Slide 1

Administrivia

- Reminder: Homework 3 due today. Homework 2 past due.
- Reminder: Midterm Monday. Review sheet on Web. Campus network may not be available Saturday night / Sunday.
- Sample solutions for written problems available in hardcopy only (outside my office door). Sample solutions for programming problems on the Web (early tomorrow).

Slide 2

Evaluating Scheduling Algorithms

- How to decide which scheduling algorithm to use?
- One way — evaluate several choices, see which one best meets system goal(s). E.g., if the goal is minimum turnaround time, try to come up with an average turnaround time for each proposed choice.
- Several approaches possible . . . (This discussion is from another operating systems textbook, by Silberschatz and Galvin.)

Deterministic Modeling

- Idea — use a predetermined workload, compute values of interest (e.g., average turnaround time, as in Homework 3 problem).
- How well does it work?

Deterministic Modeling, Continued

- Simple, fast, gives exact numbers.
- Requires exact numbers as input, and only applies to them.

Slide 5

Queueing Models

- Idea — use “queueing theory” to model system as a network of “servers”, each with a queue of waiting processes. (E.g., CPU is a server, with input queue of ready processes.)
- Input to model — distribution of process arrival times, CPU and I/O bursts for processes, as mathematical formulas. (Base this on measuring, approximating, or estimating.) In queueing-theory terms, “arrival rates” and “service rates”.
- Queueing theory lets you then compute utilization, average queue length, average wait time, etc.
- How well does it work?

Slide 6

Queueing Models, Continued

- Seems more general than deterministic modeling.
- But can be tricky to set up model correctly, and need to approximate / make assumptions may be a problem.

Slide 7

Simulations

- Idea — program a model of the computer system, simulating everything, including hardware.
- Two ways to get input for simulation:
  - Generate processes, burst times, arrivals, departures, etc., using probability distributions and random-number generation.
  - Create "trace tape" from running system.
- How well does it work?

Slide 8

Simulations, Continued

- Potentially very accurate.
- Time-consuming to program and to run!

Slide 9

Implementation

- Idea — code it up and try it!
- How well does it work?

Slide 10

Implementation, Continued

- Seems like potentially the most accurate approach.
- Requires a lot of work, resources.
- Involves implicit assumption that users' behavior is fairly constant.

  (So it's good to build into the algorithm some parameters that can be changed at run time, by users and/or sysadmin. In textbook's phrase, "separate mechanism from policy". Notice, though, users are apt to figure out how to game any system.)

Slide 11

Recap — Scheduling Algorithms

- Main idea — decide which process to run next (when running process exits, becomes blocked, or is interrupted).
- Many possibilities, ranging from simple to complex. Real systems seem to use hybrid strategies.
- How to choose one?
  - Be clear on goals.
  - Maybe evaluate some possibilities to see which one(s) meet goals — analytic or experimental evaluation.
  - Build in some tuning knobs — “separate policy from mechanism”.

Slide 12

About the Midterm

- Review class notes and minute essays, readings. If I didn’t mention it in class, odds are I won’t ask about it on the exam.
- Questions will be a mix of problems similar to those in homework (but shorter), mini-essay, and multiple choice.
- Open book, open notes. Okay to point a browser at the course Web site, but no other Web access.
- (Topic by topic through the review sheet.)
- (Review homework solutions.)
- (Review minute essays.)

Slide 13

Quotes of the Day/Week/?

- From a key figure in the early days of computing:

  "As soon as we started programming, we found to our surprise that it wasn't as easy to get programs right as we had thought. Debugging had to be discovered. I can remember the exact instant when I realized that a large part of my life from then on was going to be spent finding mistakes in my own programs." (Maurice Wilkes: 1948)

- From someone in a discussion group for the Java programming language:

  "Compilers aren't friendly to anybody. They are heartless nitpickers that enjoy telling you about all your mistakes. The best one can do is to satisfy their pedantry to keep them quiet :)"

Slide 14

Minute Essay

- What have you found interesting/educational about the homework?