Sushant Bhattarai
074|BCT|045
Assignment #4 - Deadlock

Page No.
Date: 12/12/2020

1. How can deadlock be prevented. Explain.

Ans: Deadlock is a situation where a set of processes are blocked because each process is holding a resource and waiting for another ~~process~~ resource acquired by some other processes.

Deadlock can be prevented by eliminating any of the following four conditions:-

i) Eliminate mutual exclusion

Mutual exclusion is the condition in which at least one resource must be held in a non-sharable mode. If any other process requests this resource, that process must wait for the resource to be released.

- Shared resources such as read-only files do not lead to deadlocks
- Unfortunately, some resources, such as printers and tape drives, require exclusive access by a single process.

ii) Eliminate hold and wait

- Allocate all required resources to the process before the start of its execution, this way hold and wait condition is eliminated but it will lead to low device utilization. For example, if a process requires printer at a later time and we have allocated printer before the start of its execution printer will remain blocked till it has completed its execution.
- The process will make a new request for resources after releasing the current set of resources. This solution may lead to starvation.

iii) Eliminate No preeption:

- Deadlock arises due to the fact that a process can't be stopped once it starts. However, if we take the resource away from the process which is causing deadlock then we can prevent deadlock.
- This is not a good approach at all since if we take a resource away which is being used by the process then all the work which it has done till now can become inconsistent.
- Consider a printer is being used by any process. If we take the printer away from that process and assign it to some other process then all the data which has been printed can become inconsistent and ineffective and also the fact that the process can't start printing again from where it has left which causes performance inefficiency.

iv) Eliminate circular wait:

To violate circular wait, we can assign a priority number to each of the resource. A process can't request for a lesser priority resource. This ensures that not a single process can request a resource which is being utilized by some other process & no cycle will be formed.

Summary:

Condition	Approach	Practically possible?
1. Mutual exclusion	Spooling	X
2. Hold and wait	Request for all resources initially	X
3. No preemption	Snatch all resources	X
4. Circular wait	Assign priority to each resource & order them numerically	✓

2. Short notes:

i) Integrated deadlock strategy:

Integrated deadlock strategy is the process of combining or grouping all the deadlock handling mechanisms that include deadlock prevention, deadlock detection and deadlock avoidance in an integrated form. The following points summarize integrated deadlock strategy:

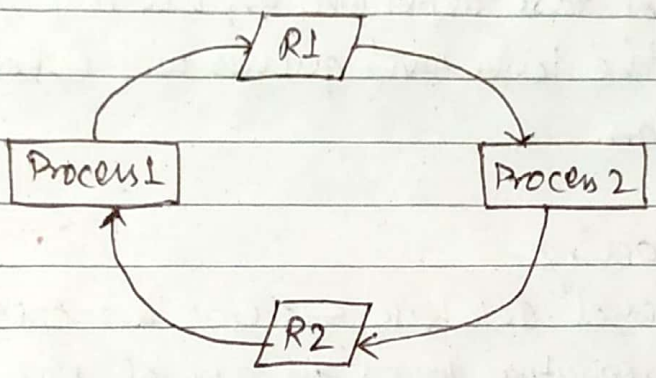
- Rather than attempting to design an operating system that employs only one of the above deadlock handling mechanisms, it might be more efficient to use different strategies in different situations:
  i) Group resources into a number of different resource classes
  ii) Use the linear ordering strategy defined previously for the prevention of circular wait to prevent deadlocks between resource classes.
  iii) Within a resource class, use the algorithm that is the most appropriate for that class.

For example:

i) Swappable space: Blocking of memory on secondary storage for use in swapping process.
ii) Process resources: Assignable devices, such as tape drives & file
iii) Main memory: Assignable in pages or segments to processes
iv) Internal resources: such as Input/Output channels.

ii) Livelock

Livelock occurs when two or more processes continually repeat the same interaction in response to changes in the other processes without doing any useful work. These processes are not in the waiting state, and they are running concurrently. This is different from a deadlock because in a deadlock, all processes are in the waiting state.



Process P1 holds resource R2 and requires R1 while Process P2 holds resource R1 and requires R2

livelock occurs when the total number of allowed processes in a specific system should be defined by the total number of entries in the process table. Therefore, process table slots should be referred to as finite resources.

iii) Starvation and ageing

Starvation is the situation where all the low priority processes get blocked, and the high priority processes proceed. In any system, requests for high/low priority ~~proc~~ resources keep on ~~hang~~ happening dynamically. Thereby, some policy is required to decide who gets the support when.

Using some algorithms, some processes may not get the desired serviced even though they are not deadlocked. Starvation occurs when some threads make shared resources unavailable for a long period of time

Ageing is a condition which is used to reduce starvation of low priority tasks. It is a process which gradually increases the priority of task depending on waiting time. It ensures that jobs in the lower level queues will eventually complete their execution.

iv) Two phase locking

In several database systems, an operation that occurs frequently is requesting locks on several records and then updating all the locked records. There is a real danger of deadlock whenever multiple processes are running at the same instance of time. The approach used is called the two-phase locking.

In first phase, the process tries to lock all the records that it needs, one at a time. And if it succeeds, then it begins the second phase, performing its updates and releasing the locks. No any real work is done in first phase.

If during the first phase, some record is needed that is already locked, the process just releases all its locks and starts the first phase all over.

v) Communication deadlock:

Communication deadlock is a type of a distributed deadlock.

If each process in the set is waiting to communicate with another process ~~at the s~~ in same set and no process ever initiates any communication until it receives communication for which it is waiting.

Example:

A communication deadlock occurs when process A is trying to send a message to process B, which is trying to send a message to process C which is trying to send a message to process A.