1. Detail each of the four conditions which if all are true result in a deadlock.

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 {P0, P1, …, Pn} of waiting processes such that P0 is waiting for a resource that is held by P1, P1 is waiting for a resource that is held by P2, …, Pn–1 is waiting for a resource that is held by Pn, and Pn is waiting for a resource that is held by P0.

1. Define a safe state in relation to deadlocks.

if there exists a sequence <*P1, P2, …, Pn*> of ALL the processes in the systems such that for each Pi, the resources that Pi can still request can be satisfied by currently available resources + resources held by all the *Pj*, with *j* < *I*

1. Describe the two methods for handling deadlocks and how they relate to a system being in a safe state.

**Avoidance Algorithms:** Banker’s algorithm modified resource-allocation graph, modified resource-allocation graph

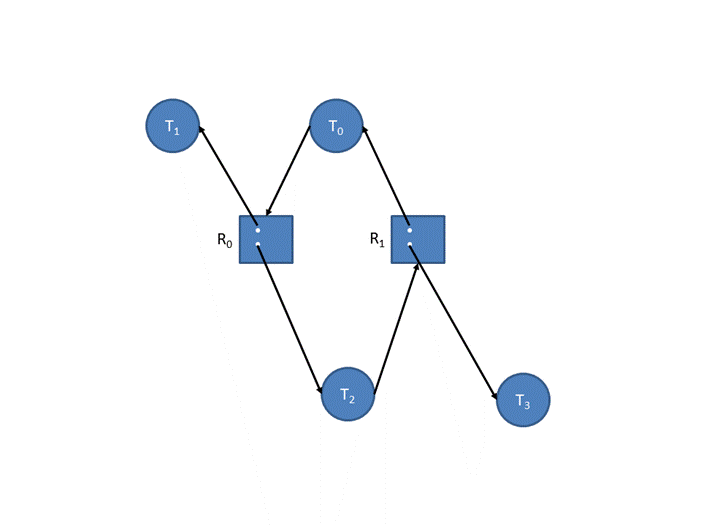
**Deadlock Detection:**  Let deadlock occur, then do preemption to handle it once occurred.

1. Is it possible for a system to exist in an unsafe state and have no deadlocks?

All safe states are deadlock free, but not all unsafe states lead to deadlocks.

1. Using the below resource allocations graph, does this demonstrate a deadlock situation?

NO



1. Using the system snapshot below and using the Banker’s Algorithms for Deadlock Prevention, show the content of the need matrix.

|  |  |
| --- | --- |
|  | ABCD |
| P0 | 2322 |
| P1 | 3013 |
| P2 | 1302 |
| P3 | 1101 |
| P4 | 1002 |

1. Using the same snapshot and using the Banker’s Algorithm for Deadlock Prevention, show that the system is in a safe state by listing the order in which processes can be executed without a deadlock.

<P3, P4, P2, P0, P1> available

|  |
| --- |
| 1 1 1 2 |
| 1 3 1 3 |
| 2 3 2 3 |
| 3 3 2 3 |
| 3 3 3 3 |

1. Using the same snapshot and using the Banker’s Algorithm for Deadlock Prevention, if Process P2 requested (A=0, B=1, C=0, D=1) additional resources, then does this result in a deadlock or could the system grant the request? If the request can be granted, show the updated need matrix, and the sequence of processes. However, if it cannot be granted, demonstrate show the updated need matrix, and processes that would cause the deadlock.

Need 0 1 0 1 <= 1 3 0 2

0 1 0 1 <= 1 1 1 1 granted

|  |  |
| --- | --- |
|  | ABCD |
| P0 | 2322 |
| P1 | 3013 |
| P2 | 1201 |
| P3 | 1101 |
| P4 | 1002 |

available

|  |
| --- |
| 1 0 1 0 |
|  |
|  |
|  |
|  |

<unsafe>

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Processes = {P0-P4} Resources =   {A (2 instances), B (3 instances), C (2 instances), D (2 instances)} | | | | |  |  | | | |  |  | | | |
| Allocation | | | | |  | Max | | | |  | Available | | | |
|  | A | B | C | D |  | A | B | C | D |  | A | B | C | D |
| P0 | 1 | 0 | 0 | 0 |  | 3 | 3 | 2 | 2 |  | 1 | 1 | 1 | 1 |
| P1 | 0 | 0 | 1 | 0 |  | 3 | 0 | 2 | 3 |  |  |  |  |  |
| P2 | 1 | 0+1 | 1 | 0+1 |  | 2 | 3 | 1 | 2 |  |  |  |  |  |
| P3 | 0 | 0 | 0 | 1 |  | 1 | 1 | 0 | 2 |  |  |  |  |  |
| P4 | 0 | 2 | 0 | 1 |  | 1 | 2 | 0 | 3 |  |  |  |  |  |

1. Using the system snapshot below and using he Banker’s Algorithm for Deadlock Detection, determine if a deadlock has been detected. If there is no deadlock, then describe the sequence of processes where they complete without deadlocking.

deadlock

1. Using the same snapshot and using the Banker’s Algorithm for Deadlock Detection, if Process P1 requested (A=0, B=0, C=1, D=0) additional resources, then does this result in a deadlock or could the system grant the request?

Deadlock, not grant

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Processes = {P0-P4} Resources =   {A (2 instances), B (2 instances), C (2 instances), D (2 instances)} | | | | |  |  | | | |  |  | | | |
| Allocation | | | | |  | Request | | | |  | Available | | | |
|  | A | B | C | D |  | A | B | C | D |  | A | B | C | D |
| P0 | 1 | 0 | 0 | 0 |  | 3 | 3 | 2 | 2 |  | 0 | 0 | 0 | 0 |
| P1 | 0 | 0 | 1 | 0 |  | 3 | 0 | 2 | 3 |  |  |  |  |  |
| P2 | 1 | 0 | 1 | 0 |  | 2 | 3 | 1 | 2 |  |  |  |  |  |
| P3 | 0 | 0 | 0 | 1 |  | 1 | 1 | 0 | 2 |  |  |  |  |  |
| P4 | 0 | 2 | 0 | 1 |  | 1 | 2 | 0 | 3 |  |  |  |  |  |