**Tutorial 3: Deadlock**

Q1. (a) Discuss how a deadlock situation can arise if the **FOUR (4)** deadlock conditions hold simultaneously in a multiprogramming environment. Recommend a solution to solve each deadlock condition.

|  |
| --- |
| • **Mutual exclusion**  Solution: Must hold for non-sharable resources; not required for mutually exclusive access. (E.g. Read-only files).  **• Hold and wait**  Solution: Must guarantee that whenever a process requests a resource, it does not hold any other resources. (2 protocols)  **• No pre-emption**  Solution: Could be bypassed by allowing OS to de-allocate resources from jobs.  **• Circular wait**  Solution: Impose a total ordering of all resource types, and require that each process requests resources in an increasing order of enumeration |

(b) Propose THREE (3) possible ways that an operating system can handle the problem of deadlock.

|  |
| --- |
| Strategies for Handling Deadlocks  • Prevent one of the four conditions from occurring.  • Avoid the deadlock if it becomes probable.  • Detect the deadlock when it occurs and recover from it gracefully |

Q2. (a) Consider a system with four processes P1, P2, P3 and P4 and three resources types A, B and C. Resource type A has 2 instances, resource type B has 3 instances and resource type C has 1 instance.

* P1 holds an instance of A and requests an instance of B.
* P2 holds an instance of A and requests an instance of B and C.
* P3 holds an instance of B and requests an instance of A.
* P4 holds an instance of C.

Draw a Directed Resources Graph for the above situation.

(b) Are these four processes mentioned in Q3.(a) in a deadlocked state? Justify your answer.

|  |
| --- |
| (a)    P2  P1  A B C  P3  P4      (b) No.P1 gets the resource B and execute.P2 has enough resources to execute after P4 releases resource C.P3 can get the resource A after P1 releases A and execute. |

Q3. Consider a system consisting of 4 resources of the same type that are shared by 3 processes, each of which needs at most 2 resources. You are required to draw a resource-allocation graph to show that the system is deadlock. Justify your answer

|  |
| --- |
| Its a deadlock due to Circular waits occurs |

Q4. Dijkstra’s algorithm is a deadlock avoidance algorithm and can be used to prevent the occurrence of deadlock. Suppose at time, *t*j, a snapshot of the system is taken and described as shown as Table 1.

|  |  |  |
| --- | --- | --- |
| **Processes** | **Allocated Resources**  **X Y Z** | **Maximum Requirements**  **X Y Z** |
| Pa | 1 1 0 | 5 6 4 |
| Pb | 2 2 3 | 2 4 5 |
| Pc | 4 2 0 | 7 3 1 |
|  |  |  |

Table 1: System snapshot at time, *t*j

1. Calculate the content of the matrix *Need*.

|  |  |  |  |
| --- | --- | --- | --- |
| **Processes** | **Allocated Resources**  **X Y Z** | **Maximum Requirements**  **X Y Z** | **Need Resources**  **X Y Z** |
| Pa | 1 1 0 | 5 6 4 | 4 5 4 |
| Pb | 2 2 3 | 2 4 5 | 0 2 2 |
| Pc | 4 2 0 | 7 3 1 | 3 1 1 |
|  |  |  |  |

1. Assuming that the available resources for X, Y and Z are 3, 1, and 2 respectively, what is the total number of resource instances for resource X, Y and Z?

X – 3+1+2+4 = 10

Y = 1+1+2+2 = 6

Z – 2 +3 = 5

1. Is the system in a safe state? If yes state the process sequence, if no identify the deadlock processes. Justify your answer by showing the progress of resources availability.

|  |  |  |  |
| --- | --- | --- | --- |
| **Processes** | **Allocated Resources**  **X Y Z** | **Need Resources**  **X Y Z** | **Available Resources**  **X Y Z** |
|  |  |  | **3 1 2** |
| Pa | 1 1 0 | 4 5 4 | 10 6 5 |
| Pb | 2 2 3 | 0 2 2 | 9 5 5 |
| Pc | 4 2 0 | 3 1 1 | 7 3 2 |
|  |  |  |  |

This is a safe state, Sequence is Pc > Pb > Pa

Q5. Table below describe the snapshot of the system taken at particular time. Using banker’s algorithm, determine if the system is in a safe state. (Note: Assume that the available resources for A, B and C are 4, 3, 2 respectively)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Processes** | **Allocation** | | | **Maximum** | | |
| **A** | **B** | **C** | **A** | **B** | **C** |
| P1 | 0 | 1 | 1 | 6 | 6 | 1 |
| P2 | 3 | 1 | 1 | 4 | 2 | 2 |
| P3 | 3 | 0 | 1 | 5 | 3 | 3 |
| P4 | 0 | 0 | 1 | 1 | 1 | 4 |
| P5 | 2 | 1 | 1 | 5 | 1 | 5 |
| P6 | 2 | 2 | 2 | 3 | 4 | 2 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Compute the need matrix (Max – Allocation) and apply the Dijkstra’s algorithm.   |  |  |  |  | | --- | --- | --- | --- | | Need | | | | |  | A | B | C | | P1 | 6 | 5 | 0 | | P2 | 1 | 1 | 1 | | P3 | 2 | 3 | 2 | | P4 | 1 | 1 | 3 | | P5 | 3 | 0 | 4 | | P6 | 1 | 2 | 0 |   The system is in a safe state as long as it follows the process progression of P2 🡪 P4 🡪 P5 🡪 P6 🡪 P3 🡪P1 |

Q6. In some operating systems, process scheduling may be performed at 3 different levels, namely short-term, medium-term, and long-term scheduling.

For short-term scheduling, some commonly used CPU scheduling policies are First-Come-First-Served (FCFS), Shortest-Remaining-Time-First (SRTF), Round-Robin and Preemptive Priority –based scheduling.

1. Which of the policies cause starvation of processes? Explain your answer.

|  |
| --- |
| •Shortest-Remaining-Time-First (SRTF) and Preemptive Priority –based scheduling policies will cause starvation of processes  •Starvation avoidance: Process aging involves temporarily increases the priority of a low-priority process that has not been run in a while to ensure that it gets a chance to get scheduled to run. |

1. Differentiate between deadlock and starvation. Which situation is more serious?

|  |
| --- |
| Deadlock is set of blocked processes which each of them holding a resource and keeps waiting for another resource that held by another process.  Starvation is a set of processes are waiting too long in the queue for execution.  Deadlock is more serious than starvation. |

**Self-Review**

Q1. There are four processes (A, B, C, and D) and five resources (R1, R2, R3, R4 and R5). Assume that all the resources are non-sharable and the numbers of instances for each resource are 2, 2, 1, 2, 1 respectively.

Process A holds one instance of R1 and R2. It requests one instance of R4.

Process B holds one instance of R2 and R3. It requests one instance of R1.

Process C holds one instance of R4. It requests another one instance of R5.

Process D holds one instance of R4 and R5. It requests one instance of R3.

1. Draw a directed resources allocation graph for the above scenario.
2. Are these four processes in a deadlock state? Justify your answer.

**R1 R2 R3 R4 R5**

**A**

**B**

**C**

**D**

* No deadlock.
* Process B can finish its process first.
* Then, it releases the instances of R3 to Process D.
* When Process D is completed. Process C can finish the process after getting the instance of R5.
* Finally, Process A can complete its process.

Q2. Assuming that there are 5 processes (P1, P2, P3, P4 and P5) which are processed by a CPU. While processing, the processes need 3 resources (A, B, C). The **Table 1** below has shown the resources allocated to each current process, maximum resources the processes require to complete their tasks and the available resources respectively.

Use ***banker’s algorithm*** to determine whether the system is in a safe state or unsafe state when the process sequence is P1→ P5. If the system is in an unsafe state, propose a process sequence where the system is in a safe state. Justify your answers.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | *A* | *B* | *C* |  |  | *A* | *B* | *C* |  | *A* | *B* | *C* |
| *P1* | *0* | *1* | *2* |  | *P1* | *0* | *3* | *4* |  | *2* | *2* | *2* |
| *P2* | *2* | *0* | *0* |  | *P2* | *3* | *3* | *4* |  |  |  |  |
| *P3* | *3* | *2* | *1* |  | *P3* | *5* | *6* | *2* |  |  |  |  |
| *P4* | *1* | *2* | *3* |  | *P4* | *2* | *3* | *4* |  |  |  |  |
| *P5* | *0* | *4* | *2* |  | *P5* | *1* | *6* | *6* |  |  |  |  |
| *Allocated resources* | | | |  | *Max resources required* | | | |  | *Available resources* | | |

Table 1: List of processes and resources

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Process | Allocated  A B C | Max  A B C | Available  A B C | Need  A B C | Available >= Need |
| P1 | 0 1 2 | 0 3 4 | 2 2 2 | 0 2 2 | OK |
| P2 | 2 0 0 | 3 3 4 | 2 3 4 | 1 3 4 | OK |
| P3 | 3 2 1 | 5 6 2 | 4 3 4 | 2 4 1 | B resources are not sufficient |
| P4 | 1 2 3 | 2 3 4 |  |  |  |
| P5 | 0 4 2 | 1 6 6 |

The system is in unsafe state if the sequence is P1, P2, P3, P4 and P5.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Process | Allocated  A B C | Max  A B C | Available  A B C | Need  A B C | Available >= Need |
| P1 | 0 1 2 | 0 3 4 | 2 2 2 | 0 2 2 | OK |
| P2 | 2 0 0 | 3 3 4 | 2 3 4 | 1 3 4 | OK |
| P4 | 1 2 3 | 2 3 4 | 4 3 4 | 1 1 1 | OK |
| P5 | 0 4 2 | 1 6 6 | 5 5 7 | 1 2 4 | OK |
| P3 | 3 2 1 | 5 6 2 | 5 9 9 | 2 4 1 | OK |
|  |  |  | 8 11 10 |  |  |

The system is in safe state if the sequence is P1, P2, P4, P5 and P3.

***(Note: Accept other answers if they are correct)***