**Operating Systems**

Shyam Rajendren

Operating Systems Environments & Administration

1. Describe each of the following in **detail**:
   1. Deadlock

Answer: A deadlock is a system-wide tangle of resource requests that begins

when two or more jobs are put on hold, each waiting for a vital resource to become

available. This occurs if the resources needed by a job are being used by other

jobs, which are also waiting to run but cannot because they’re waiting for other unavailable resources leading to a standstill. When the situation can’t be resolved

by the operating system, intervention is required because the resources which are being tied up can end up hurting the functioning of the entire system as a whole.

* 1. Livelock

Answer: A livelock is a locked system whereby two or more processes continually block the forward progress of the others without making any forward progress themselves. For example, let us take two people who are trying to cross each other. They are both facing each other and when one moves to the right, the other moves to the left and they continuous doing so in synchronization. As a result, none of them are able to pass each other and they are said to be in livelock.

* 1. Starvation

Answer: Starvation is the result of conservative allocation of resources in which a single job is prevented from completing its execution because it is waiting for resources that never become available. For example, if a bunch of people are waiting to cross the road and decide to wait for a gap in the traffic, it is possible that that gap never comes leaving them stranded on one side of the road. This would be an example of starvation.

1. What is the primary difference between Deadlock and Livelock?

Answer: Livelock is similar to deadlock except that processes in livelock are not being blocked or forced to wait. These processes are simply in a continuous state of change preventing them from progressing further.

1. How many **cases** of Deadlock are described in the textbook?

Answer: There are six cases of deadlock explained in the textbook:

1. Deadlocks on File Requests

2. Deadlocks in Databases

3. Deadlocks in Dedicated Device Allocation

4. Deadlock in Multiple Device Allocation

5. Deadlocks in Spooling

6. Livelock while Disk Sharing

1. Describe the four (4) **conditions** of Deadlock or Livelock.

Answer: 1. Mutual Exclusion: A resource cannot be shared, only one process can use it at a time.

2. Hold and Wait: A process is holding a resource while waiting for other resources.

3. No Preemption: A resources cannot be taken from a resource unless that process releases that resource.

4. Circular Wait: Process are waiting for each other in such a way that it leads to a circular waitlist.

1. Answer the following questions about diagramming Deadlocks:  
   1. What symbol is used to represent a resource?

Answer: A square.

* 1. What symbol is used to represent a process?

Answer: A circle.

* 1. What does a dashed line represent?

Answer: A request made by a process for a certain resource.

* 1. What does a solid line represent?

Answer: A resources that has been allocated to a certain process.

1. Describe each of the following Deadlock **strategies** in **detail**:
   1. Prevention

Answer: This is a design strategy used by an operating system where resources are managed in such a way that the conditions necessary for deadlock do not hold. It does this by eliminating one of the four conditions of deadlock (mutual exclusion, hold and wait, no preemption, and circular wait). Mutual exclusion can be bypassed by spooling which allows the output from many jobs to be stored in separate temporary spool files at the same time, and each completed output file is then selected for printing when the device is ready. Resource handling can be bypassed by forcing each job to request, at creation time, every resource it will need to run for completion. No preemption can be bypassed by allowing the operating system to deallocate resources from jobs. Circular wait can be bypassed simply by preventing the operating system from forming a circle of resource requests.

* 1. Avoidance

Answer: A dynamic strategy that attempts to ensure that resources are never allocated in such a way as to place a system in an unsafe state. In 1965, Dijkstra proposed the Banker’s algorithm to help avoid deadlocks. It involves a bank that operates on the following principles:

* No customer will be granted a loan exceeding the bank’s total capital.
* All customers will be given a maximum credit limit when opening an account.
* No customer will be allowed to borrow over the limit.
* The sum of all loans won’t exceed the bank’s total capital.
  1. Detection

Answer: The process of examining the state of an operating system in order to determine whether a deadlock exists. The detection algorithm takes directed resource graphs and reduces them using these three steps:

1. Find a process that is currently using a resource but not waiting for one. This process can be removed from the graph by disconnecting the link tying the

resource to the process. The resource is then returned to the available list. This is possible because the process would eventually finish and return the resource.

1. Find a process that’s waiting only for resources that aren’t fully allocated. This process isn’t contributing to deadlock since it would eventually get the resource it’s waiting for, finish its work, and return the resource to the available list.
2. Go back to Step 1 and continue with Steps 1 and 2 until all lines connecting

resources to processes have been removed.

* 1. Recovery

Answer: The set of steps that must be taken when a deadlock is detected in order to break the circle of waiting processes. There are six recovery methods:

1. Terminate every job that’s active in the system and restart them from the beginning.
2. Terminate only the jobs involved in the deadlock and ask them to be resubmitted.
3. Identify the jobs that are involved in the deadlock and terminate them one at a time, checking to see if the deadlock is eliminated after each removal, until the deadlock has been resolved. Once the system is freed, the remaining jobs are allowed to complete their processing while the halted jobs are started again from the beginning.
4. This method uses a record of the job’s progress which acts like a save point in order to interrupt the execution and prevent it from starting over from the beginning. This method can only be used if the job keeps such a record.
5. A non-deadlocked job is selected and its held resources are preempted, which are then allocated to a deadlocked process so that it can resume execution, thus breaking the deadlock.
6. New jobs are prevented from entering the system which allows the non-deadlocked jobs to run to completion so that they’ll release their resources.
7. Answer the following questions about the Prevention, Avoidance, and Detection Deadlock strategies:
   1. Which is the easiest to implement?

Answer: Prevention.

* 1. Which is the most efficient to use?

Answer: Avoidance.

* 1. Which is the least efficient to use?

Answer: Detection.

|  |  |  |  |
| --- | --- | --- | --- |
| **Job #** | **Devices** **Allocated** | **Maximum** **Required** | **Remaining** **Needs** |
| **1** | **5** | **6** | **1** |
| **2** | **4** | **7** | **3** |
| **3** | **2** | **6** | **4** |
| **4** | **0** | **2** | **2** |
| **Total Devices Allocated:** | | | **11** |
| **Total Devices in the System:** | | | **12** |

1. Refer to the table above to answer the following questions.  
   1. What are the **Remaining Needs** of each of the four (4) jobs?

Answer: Job #1: **1** Job#2: **3** Job#3: **4** Job#4: **2**

* 1. How **many** devices remain **available** for assignment to jobs?

Answer: 1

* 1. Is the system currently in a Safe or Not Safe state?

Answer: Safe state.

* 1. If the system is in a Safe state, which **job** would need to request its remaining devices and run to completion **first**?

Answer: Job # 1.

* 1. If the system is in a Not Safe state, which **job** would be the **best** choice for **termination**?

Answer: System is in a safe state.

|  |  |  |  |
| --- | --- | --- | --- |
| **Job #** | **Devices** **Allocated** | **Maximum** **Required** | **Remaining** **Needs** |
| **1** | **5** | **8** | **3** |
| **2** | **3** | **9** | **6** |
| **3** | **4** | **8** | **4** |
| **Total Devices Allocated:** | | | **12** |
| **Total Devices in the System:** | | | **14** |

1. Refer to the table above to answer the following questions.  
   1. What are the **Remaining Needs** of each of the three (3) jobs?

Answer: Job #1: **3** Job#2: **6** Job#3: **4**

* 1. How **many** devices remain **available** for assignment to jobs?

Answer: 2

* 1. Is the system currently in a Safe or Not Safe state?

Answer: Not safe state.

* 1. If the system is in a Safe state, which **job** would need to request its remaining devices and run to completion **first**?

Answer: System is not in a safe state.

* 1. If the system is in a Not Safe state, which **job** would be the **best** choice for **termination**?

Answer: Job # 2.

1. Answer the following questions about **Starvation**.
   1. Explain what it means for a job to experience Starvation.

Answer: Starvation occurs as a result of conservative allocation of resources in which a single job is prevented from completing its execution because it is waiting for resources that never become available.

* 1. How is starvation similar to or different from **Deadlock** or **Livelock**?

Answer: Starvation and deadlock are similar in that they are both conditions in which processes have been waiting for a requested resource for a long time. However, a deadlock is a condition where no process proceeds for execution and each wait for resources that are being used by other processes. In contrast, starvation is a condition where higher priority processes continuously use resources preventing lower priority processes from using them. Livelock is similar to deadlock, but involves two processes continuously doing the same steps which prevents either of them from executing. However, starvation can occur for a process without another process being cyclically blocked and so no livelock would exist in this case.

* 1. What can an operating system do to prevent a starving job from "remaining in the system forever"?

Answer: An algorithm can be used to find out how long a job had been waiting for resources. If a starving job is detected then the system can block incoming requests from other jobs until the starving job has been satisfied.

1. Using the six (6) Recovery methods discussed on pages 160 and 161, create a table with the following information (assume a **total** of seven (7) jobs are in the system and three (3) of the jobs are **deadlocked**):
   1. Method #
   2. Method Description
   3. Method Type (i.e., Terminate or Preempt)
      * Hint: If the description does not discuss termination, the method is preempting.
   4. # of Victims

Answer:

|  |  |  |  |
| --- | --- | --- | --- |
| **Method #** | **Method Description** | **Method Type** | **# of Victims** |
| 1 | Terminate every job that’s active in the system and restart them from the beginning. | Terminate | 7 |
| 2 | Terminate only the jobs involved in the deadlock and ask them to be resubmitted. | Terminate | 3 |
| 3 | Identify the jobs that are involved in the deadlock and terminate them one at a time, checking to see if the deadlock is eliminated after each removal, until the deadlock has been resolved. Once the system is freed, the remaining jobs are allowed to complete their processing while the halted jobs are started again from the beginning. | Terminate | 1-3 |
| 4 | This method uses a record of the job’s progress which acts like a save point in order to interrupt the execution and prevent it from starting over from the beginning. This method can only be used if the job keeps such a record. | Preempt | 1 |
| 5 | A non-deadlocked job is selected and its held resources are preempted, which are then allocated to a deadlocked process so that it can resume execution, thus breaking the deadlock. | Preempt | 1 |
| 6 | New jobs are prevented from entering the system which allows the non-deadlocked jobs to run to completion so that they’ll release their resources. | Preempt | All new jobs |