Dealing with Deadlock

Dr. Sean Peisert
Guest Lecture for UCD ECS 150 (Operating Systems)
June 2, 2008

peisert@cs.ucdavis.edu
http://www.sdsc.edu/~peisert
Dealing with Deadlock

1. Ignore it
   - Hope it’s rare and that your users can recover from it

2. Detect & recover
   - e.g., look for a cycle in dependencies

3. Prevent it
   - Make it impossible for deadlock to happen

4. Avoid it
   - Control allocation of resources
Ignore It
Dealing with Deadlock

1. Ignore it
   - Hope it’s rare and that your users can recover from it

2. Detect & recover
   - e.g., look for a cycle in dependencies

3. Prevent it
   - Make it impossible for deadlock to happen

4. Avoid it
   - Control allocation of resources
Detect

- Traverse resource graph

- If a cycle is found, force a process to release
  
  - Preempt (and rollback)
  
  - Abort

- This is expensive
3. Prevent

• Ensure *at least one* of the following fails:

  • Mutual exclusion
  • No pre-emption
  • Circular wait / resource waiting
  • Hold and wait / partial allocation
4. Avoid

• Determine the resource needs of processes in advance.

• System only grants resources if it can determine that the process can have everything in advance.

• This is hard (and usually not practical)
Banker’s Algorithm Example
Banker’s Algorithm Example Part 1

• Condition:

  • 10 resource units

  • 3 processes (P, Q, R)

• P has 4 units and needs 4 more

• Q has 2 units and needs 1 more

• R has 2 units and needs 7 more
Banker’s Algorithm Example Part 2

- Condition:
  - 10 resource units
  - 3 processes (P, Q, R)
  - P has 4 units and needs 4 more
  - R has 2 units and needs 7 more
Banker’s Algorithm Example Part 3

• Condition:
  
  • 10 resource units
  
  • 3 processes (P, Q, R)
  
  • R has 2 units and needs 7 more
Second Banker’s Algorithm Example Part 1

• Condition:

  • 10 resource units

  • 3 processes (P, Q, R)

• P has 4 units and needs 4 more

• Q has 2 units and needs 1 more

• R has 3 units and needs 6 more
Second Banker’s Algorithm Example Part 2

- Condition:
  - 10 resource units
  - 3 processes (P, Q, R)
  - P has 4 units and needs 4 more
  - R has 3 units and needs 6 more
Problems with the Banker’s Algorithm

- Fixed number of resources
- Fixed number of processes
- Guarantees requests will be granted in a finite time
- Requires jobs to release resources in a finite time
- Requires uses to know and state needs in advance
Questions?

- Email: peisert@cs.ucdavis.edu
- Web: http://www.sdsc.edu/~peisert