Deadlock avoidance

-What is deadlock? (Direct answer)

Deadlock happens when two or more processors re waiting for some event to happen, but ta event doesn't happen.

Deadlocks are a set of blocked processes each holding a resource and waiting to acquire a resource held by another process.

Process P1 is waiting for P2 and P2 is waiting for P1.

Necessary Deadlock Conditions:

1-Mutual exclusion:

If the resource is non-shareable then the probability of getting deadlock will increase.

If recourse is shareable then the occurrence of deadlock decrease.

2-No preemption:

A resource cannot be preempted from a process by force.

A process will not release any recourse until completion.

3-Hold and wait:

Means the process is holding one of the resources and waiting for another.

4-Circular wait:

Waiting processor make a circular of waiting:

P1 ⟶ P2 , P2 ⟶ P3 , P3 ⟶ P1

- What is the detailed algorithm of the Banker's Algorithm? (Description)

Deadlock avoidance:

We can try to avoid deadlocks by making use prior knowledge about the usage of resources by processes, including the max number of resources will be needed by this process and the sequence of requests and releases.

Ensure the system will never enter Unsafe state.The deadlock-avoidance algorithm dynamically examines the resource-allocation state to ensure that there can never be a circular-wait condition

we can define avoidance algorithms that ensure that the system will never deadlock. The idea is simply to ensure that the system will always remain in a safe state. Initially, the system is in a safe state. Whenever a process requests a resource that is currently available, the system must decide whether the resource can be allocated immediately or whether the process must wait. The request is granted only if the allocation leaves the system in a safe state.

-Safety Algorithm

(شرحي)

First, we need to specify the maximum need of all processes and the allocated resources.

Second, for each process calculate the needed resources by subtracting, (Max-Allocated) for each process.

Third, for all Processes we check if the needed number of resources is Currently available before assigning the resources.

If the condition satisfied (Need<= Available(work)) the process is given the resources.

Before checking for the next processes we add the previous Allocated resources to the available (When a process finish executing it release all its resources).

If the condition not satisfied we move to the next process.

If we can find a path (sequence) that guarantee that each process will get the number of process it needs at certain of time, then it's a safe sequence (state).

Therefore, the deadlock is avoided.

If not, the system is in unsafe state and deadlock may occur.

(شرح الكتاب)

We can now present the algorithm for finding out whether or not a system is in a safe state. This algorithm can be described as follows:

1. **Let *Work* and *Finish* be vectors of length *m* and *n*, respectively. Initialize *Work* = *Available* and *Finish*[*i*] = *false* for *i* = 0, 1, ..., *n* − 1.**
2. **Find an index *i* such that both**

a. ***Finish***[*i*] == ***false***

b. $Need_i \leq Work$

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

3. ***Work* = *Work* + *Allocation*¡ *Finish*[*i*] = *true***
   Go to step 2.
4. **If *Finish*[*i*] == *true* for all *i*, then the system is in a safe state.**

- What is a safe/unsafe condition? (Description)

Safe state:

If there's a safe sequence of processes execution for all process, all process finishes successfully.

A state is *safe* if the system can allocate resources to each process (up to its maximum) in some order and still avoid a deadlock.

a system is in a safe state only if there exists a **safe sequence**.

Safe state ➡ No Deadlock.

Unsafe state:

If the sequence of processes execution can cause Deadlock.

Unsafe state ➡ Deadlock may occur.

- What are the major four deadlock research areas? (Direct answer)

1-Deadlock **prevention**.
2-Deadlock **avoidance**.
3-Deadlock **detection**.
4-Deadlock **recovery**.

- What is the difference between each of the four deadlock areas? (Direct answer)

1-Deadlock prevention:

By ensuring that at least one of these conditions cannot hold, we can ***prevent*** the occurrence of a deadlock.

2-Deadlock avoidance:

Is to require additional information about how resources are to be requested. Given this a priori information, it is possible to construct an algorithm that ensures that the system will never enter a deadlocked state.

3-Deadlock detection:

If a system does not employ either a deadlock-prevention or a deadlock-avoidance algorithm, then a system may provide:

• An algorithm that examines the state of the system to determine whether a deadlock has occurred.

• An algorithm to recover from the deadlock.

1. If resources have single instance:

In this case for Deadlock detection we can run an algorithm to check for cycle in the Resource Allocation Graph. Presence of cycle in the graph is the sufficient condition for deadlock.

2. If there are multiple instances of resources:
Detection of cycle is necessary but not sufficient condition for deadlock detection, in this case system may or may not be in deadlock varies according to different situations.

4-Deadlock recovery:
When a detection algorithm determines that a deadlock exists, several alternatives are available.

One possibility is to inform the operator that a deadlock has occurred and to let the operator deal with the deadlock manually.
Another possibility is to let the system recover from the deadlock automatically.

There are two options for breaking a deadlock:
One is simply to **abort** one or more processes to break the circular wait.
The other is to **preempt** some resources from one or more of the deadlocked processes.

- What are the cases that may lead to an unsafe condition? (Description)

If a safe sequence does not exist, then the system is in an unsafe state, which **may** lead to deadlock.
(All safe states are deadlock free, but not all unsafe states lead to deadlocks).