Deadlocks

Readings: Sections 6.1, 6.2

Deadlock

- The permanent blocking of a set of processes that either compete for system resources or communicate with each other
- A set of processes is deadlocked when each process in the set is blocked awaiting an event that can only be triggered by another blocked process in the set
- Permanent because none of the events is ever triggered
- No efficient solution in the general case
Example 2: Memory Request

- Space is available for allocation of 200Kbytes, and the following sequence of events occur:

<table>
<thead>
<tr>
<th>Process</th>
<th>Sequence of Requests</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>Request 80 Kbytes, Request 60 Kbytes; Request 80 Kbytes;</td>
</tr>
<tr>
<td>P2</td>
<td>Request 70 Kbytes, Request 80 Kbytes;</td>
</tr>
</tbody>
</table>

- Deadlock occurs if both processes progress to their second request.

Conditions for Deadlock

- **Mutual Exclusion**: Only one process may use a resource at a time.
- **Hold-and-Wait**: A process may hold allocated resources while awaiting assignment of other resources.
- **No Pre-emption**: No resource can be forcibly removed from a process holding it.
- **Circular Wait**: A closed chain of processes exists, such that each process holds at least one resource needed by the next process in the chain.

Figure 6.2 Example of Deadlock

Space is available for allocation of 200Kbytes, and the following sequence of events occur:

1. P1 requests 80 Kbytes
2. P2 requests 70 Kbytes
3. P1 requests 60 Kbytes
4. P1 requests 80 Kbytes
5. Deadlock occurs
6. P1 requests 80 Kbytes

Figure 6.5 Examples of Resource Allocation Graphs

© 2017 Pearson Education, Inc., Hoboken, NJ. All rights reserved.
Deadlock Prevention

- Design a system in such a way that the possibility of deadlock is excluded

- Two main methods:
  - Indirect
    - Prevent the occurrence of one of the three necessary conditions
  - Direct
    - Prevent the occurrence of a circular wait

Deadlock Avoidance

- Allows the three necessary conditions but makes judicious choices to assure that the deadlock point is never reached

- A decision is made dynamically whether the current resource allocation request will, if granted, potentially lead to a deadlock

  - Requires knowledge of future process requests

Deadlock approaches

Deadlock prevention
- Disallow one of the three necessary conditions for deadlock occurrence
- Or, prevent circular wait condition from happening

Deadlock avoidance
- Do not grant a resource request if this allocation might lead to deadlock
- Or, prevent circular wait condition from happening

Deadlock detection
- Grant resource requests when possible
- And, periodically check for the presence of deadlock and take action to recover

- Allows the three necessary conditions but makes judicious choices to assure that the deadlock point is never reached
Approaches for deadlock avoidance

**Deadlock Avoidance**

**Process Initiation Denial**
- Do not start a process if its demands might lead to deadlock

**Resource Allocation Denial**
- Do not grant an incremental resource request to a process if this allocation might lead to deadlock