Slide 1

Administrivia

- Reminder: Homework 1a, 1b due Friday (was today but I'm allowing two more days because of lack of office hours).
- As noted in e-mail, reading quiz 1 graded and sample solution and grades uploaded. I was generous with points, so I say worth looking at sample solution even if you did well.
- More assignments coming soon.

Slide 2

CPU Scheduling — Review/Recap

- Several situations in which we need to, or might want to, switch between processes.
- Nuts and bolts of how — "mechanism" that doesn't change. What changes are the decisions — "policy".
- Which policy is best? Depends on several factors — workload, what we mean by "best" ("metric").
- Discussion in terms of "jobs" and starts with many unrealistic assumptions, later relaxed one by one.

Slide 3

Simple Scheduling Algorithms

- Previous class: FCFS, SJF, SJF with preemption, round robin.
- Two more the textbook doesn't discuss . . .

Slide 4

Priority Scheduling

- Basic ideas:
  - Keep a queue of ready processes, as before.
  - Assign a priority to each process.
  - When a process starts (or, if we also allow for blocking, when it becomes unblocked), add it to the end of the queue.
  - Switch when the running process exits (or blocks), or possibly when a process starts. (I.e., preemption may be allowed.)
  - Next process is the one with the highest priority.
- Points to consider:
  - What happens to low-priority processes? (So, maybe we should change priorities sometimes?)
  - How do we decide priorities? (external considerations versus internal characteristics)

Slide 5

Multiple-Queue Scheduling

- Basic idea — variant on priority scheduling:
  - Divide processes into "priority classes".
  - When picking a new process, pick one from the highest-priority class with ready processes.
  - Within a class, use some other algorithm to decide (round-robin, e.g.).
  - Optionally, periodically lower processes' priorities.
- If we let the algorithm manage priorities, then we have . . .

Slide 6

Multi-Level Feedback Queue Scheduling

- Same idea as MLQ, just with some details specified — how priorities are initially set and how they change.
- Goal is to balance good response time and good turnaround time, sometimes at odd.
- Key idea is one used in many contexts in computer science: It often works to assume that the near future will be much like the recent past.
- How this applies: A process that hasn't had long CPU bursts recently won't start having them in the near future.
- Textbook goes into some detail; I'll skip for now, except to note: "Gaming the system" (to get more than one's share of computing resources) is a real thing!

Slide 7

"Fair Share" Algorithms

- (Skim these too for now.)
- Lottery scheduling:

  Give each process one or more "lottery tickets" — more or fewer depending on its priority (so to speak); pick one at random to decide who's next.
- Guaranteed scheduling (as used in Linux Completely Fair Share?):

  "Guarantee" each process (of N) 1/N of the CPU cycles; (try to) schedule to make this true.

  Calculate, for each process, fraction of the time it has had the CPU in its lifetime, fraction it "should" have had; choose process for which actual time / entitled time is smallest.

Slide 8

Sidebar — Simulating Scheduling Algorithms

- Can be helpful in understanding how these algorithms work to simulate what they do given a particular sequence of inputs.
- Example — batch system with the following jobs.

job ID	running time	arrival time
A	6	0
B	4	0
C	10	0
D	2	2

Asked to compute turnaround times for all jobs using FCFS, what would you do . . .

Slide 9

Minute Essay

- Questions? How are you doing with the homework?