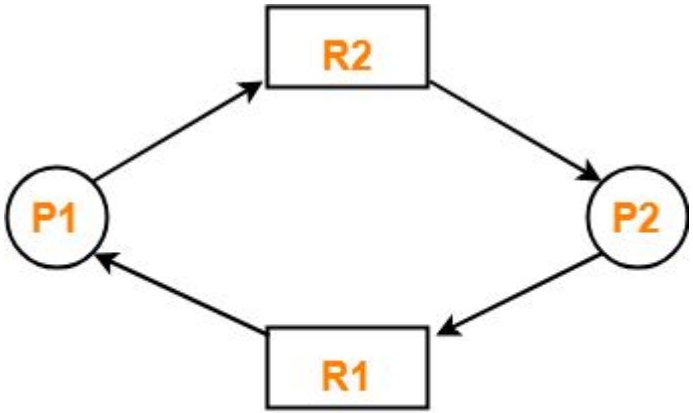
Deadlock in OS-

- The execution of two or more processes is blocked because each process holds some resource and waits for another resource held by some other process.



Example of a deadlock

Conditions For Deadlock-

There are following 4 necessary conditions for the occurrence of deadlock-

1. Mutual Exclusion
2. Hold and Wait
3. No preemption
4. Circular wait

Mutual Exclusion-

By this condition,

- There must exist at least one resource in the system which can be used by only one process at a time.
- If there exists no such resource, then deadlock will never occur.
- Printer is an example of a resource that can be used by only one process at a time.

Hold and Wait-

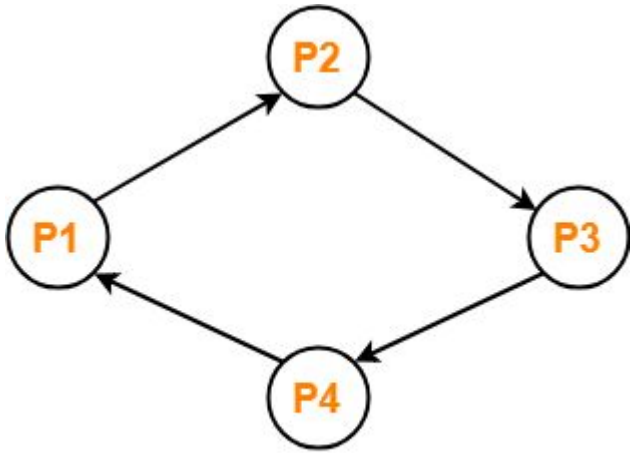
- There must exist a process which holds some resource and waits for another resource held by some other process.

No Preemption-

- Once the resource has been allocated to the process, it can not be preempted.
- It means resource can not be snatched forcefully from one process and given to the other process.
- The process must release the resource voluntarily by itself.

Circular Wait-

- All the processes must wait for the resource in a cyclic manner where the last process waits for the resource held by the first process.
-



Circular Wait

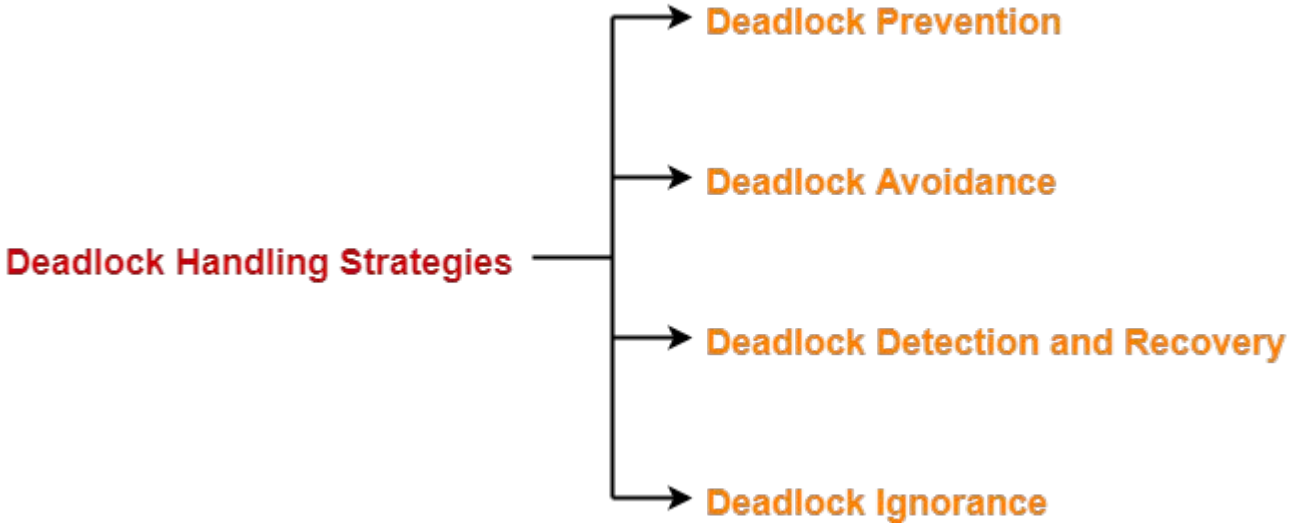
- Process P1 waits for a resource held by process P2.
- Process P2 waits for a resource held by process P3.
- Process P3 waits for a resource held by process P4.
- Process P4 waits for a resource held by process P1.

Important Note-

- All these 4 conditions must hold simultaneously for the occurrence of deadlock.
- If any of these conditions fail, then the system can be ensured deadlock free.

Deadlock Handling-

The various strategies for handling deadlock are-



Deadlock Prevention-

- This strategy involves designing a system that violates one of the four necessary conditions required for the occurrence of deadlock.
- This ensures that the system remains free from the deadlock.

1. Mutual Exclusion-

- To violate this condition, all the system resources must be such that they can be used in a shareable mode.
- In a system, there are always some resources which are mutually exclusive by nature.
- So, this condition can not be violated.

Hold and Wait-

Approach-01:

In this approach,

- A process has to first request for all the resources it requires for execution.
- Once it has acquired all the resources, only then it can start its execution.
- This approach ensures that the process does not hold some resources and wait for other resources.

Drawbacks-

The drawbacks of this approach are-

- It is less efficient.
- It is not implementable since it is not possible to predict in advance which resources will be required during execution.

Approach-02:

In this approach,

- A process is allowed to acquire the resources it desires at the current moment.
- After acquiring the resources, it start its execution.
- Now before making any new request, it has to compulsorily release all the resources that it holds currently.
- This approach is efficient and implementable.

Approach-03:

- A timer is set after the process acquires any resource.
- After the timer expires, a process has to compulsorily release the resource.

3. No Preemption-

- This condition can by violated by forceful preemption.
- Consider a process is holding some resources and request other resources that can not be immediately allocated to it.
- Then, by forcefully preempting the currently held resources, the condition can be violated.

A process is allowed to forcefully preempt the resources possessed by some other process only if-

- It is a high priority process or a system process.
- The victim process is in the waiting state.

Circular Wait-

- A natural number is assigned to every resource.
- Each process is allowed to request for the resources either in only increasing or only decreasing order of the resource number.
- In case increasing order is followed, if a process requires a lesser number resource, then it must release all the resources having larger number and vice versa.
- This approach is the most practical approach and implementable.
- However, this approach may cause starvation but will never lead to deadlock.

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‘sAlgo** is an example of a deadlock avoidance strategy.

Deadlock Detection and Recovery-

- This strategy involves waiting until a deadlock occurs.
- After deadlock occurs, the system state is recovered.
- The main challenge with this approach is detecting the deadlock.

Deadlock Ignorance-

- This strategy involves ignoring the concept of deadlock and assuming as if it does not exist.
- This strategy helps to avoid the extra overhead of handling deadlock.
- Windows and Linux use this strategy and it is the most widely used method.
- It is also called as **Ostrich approach**.

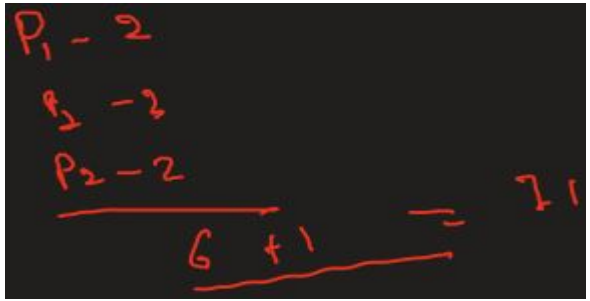
Problem-

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?

Ans p2

	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

A system contains three programs and each requires three tape units for its operation. The minimum number of tape units which the system must have such that deadlocks never arise is



Suppose n processes, P1, …. Pn share m identical resource units, which can be reserved and released one at a time. The maximum resource requirement of process Pi is Si, where Si > 0. Which one of the following is a sufficient condition for ensuring that deadlock does not occur?

Answer C

(a) $\forall i, s_i < m$

(b) $\forall i, s_i < n$

(c) $\sum_{i=1}^{n} s_i < (m + n)$

(d) $\sum_{i=1}^{n} s_i < (m * n)$

p1=s1-1

p2=s2-1

….

pn=sn-1

Toatal Sum = 

$\sum_{i=1}^{n} s_i$
$-m < m$
$s_i < m+1$
$\sum_{i=1}^{n}$