Chapter - 5

Deadlock

Introduction to Deadlocks →

→ In the multiprogramming environment, processes compete for physical as well as logical resources. A process requests resources, if they are not available at that time, a process enters into the wait state. It may happen that waiting processes will never change the state again because resources requested by the process are occupied by some other process. This is known as deadlock.

Example → Let's assume two processes Pa & Pb. Pa hold tape drive and Pb hold printer. Deadlock can occur if Pa request for printer and Pb request for tape drive.

Necessary Condition of DeadLock Or DeadLock Detection

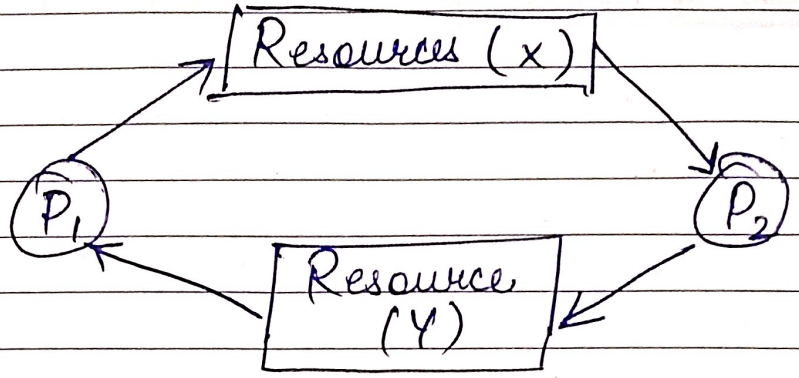
- Mutual Exclusion
- Hold & Wait
- No Preemption
- Circular Wait

(1) Mutual Exclusion → In Mutual Exclusion one process can utilize one resource at the same time. Whenever another process make the request for the same resources at the same time, the requesting process needs to be delayed till the resource is released by previous process using it.

(2) Hold & Wait → There must be at least one resource that is holding by a process and is waiting for other resources that are currently held by other processes.

(3) No Preemption → Resources granted to the requesting process cannot be taken away by compelling (forcefully) it. These resources should be released by the process itself that is holding it.

(4) Circular Wait → There should be a collection or set of waiting process like ($P_0, P_1, P_2 ---- P_n$). In this set process P_0 is waiting for some resource that is being utilized by P_1. Also P_1 is waiting for the resource that is being used by P_2 and so on.



DeadLock Prevention → We may prevent deadlock if one of these above condition do not hold.

(1) Elimination of mutual exclusion :

DeadLock can be prevented by denying ME.

(2) Elimination of hold & wait Condition :

To ensure the removal of hold & wait condition, we should ensure that whenever process request ~~req~~ resources, it should not hold any other resources.

(3) Removal of "No Preemption" condition :

Resources should not be preempted that are already allocated.

④ Elimination of Circular Wait : In case of elimination of circular wait, it is necessary to order the resources in such way that circular waiting will never occur.

Deadlock Avoidance → the technique to ensure that deadlock should not occur is deadlock avoidance technique.

→ The deadlock avoidance algorithm check the state of the resource allocation. The state should be safe or unsafe

Safe → A state can be called as safe if system is capable of providing the resources to each and every process up to its maximum value or as per its requirement.

Unsafe → A state can be called as unsafe if system is not capable of providing / allocating the different resources required by the processes.

Banker's Algorithm → As its name suggest, it work like the banking system. The cash resource of bank is allocated to customer in a particular manner.

Following data structure could be used:—

1. **Available** → it is a vector of length m. This indicates the available resources of each and every type.
2. **Max** → this indicates the matrix n x m. that define the maximum demand.
3. **Allocation** → The n x m matrix defines that the no. of resources of each type allocated to each process at the given moment.
4. **Need** → the requirement or need of remaining resources can be indicated by n x m matrix.