Slide 1

Administrivia

- A question: One student commented in a previous minute essay that the questions on reading quizzes and homeworks didn't seem to match up with the reading mentioned in the assignment — the material could be found, just not in the specified chapter. Do you agree? I don't mean for that to happen, but since the textbook is online it could have changed without my knowing about it. Examples might help me know what to avoid — ?

Slide 2

Multi-Level Feedback Queue (MLFQ) Scheduling — A Bit More

- Good turnaround time and good response time seem to pull in different directions. MLFQ tries to meet both goals.
- Starts out with two basic rules, like the ones I showed for Multi-Level Queue and then adds others: How do we set priorities initially? When should they be increased? When decreased? Textbook's presentation seems clear if long.
- One thing worth noting — "voo-doo" constants. Term adopted from someone who taught the course the authors took on operating systems, but not a new idea. May be a general rule here — when the "right" or "best" value for something isn't clear, make it a parameter and let (future) sysadmin decide. Another way of separating mechanism from policy?

Slide 3

Proportional-Share Scheduling

- Lottery scheduling: Give each process “tickets”, more for higher priority, and choose randomly. Surprising that this works, but similar approaches work in other contexts too (e.g., “Monte Carlo” in simulations).
- Stride scheduling: Similar idea with tickets, but choose in what I’d describe as weighted round robin.
- With both approaches, open question how many tickets to give each process.
- Linux CFS (Completely Fair Scheduler): Fairly detailed presentation. Not particularly important for its own sake (I say), but interesting as an example of how complicated a real-world scheduler can be. So many details!

Slide 4

Linux CFS

- A key goal — spend less time scheduling, since apparently it matters. LCFS tries to minimize through good design, good choice of data structures(!).
- Basic idea: Track (virtual) running time of each process, and when it’s time to choose a new one, choose the one with the smallest one of those.
- How often to consider switching? Tunable “latency” parameter.
- Minimum time between switches? Tunable “granularity” parameter.
- (Potentially) gets control on periodic timer interrupts only.

Slide 5

Linux CFS, Continued

- Allows users to affect priority with UNIX `nice` and `renice` commands, via complicated weighting scheme.
- Tries to deal gracefully with unusual(?) cases, such as processes just waking up from being blocked.

Slide 6

CPU Scheduling — Wrap-Up

- Who knew it could be so complicated? now you do.
- Some amusing things in bibliographies. This textbook can be fun!

Slide 7

Virtualizing the CPU — Wrap-Up

- Nice summary in Chapter 11 of textbook.
- To me the key idea is the mental image of arbitrarily many virtual CPUs (one per process), each with an associated address space. And then the challenge is mapping this image onto the computer's actual hardware, and all the details flow from that. (But what a lot of details!)
- So the O/S is both provider of virtual machine and manager of physical resources.
- Note the interplay between hardware and software.
- Note the level of paranoia involved. Inevitable? Recall my two well-aged war stories.

Slide 8

Minute Essay

- What stands out for you about this group of chapters about virtualizing the CPU? Do you feel like there's something you understand now that you didn't before?
- Do you recognize the name Edsger Dijkstra (from one bibliography entry), and if so from what context? (Cultural(?) aside: You've heard the claim "goto considered harmful"? It's from a letter to the editor of *CACM* he wrote.)