Chapter 8 Deadlocks

- What is a Deadlock?
- Deadlocks in Resource Allocation
- Handling Deadlocks
- Deadlock Prevention
- Deadlock Avoidance
- Deadlock Handling in Practice

What is a Deadlock?

**Definition 8.1 Deadlock**  A situation involving a set of processes $D$ in which each process $P_i$ in $D$ satisfies two conditions:
1. Process $P_i$ is blocked on some event $e_i$.
2. Event $e_i$ can be caused only by actions of other process(es) in $D$.

- **Resource deadlock** $\leftarrow$ primary concern of OS

<table>
<thead>
<tr>
<th>Process $P_i$</th>
<th>Process $P_j$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Request tape drive;</td>
<td>Request printer;</td>
</tr>
<tr>
<td>Request printer;</td>
<td>Request tape drive;</td>
</tr>
<tr>
<td>Use tape drive and printer;</td>
<td>Use tape drive and printer;</td>
</tr>
<tr>
<td>Release printer;</td>
<td>Release tape drive;</td>
</tr>
<tr>
<td>Release tape drive;</td>
<td>Release printer;</td>
</tr>
</tbody>
</table>

- $P_i, P_j$ are deadlocked after their second requests
- Deadlocks can also arise in synchronization and message communication $\leftarrow$ user concern
Deadlocks in Resource Allocation

- OS may contain several resources of a kind
  - *Resource unit* refers to a resource of a specific kind
  - *Resource class* refers to the collection of all resource units of a kind
- Resource allocation in a system entails three kinds of events:
  - *Request* for the resource
  - *Actual allocation* of the resource
  - *Release* of the resource
    - Released resource can be allocated to another process

---

**Table 8.1** Events Related to Resource Allocation

<table>
<thead>
<tr>
<th>Event</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Request</td>
<td>A process requests a resource through a system call. If the resource is free, the kernel allocates it to the process immediately; otherwise, it changes the state of the process to <em>blocked</em>.</td>
</tr>
<tr>
<td>Allocation</td>
<td>The process becomes the <em>holder</em> of the resource allocated to it. The resource state information is updated and the state of the process is changed to <em>ready</em>.</td>
</tr>
<tr>
<td>Release</td>
<td>A process releases a resource through a system call. If some processes are blocked on the allocation event for the resource, the kernel uses some tie-breaking rule, e.g., FCFS allocation, to decide which process should be allocated the resource.</td>
</tr>
</tbody>
</table>
Conditions for a Resource Deadlock

<table>
<thead>
<tr>
<th>Condition</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nonshareable resources</td>
<td>Resources cannot be shared; a process needs exclusive access to a resource.</td>
</tr>
<tr>
<td>No preemption</td>
<td>A resource cannot be preempted from one process and allocated to another process.</td>
</tr>
<tr>
<td>Hold-and-wait</td>
<td>A process continues to hold the resources allocated to it while waiting for other resources.</td>
</tr>
<tr>
<td>Circular waits</td>
<td>A circular chain of hold-and-wait conditions exists in the system; e.g., process $P_i$ waits for $P_j$, $P_j$ waits for $P_k$, and $P_k$ waits for $P_i$.</td>
</tr>
</tbody>
</table>

Another condition is also essential for deadlocks:
- **No withdrawal of resource requests**: A process blocked on a resource request cannot withdraw its request.

Modeling the Resource Allocation State

- **(Resource) allocation state**:  
  - Information about resources allocated to processes and about pending resource requests  
  - Used to determine whether a set of processes is deadlocked  

Two kinds of models are used to represent the allocation state of a system:
- **A graph model**: for systems in which a process can request exactly one resource unit of each resource class  
- **A matrix model**: for systems that permit a process to request any number of units of a resource class
Resource Request and Allocation Graph (RRAG)

- Two kinds of nodes exist in an RRAG
  - A circle is a process
  - A rectangle is a resource class
    - Each bullet in a rectangle is one resource unit
- Edges can also be of two kinds
  - An edge from a resource class to a process indicates a resource allocation
  - An edge from a process to a resource class indicates a pending resource request

Wait-For Graph (WFG)

- A WFG can be used to depict the resource state of a system in which every resource class contains only one resource unit
  - A node in the graph is a process
  - An edge is a wait-for relationship between processes
    - A wait-for edge \((P_i, P_j)\) indicates that
      - Process \(P_j\) holds the resource unit of a resource class
      - Process \(P_i\) has requested the resource and it has become blocked on it
      - In essence \(P_i\) waits for \(P_j\) to release the resource
Example of Graph Models

![Graph Models Diagram]

Figure 8.1 (a) Resource request and allocation graph (RRAG); (b) Equivalence of RRAG and wait-for graph (WFG) when each resource class contains only one resource unit.

Paths in WFG and RRAG

- A path in a graph is a sequence of edges such that the destination node of an edge is the source node of the subsequent edge.
  - Consider an RRAG path $P_1 - R_1 - P_2 - R_2 - \ldots - P_{n-1} - R_{n-1} - P_n$
  - This path indicates that
    - Process $P_n$ has been allocated a resource unit of $R_{n-1}$
    - Process $P_{n-1}$ has been allocated a resource unit of $R_{n-2}$ and awaits a resource unit of $R_{n-1}$, etc.
  - In WFG, the same path would be $P_1 - P_2 - \ldots - P_{n-1} - P_n$

- A cycle is a path that ends on the same node on which it begins.
  - $P_1 - R_1 - P_2 - R_2 - \ldots - P_{n-1} - R_{n-1} - P_1$ is a cycle in RRAG
  - In WFG, the same cycle would be $P_1 - P_2 - \ldots - P_{n-1} - P_1$
Graph Models (continued)

• A deadlock cannot exist unless an RRAG, or a WFG, contains a cycle

![Diagram of RRAG](image1)

**Figure 8.2** RRAG for the system of Example 8.1.

• A cycle in an RRAG does not necessarily imply a deadlock if a resource class has multiple resource units

![Diagram of RRAG](image2)

**Figure 8.3** RRAG after all requests of Example 8.4 are made.

Matrix Model

• Allocation state represented by two matrices:
  – *Allocated_resources*
  – *Requested_resources*
  – If system has *n* processes and *r* resource classes, each of these two matrices is an *n* × *r* matrix

• Auxiliary matrices: *Total_resources* and *Free_resources*

![Matrix Example](image3)

**Figure 8.3** RRAG after all requests of Example 8.4 are made.