

CHAPTER THREE

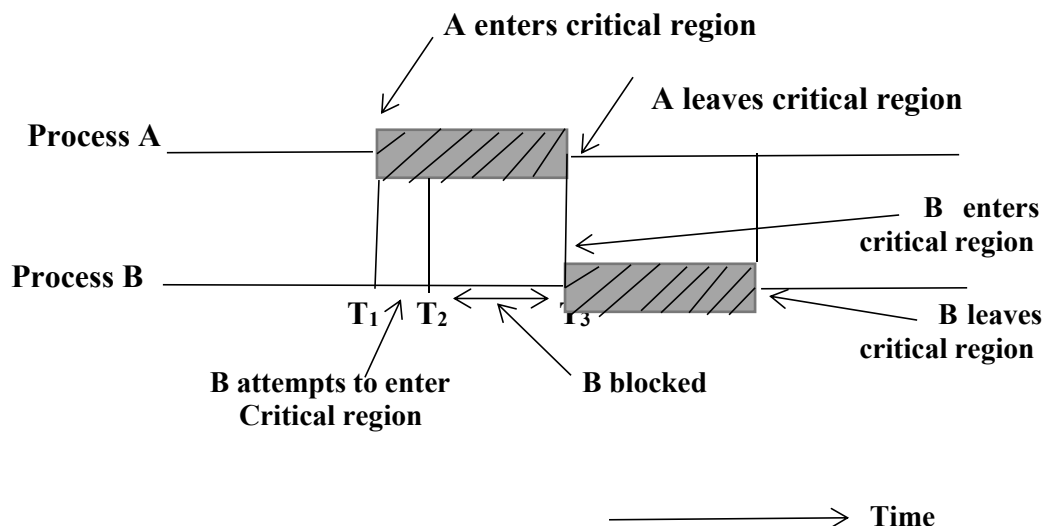
MUTUAL EXCLUSION

Definition

Mutual exclusion- implies that only one process can be inside the critical section at any time, if any other processes require the critical section, they must wait until it is free. In other words mutual exclusion mean at a time only one process can get an entry.

Critical section is a code segment where the shared variables can be accessed.

An atomic action is required in a critical section i.e. only one process can execute in its critical section at a time. All other processes have to wait to execute in their critical sections.



Critical region use in mutually exclusive way.

Mutual exclusive condition

If we could arrange matters such that no two processes were ever in their Critical Section simultaneously.

We need **four** condition to hold to have a good solution for the critical section problem

1. No two processes may at the same moment inside their critical sections.
2. No assumptions are made about relative speeds of processes or number of CPUs.
3. No process should outside its critical section should block other processes.
4. No process should wait arbitrary long to enter its critical section.

Mutual exclusive problem

In this problem, there is a collection of asynchronous process(process occur at different times), each alternately executing a critical and a noncritical section, that must be combined, so that no two process ever execute their critical sections concurrently.

Mutual exclusive is one of the key problem in distributed systems to allocate resource shared by processes.

Originally, mutual exclusive is defined as that a set of processes intermittently requires access to an identical resource called the critical section and no more than one process can be in critical section at any given time.

Deadlock

A deadlock happens in operating system when **two or more processes** need some resource to complete their execution that is held by other process.

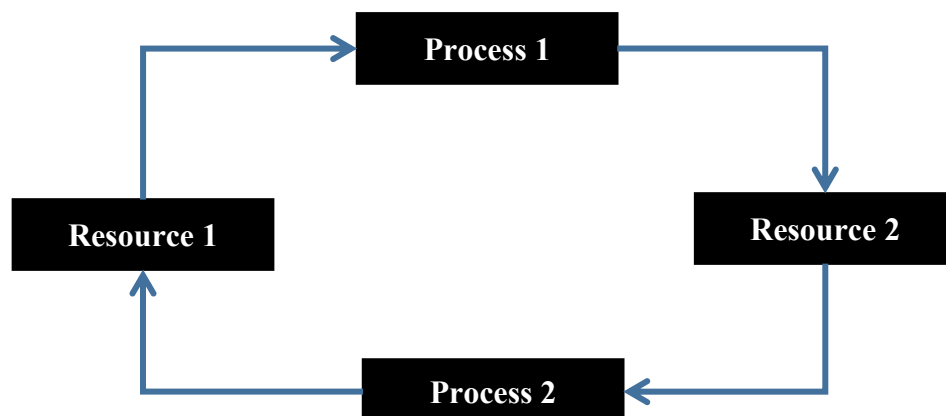


Fig: deadlock

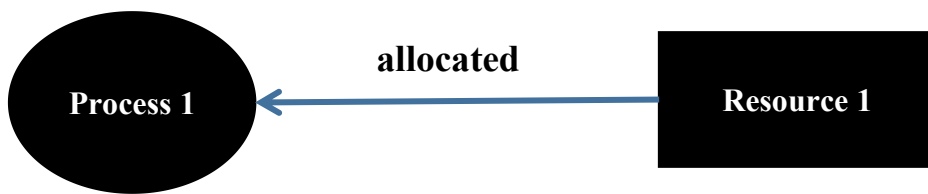
- Process 1 has resource 1 and needs to acquire resource 2.
- Similarly process 2 has resource 2 and needs to acquire resource 1.
- Process 1 and process 2 are in deadlock as each of them needs the others's resource to complete their execution but neither of them is willing to relinquish their resources.

Condition for deadlock

1. Mutual exclusion

- ✓ There should be a resource that can only be held by one process at a time.

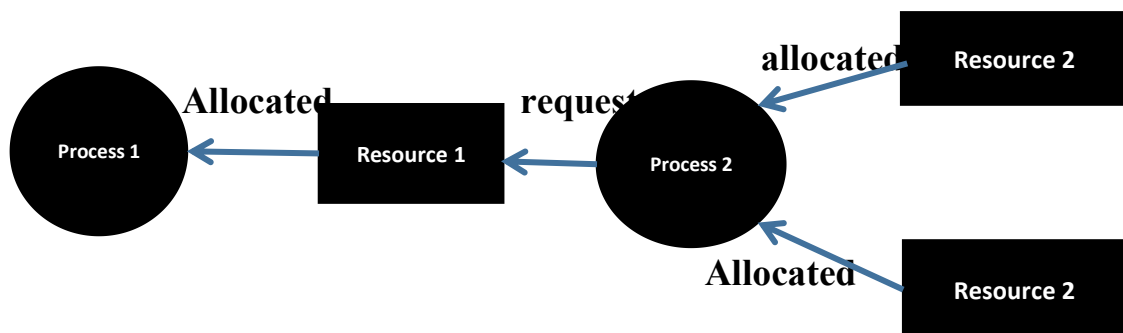
In the diagram below, there is a single instance of resource 1 and it is held by process 1 only.



2. Hold and wait

- A process can hold multiple resources and still requires more resources from other process which are holding them.

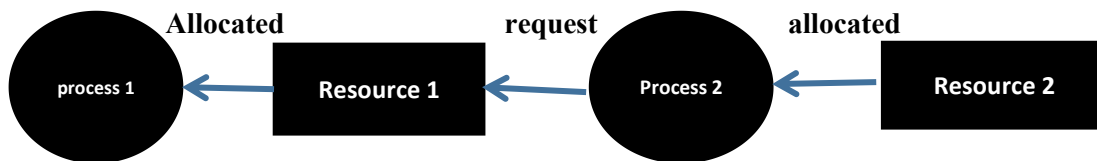
In the diagram, process 2 holds resource 2 and resource 3 and is requesting the resource 1 which is held by process 1.



3. No preemption

- ✓ A resource cannot be preempted from a process by force.
- ✓ A process can only release a resource voluntarily.

In the diagram below, process 2 cannot preempt resource 1 from process 1. it will only be released when process 1 relinquishes it voluntarily after its execution is complete.

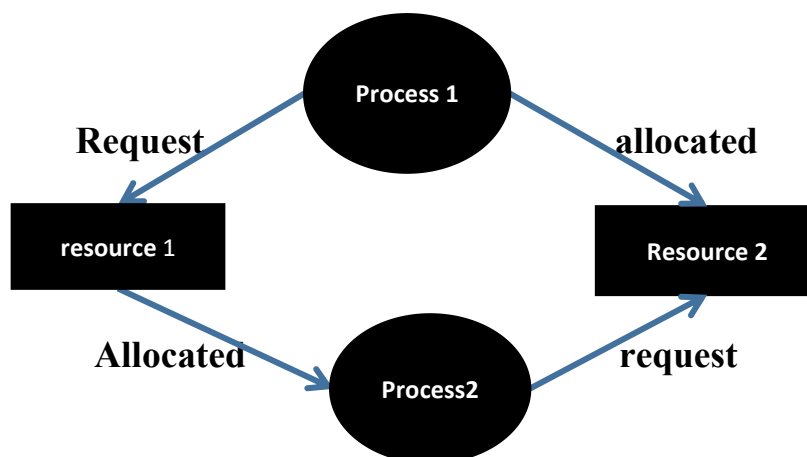


4. Circular wait

- ✧ A process is waiting for the resource held by the **second process**, which is waiting for the resource held by the **third process** and so on, till the **last process** is waiting for a resource held by the **first process**.
- ✧ This forms a circular chain.

For example: process 1 is allocated resource 2 and it is requesting resource 1.

Similarly, process 2 is allocated resource 1 and it is requesting resource 2. This forms a circular wait loop.



Deadlock detection

A deadlock can be detected by a resource scheduler as it keeps track of all the resources that are allocated to different processes.

After a deadlock is detected, it can be resolved using the following methods:

- ✓ All the processes that are involved in deadlock are terminated. This is not a good approach as all the progress made by the processes is destroyed.
- ✓ Resources can be preempted from some processes and given to others till the deadlock is resolved.

Deadlock prevention

It is very important to prevent a deadlock before it occurs.

So, the system checks each transaction before it is executed to make sure it does not lead to deadlock.

If there is even a slight chance that a transaction may lead to deadlock in the future, it is never allowed to execute.

Deadlock prevention strategy

Design a system in such a way that the possibility of deadlock is excluded.

Two main methods:

1. Indirect

- Prevent the occurrence of one of the three necessary conditions.

2. Direct

- Prevent the occurrence of a circular wait.

Deadlock avoidance

Requires knowledge of future process requests.

Two approaches to deadlock avoidance:

1) Resource allocation denial

- Do not grant an incremental resource request to a process if this allocation might lead to deadlock.
- Referred to as the **banker's algorithm**.

- ◆ **State:** of the system reflects the current allocation of resources to processes

- ◆ **Safe state:** is one in which there is at least one sequence of resource allocation to processes that does not result in a deadlock.
- ◆ **Unsafe state:** is a state that is not safe.

2) Process initiation denial

- Do not start a process if its demands might lead to deadlock.

Wait for graph(WFG)

The state of the system can be modeled by directed graph, called wait for graph.

In a WFG, nodes are processes and there is a directed edge from node p_1 to node p_2 , if p_1 is blocked and is waiting for p_2 to release some resource. A system is deadlock if and only if there exists a directed cycle or knot in the WFG.

Knot: subgraph such that starting from any node.

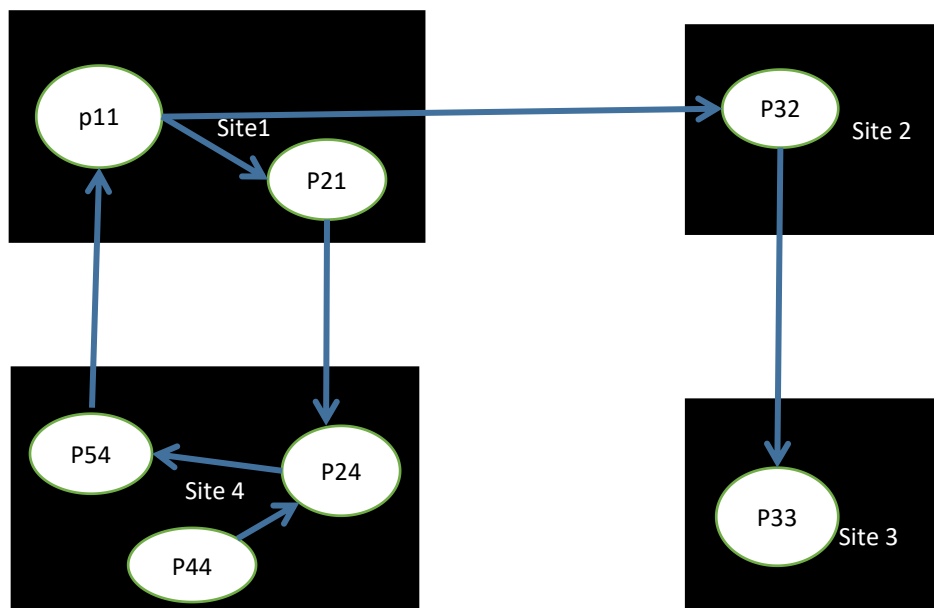


Figure: WFG

Where processes p_{11} of site1 has an edge to process p_{21} of site1 and p_{32} of site2 is waiting for a resource which is currently held by process p_{21} . At the same time process p_{32} is waiting on process p_{33} to release a resource.

If p_{21} is waiting on process p_{11} , then processes p_{11} , p_{32} and p_{21} form a cycle and all the four processes are involved in a deadlock depending upon the request model.

Models of deadlock

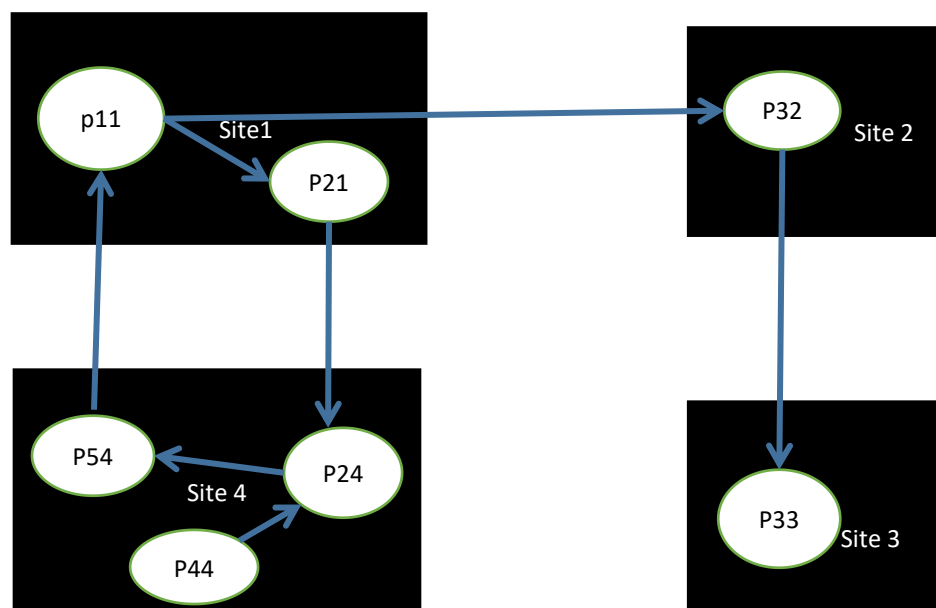
A hierarchy of request models are described as follows:

i. The single resource model(single unit request model)

- ✓ In this a process can have at most one outstanding request for only one unit of a resource,
- ✓ Since the maximum node in a WFG for the single resource model can be 1, the presence of a cycle in the WFG shall indicate that there is a deadlock.

ii) The AND model

- A process can request for more than one resource simultaneously and the request is satisfied only after all the requested resources are granted to the process.
- The node in the WFG for AND model can be more than 1.
- The presence of a cycle in the WFG indicates a deadlock in the AND model.

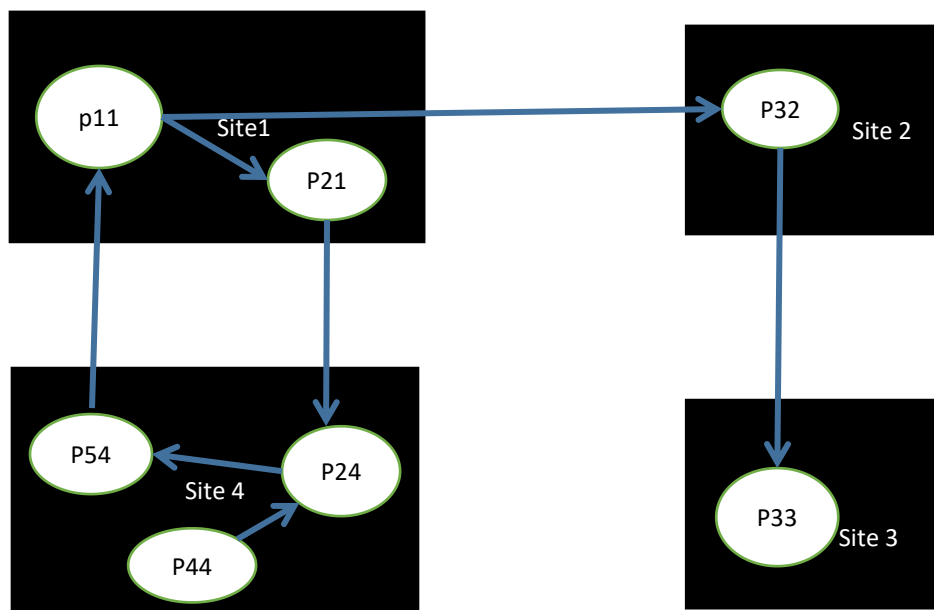


Consider the example WFG is described in the figure.

- ✓ P11 has two outstanding resource requests. In case of the AND model, p11 shall become **active** from **idle state** only after both the resources are granted.
- ✓ There is a cycle $p11 \rightarrow p21 \rightarrow p24 \rightarrow p54 \rightarrow p11$ which corresponds to a deadlock situation.
- ✓ P44, is not a part of any cycle but is still deadlocked as it dependent on p24 which is deadlocked.

iii) the OR model

- ✧ In this, a process can make a request for numerous resources simultaneously and the request is satisfied if any one of the requested resources is granted.
- ✧ Presence of a cycle in the WFG of OR model does not imply a deadlock in the OR model.



Consider the example WFG is described in the figure.

- If all nodes are OR nodes, then process p11 is not deadlocked because once process p33 releases its resources, p32 shall become active as one of its requests is satisfied.
- After p32 finishes execution and releases its resources, process p11 can continue with its processing.

iv) The AND - OR model

- ✧ A generalization of OR and AND model.
- ✧ In this, a request may specify any combination of AND and OR in the resource request.
- ✧ Example: in the AND - OR model, a request for multiple resources can be of the form X and (Y or Z).
- ✧ Deadlock in the AND - OR model can be detected by repeated application of test for OR-model deadlock.

Semaphore

A bit, token, fragment of code or some other mechanism which is used to restrict access to a shared function or device to a single process at a time, or to synchronize and coordinate events in different processes.

Also is a variable that has an integer value which only three operations are defined

- ✓ Initialize
- ✓ Decrement and
- ✓ Increment

There is no way to inspect or manipulate semaphores other than these three operations.

Strong/ weak semaphores

Strong semaphores

The process that has been blocked the longest is released from the queue first (FIFO).

Weak semaphores

The order in which processes are removed from the queue is not specified.

Queue:- a data structure in which objects are added to one end, called the **tail** and removed from other called the **head**.

Monitors

Programming language construct that provides equivalent functionality to that of semaphores and it is easier to control.

Implemented on a number of programming language.

Has also been implemented as a program library.

Software module consisting of one or more procedures, an initialization sequence, and local data.

Monitor characteristics

- Local data variables are accessible only by the monitor's procedures and not by any external procedure.
- Process enters monitor by invoking one of its procedures.
- Only one process may be executing in the monitor at a time.

Condition variable

A data type that is used to block a process or thread until a particular condition is true.

Producer - consumer problem

- ✓ One or more producers are generating data and placing these in a buffer.
- ✓ A single consumer is taking items out of the buffer one at a time.
- ✓ Only one procedure or consumer may access the buffer at any one time.

The problem

Ensure that the producer can't add data into full buffer and consumer can't remove data from an empty buffer.

Buffer:- a portion of memory set aside to store data, often before it is sent to an external device or as it is received from an external device.

Buffer:- a portion of memory set aside to store data, often before it is sent to an external device or as it is received from an external device.

Synchronization

Achieved by the use of condition variables that are contained within the monitor and accessible only within the monitor.

A condition variables are operated on by two functions:

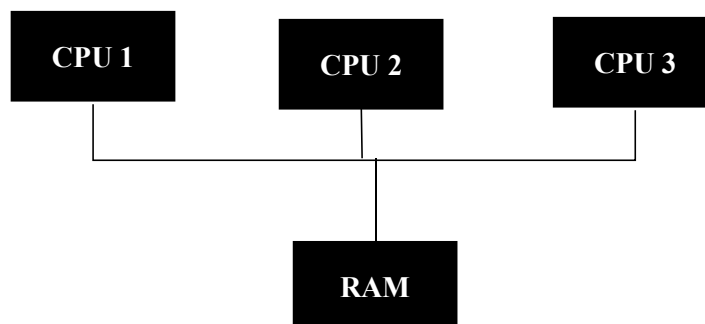
- ✓ **Cwait(c):** suspend execution of the calling process on condition c.
- ✓ **Csignal(c):** resume execution of some process blocked after a cwait on the same condition.

Multiprocessor

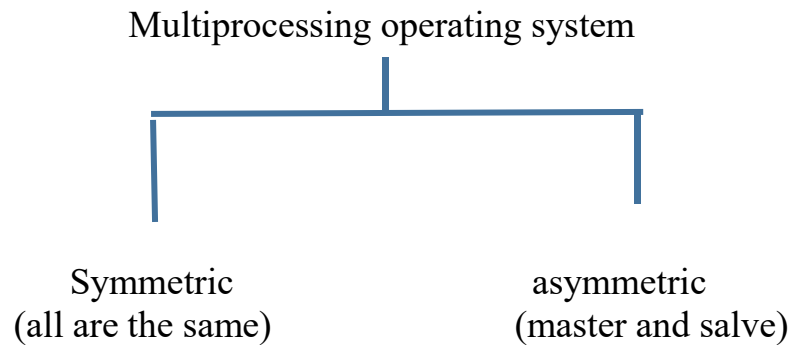
Multiprocessing means multiple processor. In a single system, we have multiple processor, like:

- Octal core -> 8 processor
- Dual core -> 2 processor

In multiprocessing operating system more than one process can run on different CPU at the same time, so there is real parallel execution of process.



It is required when number of process is very high and a single processes takes very long time to deal with all these processes.



1. **Symmetric:-** all CPU have the same function and access control.
2. **Asymmetric:-** one CPU will be master and other will be slave.
- ✓ Master CPU will decide which process will handle by which CPU.