PAGE NO. :
DATE : / /

Unit 4

Introduction to Deadlock.

Normal mode of operation A system consist of ~~Infinit~~ finite number of resources to be distributed among a number of competing processes the resources are partion into several types, each type consisting of sum no of identical instances. memory space, CPU cycles, files, Input / Output devices are examples of Resource types. If a system has two printers than the resource type printer has two instances.

A process must request a resource before using it and must release the resource after using it A process may request as many resources as it requires to carryout to it's tasks. the no of resources requested may not exceed the total no of resources available in the system.
On a normal mode of operation a process may utilize the resource in the following sequence

1. Request
2. Use
3. Release.

PAGE NO. :
DATE : / /

Request – A process first make a request to the OS for required resources. The process can use the resources only after it is allocated to that process. If the request is not granted then the process may have to wait until the resource is allocated to it.

Use – The process can operate on the resource, release.

Release :– After using the resources, the process releases all the resources

Deadlock. – A set of processes is in deadlock state when every process in the set is waiting for an event that can be cause only by another process in the set the event which generally causes deadlock is resource acquisition & release
In a deadlock, a process never finishes execution & system resources are tide up preventing other processes from execution.
~~necessary and sufficient condition.~~
Charecterization in Deadlock.
necessary and sufficient condition for deadlock
Resource allocation graph.

necessary and Sufficient Condition for deadlock

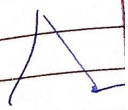
PAGE NO. :
DATE : / /

A deadlock situation may occur in a system if the following four condition hold simultaneously.

1. Mutual exclusion
2. Hold & wait
3. No preemption
4. Circular wait

Mutual exclusion: At least one resource in a system must be held in non-shareable mode that is only process can used the resource if another process request that resource, the requesting process must be delayed until the resource has been released.

Processes
resources
request edge



request edge

Hold and wait — A process must be holding at least one resource and waiting to acquire additional resources that are currently being held by other processes

No preemption — resources cannot be preemptive that is a resource can be released voluntarily by the process holding it after completing it's execution.

Circular wait — A set of $\{P_0, P_1, P_2, P_3 - P_n\}$ waiting process must such that resource held by P_1 is P_1 is waiting for a resource held by P_2 and so on P_n is

PAGE NO. :
DATE : / /

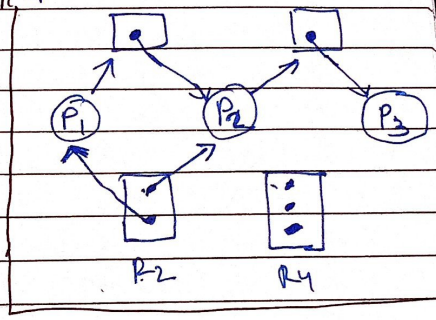
Waiting for a resource held by Po..
11
All the four conditions must hold for deadlock to occur.

Resource allocation graph →

Processes $P = \{P_1, P_2, P_3\}$

resources $R = \{R_1, R_2, R_3, R_4\}$

request edge $\{P_1 \Rightarrow R_1, P_2 - R_3, R_1 \rightarrow P_2 \; R_3 \rightarrow P_3, R_2 - P_1 \; R_2 \rightarrow P_2\}$



request edge

assignment or allocation.

R_1	1	→ Instances
R_2	2	
R_3	1	
R_4	3	

Deadlock can be describe more preciously using directed graph called resource allocation graph a graph set of vertices and a set of edges the resoure allocation graph has two different type of vertices The ciscular vertices represents the process and the rectangular vertices represents the resource types of the system the no of dots in the resource type represents the no of instances of that resource type. In resource allocation graph the edges also

PAGE NO. :
DATE : / /

represents two different type of edges a directed edge $P_i \rightarrow R_j$ is called as a Request edge. and a directed edge $R_j \rightarrow P_i$ is represents an Assignment edge.

Note that a request edge points to only the rectangle R_j, where as an assignment edge must also designate one of the dots in the rectangle. When a request for a resource is fulfilled then the request edge is transformed into assignment edge. When the resource is released then the assignment edge is simply deleated from the resource allocation graph.

The resource allocation graph can be used to define the following.

1. If the graph contains no cycle then no process in the system is deadlocked.
2. If the graph does contains a cycle then a deadlock may exist. If each resource type has exactly one instance then a cycle implies that a deadlock has occured. If the cycle involves only a set of resource types, each of which has only a single instance then a deadlock has occured. each process in the cycle is deadlocked
3. Each resource type has Several Instances

PAGE NO. :
DATE : / /

then a cycle does not necessarily imply that a deadlocked has occured
Consider the following example.

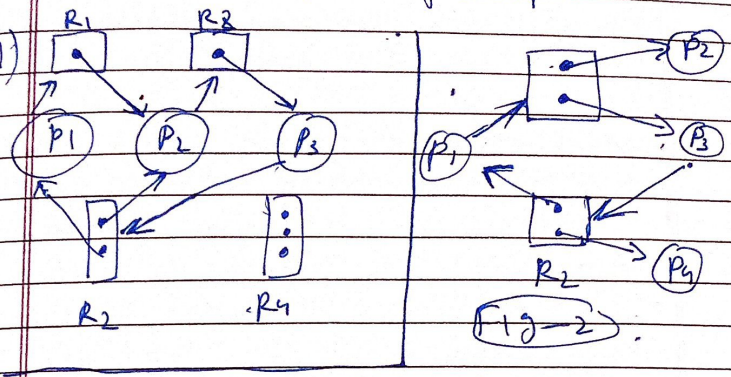


Fig - 2

RAG with a cycle & deadlock

From Fig (2) →
Resource allocation graph with a cycle but no deadlock

Deadlock handling method:
1 Deadlock avoidance
2 Deadlock prevention
There are two ways to handle deadlock
Deadlock avoidance : Deadlock avoidance requires that the Operating System be given in advance additional information concerning which resources a process will request and use during it's lifetime. With all the information the System can decide. The correct way to allocate resources so that

PAGE NO. :
DATE : / /

the deadlock does not occur.

Deadlock prevention: DP provides a set of methods for ensuring that atleast one of the necessary condition cannot hold. The conditions can be prevented by following ways.

Mutual exclusion: This condition can be prevented by making the resources shareable but it is not always possible to make resources shareable because some resources are intrinsically non-shareable. In this situation the no of instances of that resource can be increased but it is not very cost efficient.

Hold & wait: This condition can be prevented by guaranteeing that whenever a process request a resource it cannot hold any other resource. In other words a process must request for all the resources. In the beginning and these resources should be collocated before the beginning of execution.

No Preemption: The third necessary condition for deadlock is that there be no preemption of resources that had already being allocated. To ensure that this condition doesnot hold we can use the following protocols.

PAGE NO. :
DATE : / /

If a process is holding some resources and request another resource that cannot be immediately allocated then all the resources currently being held are preemptive.

Circular wait: One way to ensure that this condition never holds is to impose a total ordering of all resource types and to require that each process request resources in an increasing order of number. for example.
Consider a set of resource type

$$R = \{R_1, R_2, R_3, \ldots, R_m\}$$

We assign to each resource type a unique number which allows us to compare ~~two~~ compare two resources and to determine whether one other precedes another in ordering

Now a process can request resources only in and increasing order of number
A process can request any no of instances of a resource type R_i. after that process can request instances of resource type are R_j if and only if $F(R_j) > F(R_i)$ where

PAGE NO. :
DATE : / /

F is a 1 to 1 function such that
F : R → N
function resource natural

If several instances of the same resource are needed then the process must send a single request
Using this protocol we can prevent circular wait.