CHANDIGARH UNIVERSITY

Discover. Learn. Empower.

UNIVERSITY INSTITUTE OF ENGINEERING

Bachelor of Engineering (Computer Science & Engineering)

Operating System (CST-328)

Subject Coordinator: Er. Puneet kaur(E6913)

Deadlocks

DISCOVER . LEARN . EMPOWER

Lecture 11
Deadlocks
List-of-content

Deadlocks: Deadlock avoidance-safe state, Banker's algorithms-Safety algorithm

Deadlock Avoidance

- This strategy involves maintaining a set of data using which a decision is made whether to entertain the new request or not.
- If entertaining the new request causes the system to move in an unsafe state, then it is discarded.
- This strategy requires that every process declares its maximum requirement of each resource type in the beginning.
- The main challenge with this approach is predicting the requirement of the processes before execution.
- Banker's Algorithm is an example of a deadlock avoidance strategy.

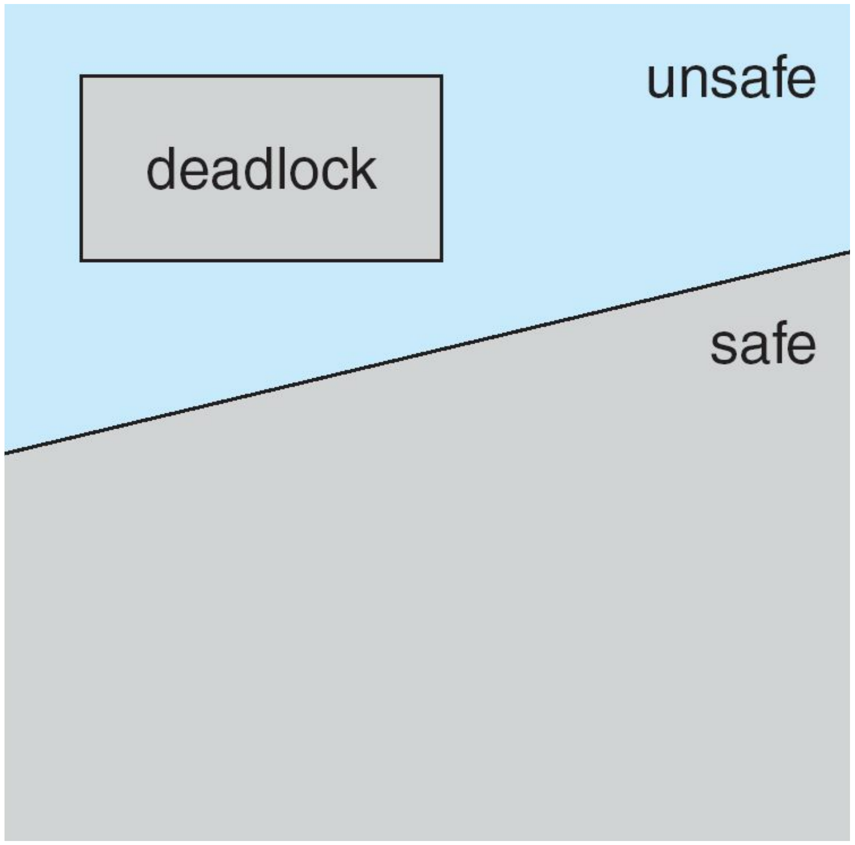
Safe State

- When a process requests an available resource, system must decide if immediate allocation leaves the system in a safe state
- System is in **safe state** if there exists a sequence $<P_1, P_2, \ldots, P_n>$ of ALL the processes in the systems such that for each P_i, the resources that P_i can still request can be satisfied by currently available resources + resources held by all the P_j, with $j < I$
- That is:
  - If P_i resource needs are not immediately available, then P_i can wait until all P_j have finished
  - When P_j is finished, P_i can obtain needed resources, execute, return allocated resources, and terminate
  - When P_i terminates, P_{i+1} can obtain its needed resources, and so on

Basic Facts

- If a system is in safe state ⇒ no deadlocks
- If a system is in unsafe state ⇒ possibility of deadlock
- Avoidance ⇒ ensure that a system will never enter an unsafe state.

Safe, Unsafe, Deadlock State

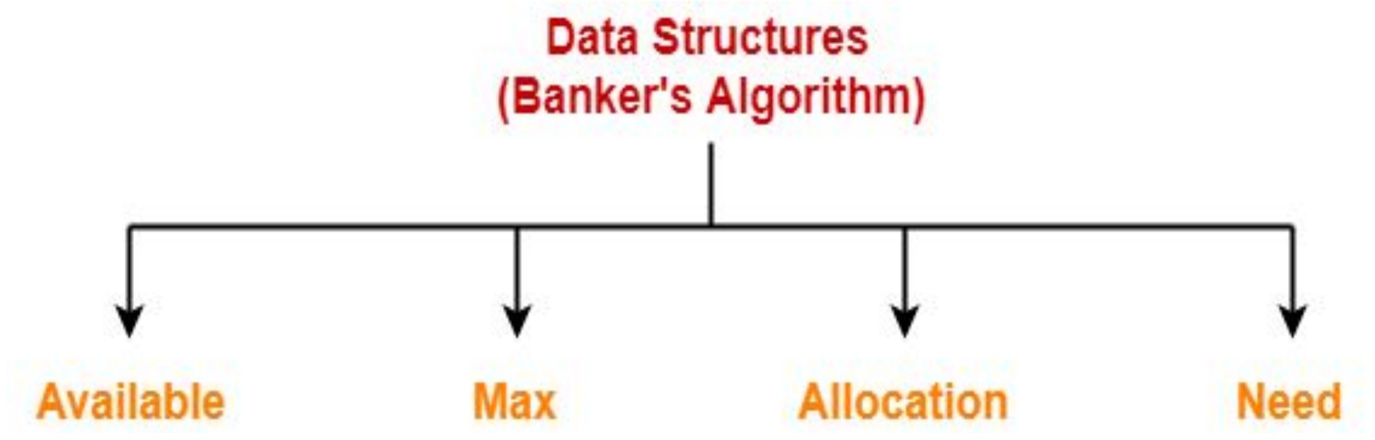


Deadlock Avoidance Algorithms

- Single instance of a resource type
  - Use a resource-allocation graph
- Multiple instances of a resource type
  - Use the banker's algorithm

Banker's Algorithm

- Banker's Algorithm is a deadlock avoidance strategy.
- It is called so because it is used in banking systems to decide whether a loan can be granted or not.
- Banker's Algorithm requires whenever a new process is created, it specifies the maximum number of instances of each resource type that it exactly needs.
- To implement banker's algorithm, following four data structures are used-



Data Structures for the Banker's Algorithm

Let n = number of processes, and m = number of resources types.

- Available: Vector of length m. If available [j] = k, there are k instances of resource type Rj available.
- Max: n x m matrix. If Max [i,j] = k, then process Pi may request at most k instances of resource type Rj.
- Allocation: n x m matrix. If Allocation[i,j] = k then Pi is currently allocated k instances of Rj.
- Need: n x m matrix. If Need[i,j] = k, then Pi may need k more instances of Rj to complete its task.

- Need [i,j] = Max[i,j] – Allocation [i,j].

Working

Step-01:

- Banker's Algorithm checks whether the request made by the process is valid or not.
- A request is considered valid if and only if-
- The number of requested instances of each resource type is less than the need declared by the process in the beginning.
- If the request is invalid, it aborts the request.
- If the request is valid, it follows step-02.

Step-02:

- Banker's Algorithm checks if the number of requested instances of each resource type is less than the number of available instances of each type.
- If the sufficient number of instances are not available, it asks the process to wait longer.
- If the sufficient number of instances are available, it follows step-03.

Working

Step-03:

- Banker's Algorithm makes an assumption that the requested resources have been allocated to the process.
- Then, it modifies its data structures accordingly and moves from one state to the other state.
  - Available = Available - Request(i)
  - Allocation(i) = Allocation(i) + Request(i)
  - Need(i) = Need(i) - Request(i)
- Now, Banker's Algorithm follows the safety algorithm to check whether the resulting state it has entered in is a safe state or not.
- If it is a safe state, then it allocates the requested resources to the process in actual.
- If it is an unsafe state, then it rollbacks to its previous state and asks the process to wait longer.

Safety Algorithm

Safety Algorithm is executed to check whether the resultant state after allocating the resources is safe or not.

1. When a process gets all its resources it must return them in a finite amount of time. Let *Work* and *Finish* be vectors of length *m* and *n*, respectively. Initialize:

    a) *Work* = *Available*

    b) *Finish* [*i*] = *false* for *i* - 1,3, …, *n.*

2. Find an *i* such that both:

    (a) *Finish* [*i*] = *false*

    (b) $Need_i$ <= *Work*

- If no such *i* exists, go to step 4.

Safety Algorithm

3. $Work = Work + Allocation_i$

   $Finish[i] = true$

   go to step 2.

4. If $Finish\ [i]$ == true for all i, then the system is in a safe state.

Example of Banker's Algorithm

- 5 processes P_0 through P_4; 3 resource types *A*(10 instances), *B* (5instances, and *C* (7 instances).
- Snapshot at time T_0:

	Allocation	*Max*	*Available*
	A B C	*A B C*	*A B C*
P_0	0 1 0	7 5 3	3 3 2
P_1	2 0 0	3 2 2	
P_2	3 0 2	9 0 2	
P_3	2 1 1	2 2 2	
P_4	0 0 2	4 3 3	

Example (Cont.)

- The content of the matrix. Need is defined to be Max – Allocation.

Need

$$\begin{array}{cc} & A\ B\ C \\ P_0 & 7\ 4\ 3 \\ P_1 & 1\ 2\ 2 \\ P_2 & 6\ 0\ 0 \\ P_3 & 0\ 1\ 1 \\ P_4 & 4\ 3\ 1 \end{array}$$

The system is in a safe state since the sequence < P1, P3, P4, P2, P0> satisfies safety criteria.

Example P1 Request (1,0,2) (Cont.)

- Check that Request ≤ Available (that is, (1,0,2) ≤ (3,3,2) ⇒ true.

	Allocation	*Need*	*Available*
	A B C	*A B C*	*A B C*
P_0	0 1 0	7 4 3	2 3 0
P_1	3 0 2	0 2 0	
P_2	3 0 1	6 0 0	
P_3	2 1 1	0 1 1	
P_4	0 0 2	4 3 1	

Example P1 Request (1,0,2) (Cont.)

- Executing safety algorithm shows that sequence <P1, P3, P4, P0, P2> satisfies safety requirement.
- Can request for (3,3,0) by P4 be granted?
- Can request for (0,2,0) by P0 be granted?

Example P1 Request (1,0,2) (Cont.)

- Executing safety algorithm shows that sequence <P1, P3, P4, P0, P2> satisfies safety requirement.
- Can request for (3,3,0) by P4 be granted?
- Can request for (0,2,0) by P0 be granted?

Problem-01

A single processor system has three resource types X, Y and Z, which are shared by three processes. There are 5 units of each resource type. Consider the following scenario, where the column alloc denotes the number of units of each resource type allocated to each process, and the column request denotes the number of units of each resource type requested by a process in order to complete execution. Which of these processes will finish LAST?

- Option 1. P0
- Option 2. P1
- Option 3. P2
- Option 4. None of the above since the system is in a deadlock

Problem-01 cont…

	Alloc			Request		
	X	Y	Z	X	Y	Z
P0	1	2	1	1	0	3
P1	2	0	1	0	1	2
P2	2	2	1	1	2	0

Solution

- According to question-
- Total = [X Y Z] = [5 5 5]
- Total _Alloc = [X Y Z] = [5 4 3]
-
- Now,
- Available
- = Total – Total_Alloc
- = [5 5 5] – [5 4 3]
- = [0 1 2]

Solution

- Step-01:

- With the instances available currently, only the requirement of the process P1 can be satisfied.
- So, process P1 is allocated the requested resources.
- It completes its execution and then free up the instances of resources held by it.

Then,

Available

= [0 1 2] + [2 0 1]

= [2 1 3]

Solution

Step-02:

- With the instances available currently, only the requirement of the process P0 can be satisfied.
- So, process P0 is allocated the requested resources.
- It completes its execution and then free up the instances of resources held by it.

Then-

Available

= [2 1 3] + [1 2 1]

= [3 3 4]

Solution

<u>Step-03</u>:

- With the instances available currently, the requirement of the process P2 can be satisfied.
- So, process P2 is allocated the requested resources.
- It completes its execution and then free up the instances of resources held by it.

Then Available = [3 3 4] + [2 2 1]= [5 5 5]

Thus,

- There exists a safe sequence P1, P0, P2 in which all the processes can be executed.
- So, the system is in a safe state.
- Process P2 will be executed at last.

Thus, Option (3) is correct.

Problem-02

- An operating system uses the banker's algorithm for deadlock avoidance when managing the allocation of three resource types X, Y and Z to three processes P0, P1 and P2. The table given below presents the current system state. Here, the Allocation matrix shows the current number of resources of each type allocated to each process and the Max matrix shows the maximum number of resources of each type required by each process during its execution.

	Allocation			Max		
	X	Y	Z	X	Y	Z
P0	0	0	1	8	4	3
P1	3	2	0	6	2	0
P2	2	1	1	3	3	3

Problem-02 cont…

There are 3 units of type X, 2 units of type Y and 2 units of type Z still available. The system is currently in safe state. Consider the following independent requests for additional resources in the current state-

REQ1: P0 requests 0 units of X, 0 units of Y and 2 units of Z
REQ2: P1 requests 2 units of X, 0 units of Y and 0 units of Z

Which of the following is TRUE?

1. Only REQ1 can be permitted
2. Only REQ2 can be permitted
3. Both REQ1 and REQ2 can be permitted
4. Neither REQ1 nor REQ2 can be permitted

Solution

According to question,

- Available = [X Y Z] = [3 2 2]

Now,

- Need = Max – Allocation

So, we have-

	Allocation			Max			Need		
	X	Y	Z	X	Y	Z	X	Y	Z
P0	0	0	1	8	4	3	8	4	2
P1	3	2	0	6	2	0	3	0	0
P2	2	1	1	3	3	3	1	2	2

Solution

Checking Whether REQ1 Can Be Entertained-

- Need of P0 = [0 0 2]
- Available = [3 2 2]

Clearly,

- With the instances available currently, the requirement of REQ1 can be satisfied.
- So, banker's algorithm assumes that the request REQ1 is entertained.
- It then modifies its data structures as-

Solution

	Allocation			Max			Need		
	X	Y	Z	X	Y	Z	X	Y	Z
P0	0	0	3	8	4	3	8	4	0
P1	3	2	0	6	2	0	3	0	0
P2	2	1	1	3	3	3	1	2	2

Solution

Available= [3 2 2] – [0 0 2]= [3 2 0]

- Now, it follows the safety algorithm to check whether this resulting state is a safe state or not.
- If it is a safe state, then REQ1 can be permitted otherwise not.

Step-01:

- With the instances available currently, only the requirement of the process P1 can be satisfied.
- So, process P1 is allocated the requested resources.
- It completes its execution and then free up the instances of resources held by it.

Then Available= [3 2 0] + [3 2 0]= [6 4 0]

Now,

- It is not possible to entertain any process.
- The system has entered the deadlock state which is an unsafe state.
- Thus, REQ1 will not be permitted.

Solution

Checking Whether REQ2 Can Be Entertained-

- Need of P1 = [2 0 0]
- Available = [3 2 2]

Clearly,

- With the instances available currently, the requirement of REQ1 can be satisfied.
- So, banker's algorithm assumes the request REQ2 is entertained.
- It then modifies its data structures as-

Solution

	Allocation			Max			Need		
	X	Y	Z	X	Y	Z	X	Y	Z
P0	0	0	1	8	4	3	8	4	2
P1	5	2	0	6	2	0	1	0	0
P2	2	1	1	3	3	3	1	2	2

Solution

Available

= [3 2 2] – [2 0 0] = [1 2 2]

Now, it follows the safety algorithm to check whether this resulting state is a safe state or not.

If it is a safe state, then REQ2 can be permitted otherwise not.

Step-01:

With the instances available currently, only the requirement of the process P1 can be satisfied.

So, process P1 is allocated the requested resources.

It completes its execution and then free up the instances of resources held by it.

Then-

Available

= [1 2 2] + [5 2 0]

= [6 4 2]

Solution

Step-02:

- With the instances available currently, only the requirement of the process P2 can be satisfied.
- So, process P2 is allocated the requested resources.
- It completes its execution and then free up the instances of resources held by it.

Then-
Available
= [6 4 2] + [2 1 1]
= [8 5 3]

Solution

Step-03:

With the instances available currently, the requirement of the process P0 can be satisfied.

- So, process P0 is allocated the requested resources.
- It completes its execution and then free up the instances of resources held by it.

Then-

Available = [8 5 3] + [0 0 1]= [8 5 4]

Thus,

- There exists a safe sequence P1, P2, P0 in which all the processes can be executed.
- So, the system is in a safe state.
- Thus, REQ2 can be permitted.

Thus, Correct Option is (B).

Conclusion

This lecture enables you to have a deep insight to Deadlock avoidance, a safe state, Banker's algorithms and Safety algorithm to avoid deadlock.

References

https://www.includehelp.com/c-programming-questions/

https://www.studytonight.com/operating-system/

https://computing.llnl.gov/tutorials/

https://www.tutorialspoint.com/operating_system/index.htm#:~:text=An%20operating%20system%20(OS)%20is,software%20in%20a%20computer%20system.

https://www.javatpoint.com/os-tutorial

https://www.guru99.com/operating-system-tutorial.html
https://www.geeksforgeeks.org/operating-systems/