CSE 120: Principles of Operating Systems

Lecture 6

Deadlocks

October 14, 2003

Prof. Joe Pasquale
Department of Computer Science and Engineering
University of California, San Diego

© 2003 by Joseph Pasquale
Before We Begin ...

Read Chapter 8 (on Deadlock)

Midterm is on Tuesday of next week (Oct 21)

- Covers all Lectures including this week
- Covers Programming Assignment #1
- Sample midterm posted on class web page
What is Deadlock?

The state of a set of permanently blocked processes

- Unblocking of one relies on progress of another
- But none can make progress!

Example

- Processes A and B
- Resources X and Y
- A holding X, waiting for Y
- B holding Y, waiting for X
- Each is waiting for the other; will wait forever
Traffic Jam as Example of Deadlock

Cars deadlocked in an intersection

Resource Allocation Graph
More Examples of Deadlock

Memory (a reusable resource)

- total memory = 200KB
- $P_1$ requests 80KB
- $P_2$ requests 70KB
- $P_1$ requests 60KB (wait)
- $P_2$ requests 80KB (wait)

Messages (a consumable resource)

- $P_1$: receive $M_2$ from $P_2$
- $P_2$: receive $M_1$ from $P_1$
Conditions for Deadlock

Mutual Exclusion
- Only one process may use a resource at a time

Hold-and-Wait
- Process holds resources while waiting for others

No Preemption
- Can’t take a resource away from a process

Circular Wait
- The waiting processes form a cycle
How to Attack the Deadlock Problem

Deadlock Prevention
- Prevent any possibility of a deadlock

Deadlock Avoidance
- Avoid situations that lead to deadlock

Deadlock Detection
- Don’t try to stop deadlocks
- Rather, if they happen, detect and resolve
Deadlock Prevention

Simply prevent any one of the conditions for deadlock

- **Mutual exclusion**
  - Relax where sharing is possible

- **Hold-and-wait**
  - Get all resources simultaneously, wait until all free

- **No preemption**
  - Allow resources to be taken away

- **Circular wait**
  - Order all the resources, force ordered acquisition
Preventing a Traffic Jam

To apply deadlock prevention to traffic jam problem, just add a traffic light!

Which condition is being prevented?
Deadlock Avoidance

Works with *incremental* resource requests

- Resources are asked for in increments
- Do not grant request that can lead to a deadlock

Requires knowledge of maximum resource requirements
Banker's Algorithm: Concepts

System has a fixed number of processes and resources
- each process has zero or more resources allocated

System state: either safe or unsafe
- depends on allocation of resources to processes

Safe state
- can avoid deadlock by certain order of execution

Unsafe state
- deadlock is possible (but not necessarily certain)
Safe, Unsafe, and Deadlock States
Banker’s Algorithm

Given

- a process/resource claim matrix
- a process/resource allocation matrix
- a resource availability vector

Is there a sequence of process executions such that

- a process can run to completion, return resources
- resources can then be used by another to complete
- eventually, all the processes complete?
# Example of a Safe State

**Current state**

<table>
<thead>
<tr>
<th></th>
<th>Claim</th>
<th>Allocation</th>
<th>Avail</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$P_1$</td>
<td>$P_2$</td>
<td>$P_3$</td>
<td>$P_4$</td>
</tr>
<tr>
<td>$R_1$</td>
<td>3</td>
<td>6</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>$R_2$</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>$R_3$</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>2</td>
</tr>
</tbody>
</table>

This is a Safe State

- Who can run to completion? $P_2$
- After $P_2$ completes, it's resources are returned
- Next select $P_1$, then $P_3$, then $P_4$
Example of an Unsafe State

Current state

<table>
<thead>
<tr>
<th></th>
<th>Claim</th>
<th>Allocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>P_1, P_2, P_3, P_4</td>
<td>P_1, P_2, P_3, P_4</td>
<td>Avail</td>
</tr>
<tr>
<td>R_1 3 6 3 4</td>
<td>1 5 2 0</td>
<td>1 9</td>
</tr>
<tr>
<td>R_2 2 1 1 2</td>
<td>0 1 1 0</td>
<td>1 3</td>
</tr>
<tr>
<td>R_3 2 3 4 2</td>
<td>0 1 1 2</td>
<td>2 6</td>
</tr>
</tbody>
</table>

So far, this is a safe state, but ...

- if P_2 asks for 1 unit of R_1 and 1 unit of R_3: safe
- if P_1 asks for 1 unit of R_1 and 1 unit of R_3: unsafe
- no one may be able to complete: possible deadlock
Avoiding a Traffic Jam

To apply deadlock avoidance to traffic jam problem, allow at most 3 cars to enter intersection.

What kind of request is being denied?
Deadlock Detection and Recovery

Don’t do anything special to prevent or avoid deadlocks

- If they happen, they happen
- Periodically, try to detect if a deadlock occurred
- Do something (or even nothing) about it

Reasoning

- Deadlocks rarely happen
- Cost of prevention or avoidance is not worth it
- Deal with them in special way (may be very costly)

Most systems take this approach!
Detecting Deadlocks

Construct “wait-for” graph
- Construct resource allocation graph
- Remove resource nodes
- If cycle, deadlock

Requires
- identifying all resources and tracking their use
- periodically running detection algorithm
Recovery from Deadlock

Abort all deadlocked processes
  • Will remove deadlock, but drastic and costly

Abort deadlocked processes one-at-a-time
  • Do so until deadlock goes away (need to detect)
  • What order should processes be aborted?

What happens to resources in inconsistent states?
  • files partially written
  • interrupted message (e.g., file) transfer