Chapter 8: Deadlocks

- Deadlock Characterization
- Methods for Handling Deadlocks
- Deadlock Prevention
- Deadlock Avoidance
- Deadlock Detection
- Recovery from Deadlock

Bridge Crossing Example

- Bridge has one lane
- Deadlock occurs when 2 cars of opposing direction meet
- Several cars may have to be backed up if a deadlock occurs
- Starvation is possible

The Deadlock Problem

- A set processes each holding a resource and waiting to acquire a resource held by another process in the set
- Example
  - System has 2 tape drives
  - \( P_1 \) and \( P_2 \) each hold one tape drive and each needs another one
- Example
  - Semaphores \( A \) and \( B \), initialized to 1
  
  \[
  \begin{align*}
  &P_0 & P_1 \\
  &\text{wait}(A); & \text{wait}(B); \\
  &\text{wait}(B); & \text{wait}(A)
  \end{align*}
  \]
Deadlock Conditions

- Mutual exclusion:
  - only one process at a time can use a resource
- Hold and wait:
  - a process holding at least one resource is waiting to acquire additional resources held by other processes
- No preemption:
  - a resource can be released only voluntarily by the process holding it, after that process has completed its task
- Circular wait:
  - there exists a set \( \{P_0, P_1, \ldots, P_n\} \) of processes such that:
    - \( P_0 \) is waiting for a resource that is held by \( P_1 \)
    - \( P_1 \) is waiting for a resource that is held by \( P_2 \)
    - \( \ldots \)
    - \( P_{n-1} \) is waiting for a resource that is held by \( P_n \)
    - \( P_n \) is waiting for a resource that is held by \( P_0 \)

System Model

- Processes \( P_1, P_2, \ldots, P_n \)
- Resource types \( R_1, R_2, \ldots, R_m \)
- CPU cycles, memory space, I/O devices
- Each resource type \( R_i \) has \( W_i \) instances
- Each process utilizes a resource as follows:
  - request
  - use
  - release

Resource-Allocation Graph

A set of vertices \( V \) and a set of edges \( E \):

- \( V \) is partitioned into two types:
  - \( P = \{P_1, P_2, \ldots, P_n\} \)
    - set of processes in the system
  - \( R = \{R_1, R_2, \ldots, R_m\} \)
    - set resource types in the system
- request edge – directed edge \( P_i \to R_j \)
- assignment edge – directed edge \( R_j \to P_i \)
Resource-Allocation Graph (Cont.)
- Process
- Resource Type with 4 instances
- \( P_i \) requests instance of \( R_j \)
- \( P_i \) is holding an instance of \( R_j \)

Example of a Resource Allocation Graph

Resource Allocation Graph With A Deadlock
Basic Facts

- If graph contains no cycles \( \Rightarrow \) no deadlock
- If graph contains a cycle \( \Rightarrow \)
  - if only one instance per resource type, then deadlock
  - if several instances per resource type, possibility of deadlock

Methods for Handling Deadlocks

- Ensure that the system will never enter a deadlock state
- Allow the system to enter a deadlock state and then recover
- Ignore the problem
**Deadlock Prevention**

Limit ways request can be made:
- Request must contain all resources
  - Require process to request and be allocated all its resources before it begins execution
  - Release all held resources on a new request
- Allow resource preemption
- Impose global ordering on resources:
  - Resource must be requested in order

**Deadlock Avoidance**

Requires that the system has some additional a priori information available:
- Simplest and most useful model requires that each process declare the maximum number of resources of each type that it may need
- The deadlock-avoidance algorithm dynamically examines the resource-allocation state to ensure that there can never be a circular-wait condition
- Resource-allocation state is defined by the number of available and allocated resources, and the maximum demands of the processes

**Deadlock Detection**

- Allow system to enter deadlock state
- Detection algorithm
  - Check for cycles in resource allocation graph
  - How often?
    - How often a deadlock is likely to occur?
- Recovery scheme
Recovery from Deadlock: Process Termination

- Abort all deadlocked processes
- Abort one process at a time until the deadlock cycle is eliminated
- In which order should we choose to abort?
  - Priority of the process
  - How long process has computed, and how much longer to completion
  - Resources the process has used
  - Resources process needs to complete
  - How many processes will need to be terminated
  - Is process interactive or batch?

Recovery from Deadlock: Resource Preemption

- Selecting a victim – minimize cost
- Rollback – return to some safe state, restart process for that state
- Starvation – same process may always be picked as victim, include number of rollback in cost factor

Combined Approach to Deadlock Handling

- Combine the three basic approaches
  - prevention
  - avoidance
  - detection
  - allowing the use of the optimal approach for each of resources in the system
- Partition resources into hierarchically ordered classes
- Use most appropriate technique for handling deadlocks within each class