

Slide 1

Administrivia

- Exam 1 next Thursday. Review sheet on Web. Briefly:
 - Chapters 1, 2, and 3. Questions similar to homeworks, plus multiple-choice and true/false.
 - Open (text)book, open notes. No calculators, but you should not need one.
- Review session possible, if there's interest and we can find a time that works.
- Homework 2 due Tuesday. Not accepted late. Solutions available Wednesday.

Slide 2

Scheduling in Real-Time Systems

- “Real-time system” — system in which events must (“hard real time”) or should (“soft real time”) be handled by some deadline. Often events to be handled are periodic, and we know how often they arrive and how long they take to process.
- Role of scheduler in such systems could be critical.
- An interesting question — sometimes getting everything scheduled on time is impossible (example?). If we know periodicity and time-to-handle of all types of events, can we decide this?
Suppose we have m types of events, and event type i has period P_i and time-to-handle C_i .
Derive formula on p. 149 . . .
- Complex topic, see chapter 7 for more info.

One More Recap — Scheduling Algorithms

Slide 3

- 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”.

Deadlocks — Introduction

Slide 4

- Some resources should not be shared — among processes, computers, etc.
- To enforce this, o/s (or whatever) provides mechanism to give one process at a time exclusive use, make others wait.
- Possibility exists that others will wait forever — deadlock.

Resources

Slide 5

- “Resource” is anything that should be used by only one process at a time — hardware device, piece of information (e.g., database record), etc.
Can be unique (e.g, particular database record) or non-unique (e.g., one block of a fixed-size disk area such as swap space).
- Preemptible versus non-preemptible — preemptible resources can be taken away from current owner without causing something to fail (e.g., memory); non-preemptible resources can’t (e.g., hardware device).
- Normal sequence for using a resource — request it, use it, release it. If not available when requested, block or busy-wait.
Can easily implement this using semaphores, but then deadlock is possible if processes aren’t disciplined.

Deadlocks — Definitions and Conditions

Slide 6

- Definition — set of processes is “deadlocked” if each process in set is waiting for an event that only another process in set can cause.
- Necessary conditions:
 - Mutual exclusion — resources can be used by at most one process at a time.
 - Hold and wait — process holding one resource can request another.
 - No preemption — resources cannot be taken away but must be released.
 - Circular wait — circular chain of processes exists in which each process is waiting for resource held by next.
- Modeling deadlock — “resource graphs” — examples pp. 165-166.

What To Do About Deadlocks — Nothing

- One strategy for dealing with deadlocks — “ostrich algorithm” (ignore potential for deadlocks, hope they don’t happen).
- Does this work? not always, but simple to implement, and in practice works most of the time.

Slide 7

What To Do About Deadlocks — Detection and Recovery

- How to detect deadlocks — DFS on resource graph, (or if more than one resource of each type, algorithm of pp. 171–172).
- When to check for deadlocks:
 - Every time a resource is requested.
 - At regular intervals.
 - When CPU utilization falls below threshold.
- What to do if deadlock is found?
 - Preemption.
 - Rollback.
 - Process termination.
- Does this work? yes, but potentially time-consuming, and “what to do” choices aren’t very attractive!

Slide 8

What To Do About Deadlocks — Avoidance

Slide 9

- Can base on idea of “safe” states (in which it’s possible to schedule to avoid deadlock) versus “unsafe” states (in which it’s not). Idea is to avoid unsafe states.
- “Banker’s algorithm” (Dijkstra, 1965) — idea is to never satisfy request for resource if it leads to unsafe state. Details on pp. 178–179.
- Does this work? yes, but not much used because it assumes a fixed number of processes, resource requirements known in advance.

What To Do About Deadlocks — Prevention

Slide 10

- Idea here is to make it impossible to satisfy one of the four conditions for deadlock.
- Mutual exclusion — don’t allow more than one process to use a resource. E.g., define a printer-spool process to manage printer. Solves immediate problem but may produce others.
- Hold and wait — require processes to request all resources at the same time and either get them all or wait. Works but may not be possible or efficient.
- No preemption — allow preemption. Not usually possible/desirable.
- Circular wait — impose strictly increasing ordering on resources, and insist that all processes request resources “in order”. Works, but finding an ordering may be difficult.

Deadlocks — Summary

- Take-home message — there's some interesting theory related to this topic, but not a lot of practical advice, except for deadlock prevention.

Slide 11

Minute Essay

- Any questions about the homeworks and exam? (If you think of something over the weekend, send me mail.)
- (And have a nice fall break, such as it is!)

Slide 12