FCFS)
Two kinds of processes:

- I/O bound processes
  - text editor
  - web server
  - need responsiveness: short quantum

- CPU-bound processes
  - data processing
  - need efficient use of CPU: long quantum

Ideal quantum

Higher priority (handle first)
Optimal algorithm: Shortest remaining time first (SRTF)

- Run whichever process will finish first (do I/O)

  P1: will run for 5
  P2: 5
  P3: 25

  P1, P2, P3  turnaround time  P1: 5  P3: 25  total: 50
  6  10  36

  P3, P2, P1  turnaround time  P1: 30  P3: 25  total: 90
  8  10  35

Pro:
- Optimal (waiting time)
  - If we assume user-facing procs are I/O bound:
  - Responsive
  - Minimal context switches

Con:
- Impossible to know how long procs will take
- Unfair if lots of short jobs continually arrive (long jobs would starve)
Adaptive multi-level scheduler

Idea: observe procs, I/O bound procs get short quota, CPU bound procs get long quota.

new procs (ones that returned from I/O)
go in high priority queue

- if proc uses whole quantum, move it to lower priority queue, move down.

- new procs (ones that returned from I/O) go in high priority queue.

divide time between queues:
- proc. high queue for 50 units
- med. queue for 50 units
- low queue for 50

high priority (quantum 10) → PCB → PCB

med. proc quantum = 25 → PCB → ...

low priority long quantum quantum = 50 → PCB → ...
Multiple processors

- want to schedule threads that communicate quickly (e.g. locking & unlocking) together (called gang scheduling)

- processor affinity: all things being equal, might be better to run same thread on same processor.

- fairness: want to ensure that all processes get time.
Real-time scheduling

- processes specify deadlines.
  - e.g.: audio processing (live)
  - e.g.: car, fighter jet, ...

- scheduler makes a schedule
  (can be optimal if runtimes are given)

- Soft/Hard RT sched.
  Soft: "best-effort" OS scheduler tries to
  meet constraints, but might not be possible
  
  hard: access control: scheduler can refuse requests
  for resources (CPU before deadline)
  - guarantee that demand is met.
Milk problem / mutual exclusion:
see typed lecture notes