Bharathidasan University

Centre for Differently Abled Persons

Tiruchirappalli - 620024

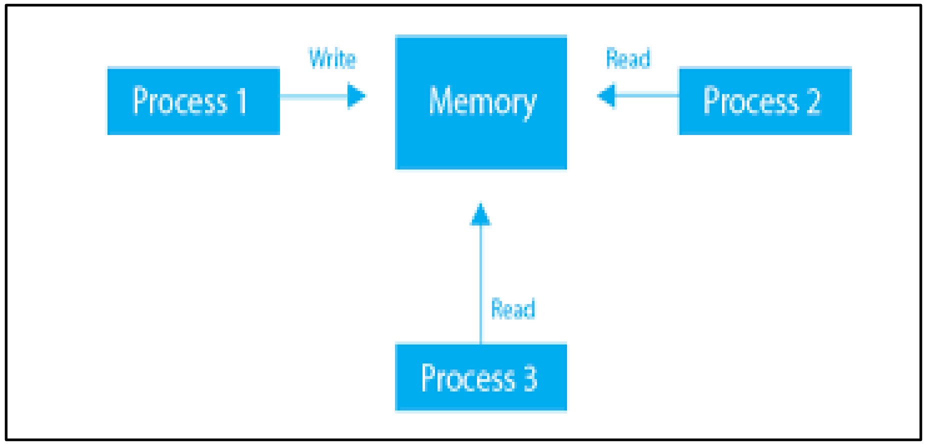
Programme Name	: Bachelor of Computer Applications
Course Code	: Operating Systems
Course Title	: 20UCA5CC5
Unit	: Unit III
Compiled by	: Dr. M. Prabavathy
	Associate Professor
	Ms. G. Maya Prakash
	Guest Faculty

Process Synchronization

PROCESS SYNCHRONIZATION

- Coordinating the execution of processes so that no two processes access the same shared resources and data is known as process synchronization.

- The main objective of process synchronization is to ensure that multiple processes access shared resources without interfering with each other.



On the basis of synchronization, processes are categorized as one of the following two types:

Independent Process: The execution of one process does not affect the execution of other processes.

Cooperative Process: A process that can affect or be affected by other processes executing in the system.

CRITICAL SECTION PROBLEM

- A critical section is a code segment that can be accessed by only one process at a time.
- The critical section contains shared variables that need to be synchronized to maintain the consistency of data variables.
- So the critical section problem means designing a way for cooperative processes to access shared resources without creating data inconsistencies.

```
do {

    entry section

         critical section

    exit section

         remainder section

   } while (TRUE);
```

Mutual Exclusion: If a process is executing in its critical section, then no other process is allowed to execute in the critical section.

Progress: If no process is executing in the critical section and other processes are waiting outside the critical section, then only those processes that are not executing in their remainder section can participate in deciding which will enter in the critical section next, and the selection can not be postponed indefinitely.

Bounded Waiting: A bound must exist on the number of times that other processes are allowed to enter their critical sections after a process has made a request to enter its critical section and before that request is granted.

SEMAPHORE

- A semaphore is a signaling mechanism and a thread that is waiting on a semaphore can be signaled by another thread.
- A semaphore uses two atomic operations, wait and signal for process synchronization.
- A Semaphore is an integer variable, which can be accessed only through two operations wait() and signal().

There are two types of semaphores: Binary Semaphores and Counting Semaphores.

Binary Semaphores:

- They can only be either 0 or 1.
- They are also known as mutex locks, as the locks can provide mutual exclusion.
- All the processes can share the same mutex semaphore that is initialized to 1.
- Then, a process has to wait until the lock becomes 0.
- Then, the process can make the mutex semaphore 1 and start its critical section.

Counting Semaphores:

- They can have any value and are not restricted over a certain domain.
- They can be used to control access to a resource that has a limitation on the number of simultaneous accesses.
- The semaphore can be initialized to the number of instances of the resource.
- Whenever a process wants to use that resource, it checks if the number of remaining instances is more than zero, i.e., the process has an instance available.
- Then, the process can enter its critical section thereby decreasing the value of the counting semaphore by 1.
- After the process is over with the use of the instance of the resource, it can leave the critical section thereby adding 1 to the number of available instances of the resource.

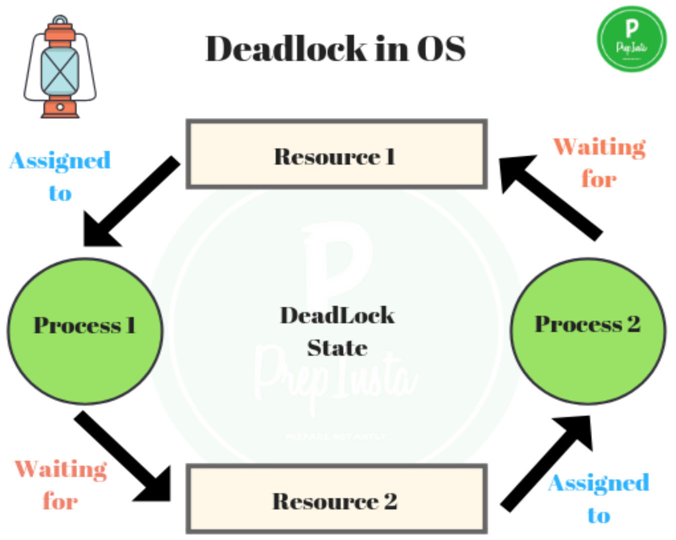
MONITORS

- Monitors are a synchronization construct that were created to overcome the problems caused by semaphores such as timing errors.
- Monitors are abstract data types and contain shared data variables and procedures.
- The shared data variables cannot be directly accessed by a process and procedures are required to allow a single process to access the shared data variables at a time.

DEADLOCK

INTRODUCTION TO DEADLOCK

- Deadlock is a situation where a set of processes are blocked because each process is holding a resource and waiting for another resource used by some other process.
-



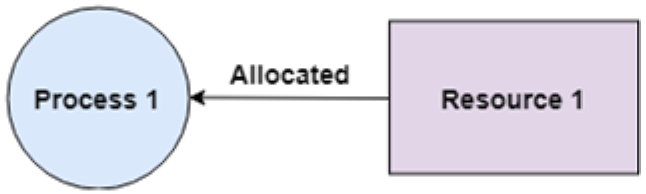
Resources

- A process in operating systems uses different resources and uses resources in following way.
- 1) Requests a resource
- 2) Use the resource
- 2) Releases the resource

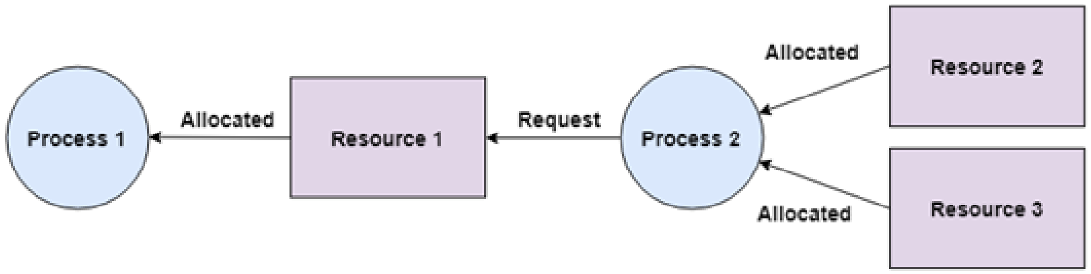
DEADLOCK PREVENTION

Deadlock arise if four condition holds simultaneously

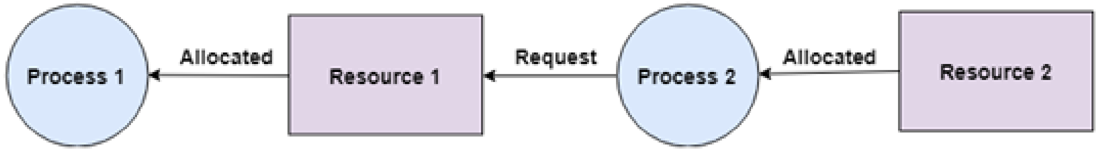
1. **Mutual Exclusion:** One or more than one resource is non-sharable (Only one process can use at a time)



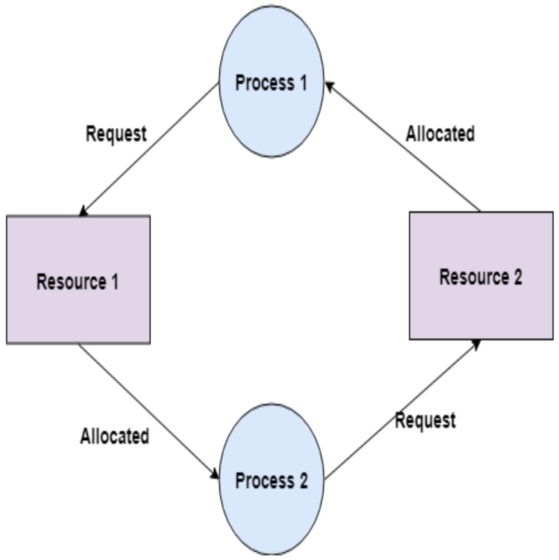
2. Hold and Wait: A process is holding at least one resource and waiting for resources.



3. No Preemption: A resource cannot be taken from a process unless the process releases the resource.



4. Circular Wait: A set of processes are waiting for each other in circular form.



DEADLOCK AVOIDANCE

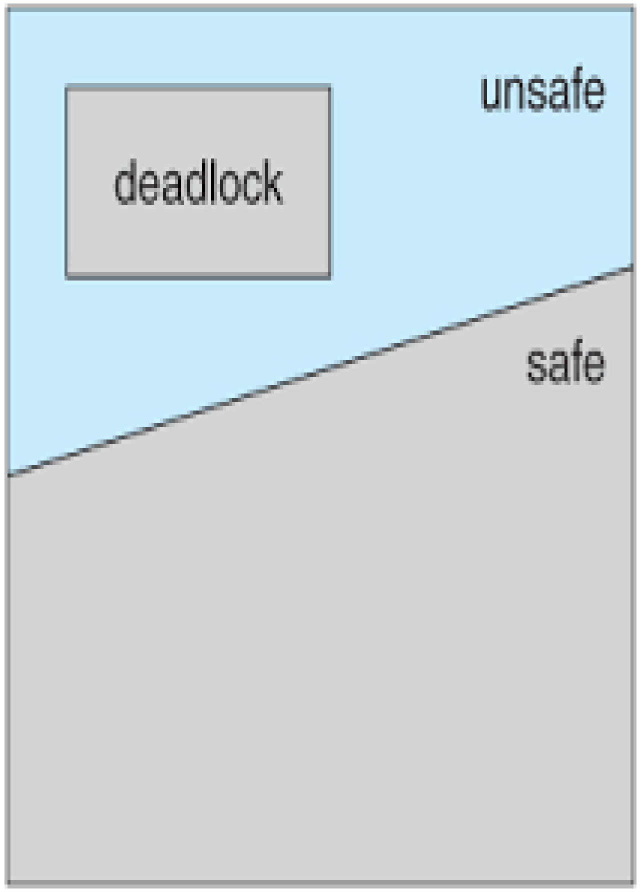
The system must know the total needed resources

1. Safe State

- A sequence of requests exists can be satisfied without deadlock
- Safe State → no deadlock

2. Unsafe State

- A sequence of requests cannot be satisfied
- Unsafe state → possibility of deadlock


unsafe
deadlock
safe

- Avoidance → ensure that system will never enter a unsafe state
- Algorithms Pseudo code explain about deadlock avoidance

1. System is initially **Safe**
2. At each **Request** for a resource

- i. **IF** allocation causes an unsafe state
- ii. **THEN** block the process
- iii. **ELSE** grant the allocation

3. At each resource **release**

- i. **WHILE** a block request can be granted safely
- Grant Allocation

4. **Unlock** the Process.

Deadlock Detection

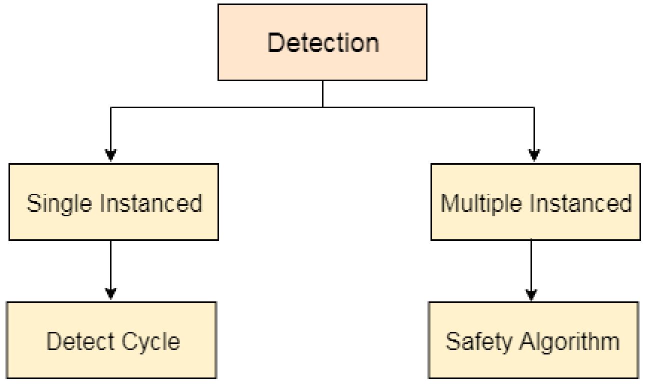
Deadlock Detection

Deadlock Detection is an important task of OS

The OS periodically checks if there is any existing deadlock in the system

Take measures to remove the deadlocks.

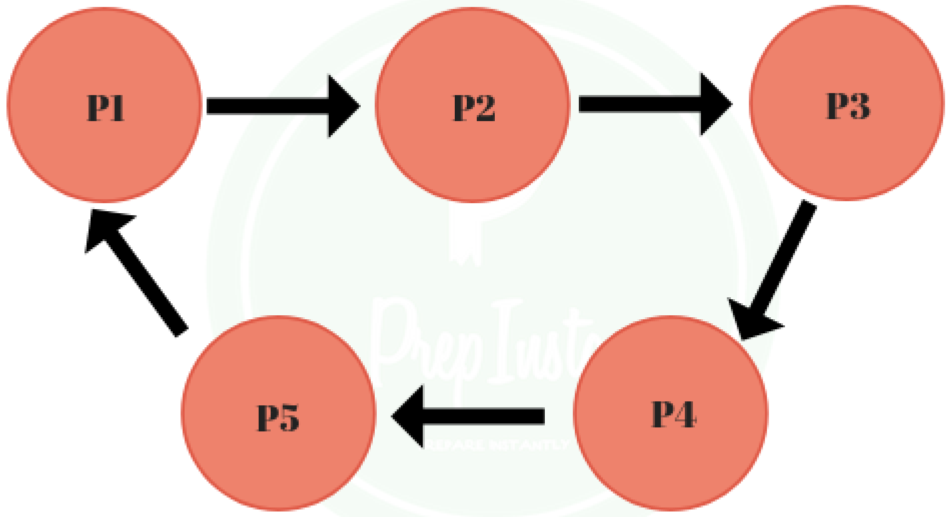
There are 2 types of detection



1. If resources have single instance:

- A wait-for graph is made.
- A Wait-for graph vertex denotes process.
- A deadlock is detected if one wait-for graph contains a cycle.
- In the above wait-for graph P1 is waiting for resource currently in use by P2
- P2 is waiting for resource currently in use by P3 and so on…
- P5 is waiting for resource currently in use by P1 which creates a cycle thus deadlock is confirmed.

Deadlock Detection
Cycle Detection



2. Several Instances of a Resource Type

- They are implemented using 4 data structure.
- Let n be the number of process in system.
- Let m be no of resources in system.

Available

- 1 D Array of Size m.
- Each element Available[i] indicates the number of resources of i type available. Denoted by Ri

Maximum

- 2 D Array of size n*m
- Determines maximum demand of each process.
- Maximum[i,j] tells the maximum demand an ith process will request of jth type resource.

Allocation

- 2 D Array of size n*m
- Defines number of resources of each type currently allocated to each process.
- Allocation[i,j] means i th process has current k instance of j th type resource

Need

- 2D Array of size n*m
- Tells about remaining resources type which are required by each process.
- Need[i,j] means i th process needs k instance of j th resource type.
- Need[i,j] = Maximum[i,j] – Allocation[i,j]
-

Deadlock Recovery

Deadlock Recovery

Killing the process:

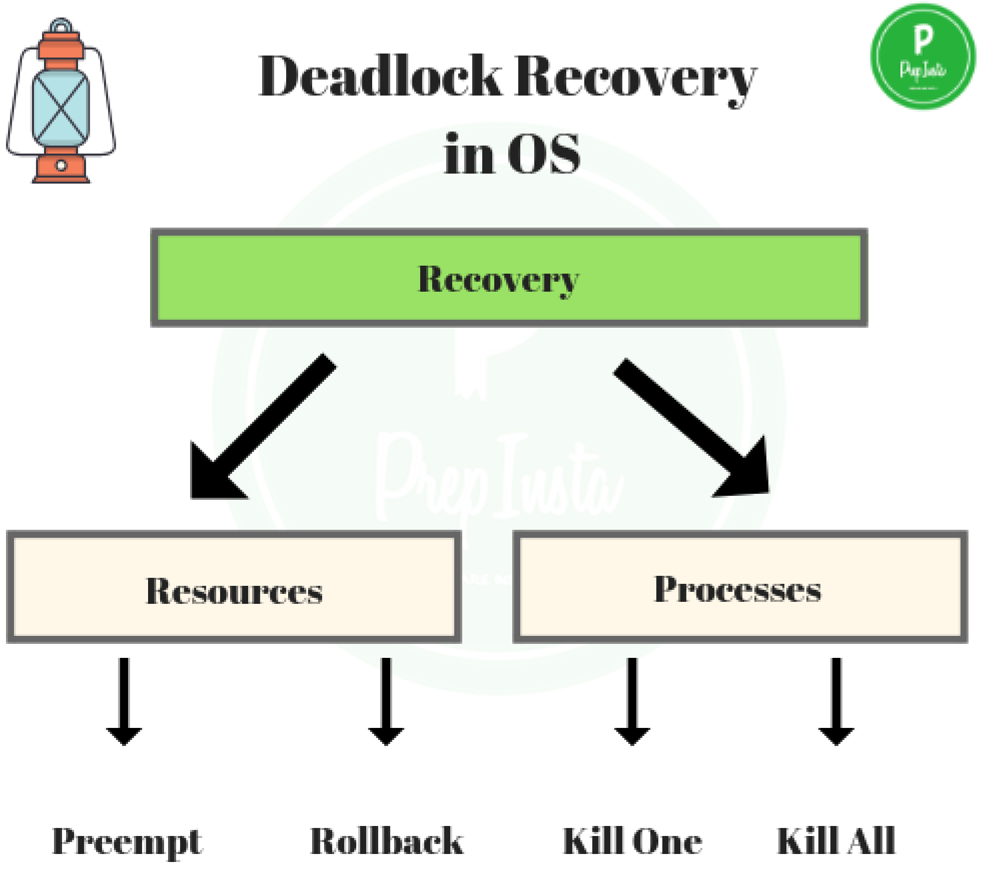
- I. killing all the process involved in the deadlock.
- II. Killing process one by one.
- III. After killing each process check for deadlock again
- IV. keep repeating the process till system recover from deadlock.

Resource Preemption:

- Resources are preempted from the processes involved in the deadlock,
- preempted resources are allocated to other processes
- By doing so, there is a possibility of recovering the system from deadlock.
- In this case, the system goes into starvation.

Rollback

- The OS maintains a database of all different states of system
- A state when the system is not in deadlock is called safe state.
- A rollback to previous ‘n’ number of safe states in iterations can help in the recover.


Deadlock Recovery in OS
Recovery
Resources
Processes
Preempt
Rollback
Kill One
Kill All

THANK YOU