

### Administrivia

- Reminder: Homework 6 due Friday.
- Homework 7 on Web; due Monday.

Slide 1

### Deadlocks — Introduction

- 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.

Slide 2

## Resources

Slide 3

- “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 4

- 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”.
- What do about them? Various approaches.

### 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?

Slide 5

### Do Nothing, Continued

- Doesn’t always work, of course.
- But simple to implement, and in practice works most of the time.

Slide 6

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 from text).
- 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?

Slide 8

### Detection and Recovery, Continued

- Does work.
- But potentially time-consuming, and “what to do” choices aren’t very attractive!

### 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.
- Does this work?

### Avoidance, Continued

Slide 10

- Does work.
- But not much used because it assumes a fixed number of processes, resource requirements known in advance.

### What To Do About Deadlocks — Prevention

Slide 11

- 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.
  - Hold and wait — require processes to request all resources at the same time and either get them all or wait.
  - No preemption — allow preemption.
  - Circular wait — impose strictly increasing ordering on resources, and insist that all processes request resources "in order".
- Do these work?

### Prevention, Continued

Slide 12

- Don't allow more than one process to use a resource:  
Solves immediate problem but may produce others.
- 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.
- Allow preemption.  
Not usually possible/desirable.
- Impose strictly increasing ordering on resources, and insist that all processes request resources "in order".  
Works, but finding an ordering may be difficult.

### Deadlocks — Related Issues

- Classical description is in terms of “resources”, but other kinds of deadlock are possible (e.g., involving communication).
- Other situations that aren’t classical deadlock but are also not good include “livelock” and “starvation” (see text).

Slide 13

### 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 14

### Minute Essay

- What's the smallest number of resources needed to have a deadlock?

Slide 15

### Minute Essay Answer

- Two — with only one, a process may wait a long time for another process to release it, but that's not true deadlock.

Slide 16