Section - II

Jeurkar. S.S

ch - 6 Deadlocks

- In multiprogramming envt. Several processes may compete for finite no. of resources.
- A process requests resources and if the resources are not available at that time, process enters a waiting state.
- Sometimes a waiting process is never again able to change the state because the resources it has requested are held by other processes
- This situation is called a deadlock.

* Dealock & Indefinite postponment (starvation)

- Deadlock occurs because the event will never occure;
- Indefinite postponment (starvation) - Here, in which a process that is not deadlocked could wait for an event that might ~~occur~~ never occur or might occure unpredictably far in future because of biases in the Sys resource - scheduling policies.

1* System model

- A System consists of finite no. of resources to be distributed among a

number of competing processes.

- 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 carry out its designated task.
- The no. of resources requested may not exceed the total no. of resources available in the sys.
- Under the normal mode of operation a process may utilize a resource in only the following sequence.

(1) Request :- If the request cannot be granted immediately (for e.g if the resource is being used by another process), then the requesting process must wait until it can acquire the resource.

(2) Use :- The process can operate on the resource.

(3) Release :- The process releases the resources.

- Request & releases of resources are accomplished by sys calls like request(), release() device, open() & close() files, allocate() & free() memory.

Certain resources are non preemptable they cannot be removed from the process to which they are assigned until the process itself release them e.g tape drive, optical scanners.
The set of processes is in a deadlock state when every process in the set is waiting for an event that can be caused only by other process in the set. This event is resource acquisition & release. The resources are either physical (printer, mem, CPU cycles) or logical (files, semaphores).

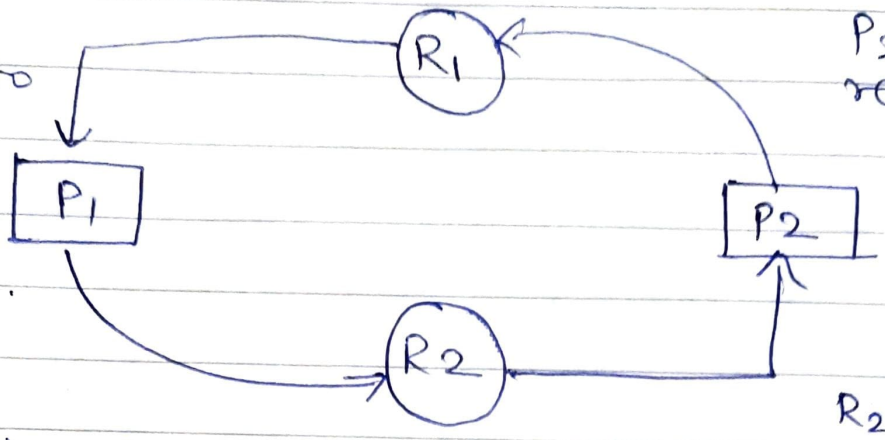
e.g of deadlocks.

(1) Consider a sys with three CD RW drives. Suppose each of three processes holds one of the these CD RW drives. Now if each process requests another drive, 3 processes will be in deadlock.

(2) 1 printer, 1 DVD drive, P_i holding DVD, P_j holding printer. P_i requests printer & P_j requests DVD drive, deadlock.

(3) Two process holding semaphores waiting other to release.

Resource allocation graph



- Above fig shows simple example of resource deadlock.
- Two processes — rectangle
  Two resources — Circle.
- Arrow from resource to process indicates that it has been allocated to process.
- Arrow from process to resources indicate that process is requesting, but has not yet been allocated.
- Above diag represents deadlock situation:-
  P1 → holds R1, needs R_2 to continue.
  P2 → holds R2, needs R_1 to continue.
- Each process is waiting for other to free a resource that the other process will not free.
- This circular wait is the characteristic of deadlocked sys.

2] ✱ Deadlock characterization

- In a deadlock,
  (1) Processes never finish executing
  (2) Resources are tied up.
  (3) Preventing other jobs from starting.

Following four conditions are necessary for deadlock to exist:

1) Mutual exclusion:— A resource may be acquired exclusively by only one process at a time.

2) Hold and wait :- A process must be holding at least one resource and waiting to acquire additional resource that are currently being held by other process.

3) No pre-emption :— A resource can not be preempted, i.e a resource can be released by only a process that owns it, after that process has complete its task.

4) Circular wait Condition :- Two or more processes are locked in "circular chain" in which each one is waiting for other resource that the next process in the chain is holding.

e.g Po waiting for P_1, P_1 for P_2 --- and so on.

3] * Methods for handling deadlock (Deadlock Solutions)

1) Deadlock Prevention – is to condition a sys to remove any possibility of deadlocks occuring. (can result in poor resource utilizat?)

2) Deadlock avoidance :- to impose less stringent conditions than deadlock prevention in attempt to get better resource utilization.
- It do not precondition the sys to remove all possibility of deadlock. Instead allow them to enter, detect and recover.

3) Deadlock detection :- determine if a deadlock has occured, identify the process & resources that are involved.

4) Deadlock recovery :- Used to clear the deadlocks from a system so that it may operate free them, and so that deadlocked process may complete their exe? and can free their resources.

PAGE NO:
DATE:

4] * Deadlock Prevention

- As we know for a deadlock to occur each of the four necessary conditions must hold.
- By ensuring that at least one of these conditions cannot hold we can prevent the occurrence of deadlock.

[1.] Mutual exclusion

- We cannot prevent deadlocks by denying the mutual exclusion cond^n because some resources are nonsharable for. e.g a printer could not be simultaneously shared by several processes.
- Sharable resources require mutual exclusive access and thus cannot be involved in deadlock. e.g several process attempt to open read-only file at the same time, they can granted simultaneous access.

[2.] Hold and wait (Denying "wait for" condi^n)

- To ensure that hold & wait cond^n never occurs in the sys we must guarantee that, whenever a process request a resource, it does not hold any other resources.
- One protocol that can be used that

all of the resources a process needs to complete its task must be requested at once. Sys must grant them on an "all or none" basis.

- If all the resources are available then sys may grant them all to process at once before it begins exe?
- If they are not available then process must wait until they are.
- While the process waits, however it may not hold any resources. Thus the "wait for" condition is denied and deadlock cannot occur.
- An alternative protocol allows the process to request additional resources only when it is none. A process may request some resources and use them. Before it can request any additional resources however it must release all the resources that it is currently allocated.

e.g. - process that copies data from DVD drive to file disk, sort the file and then prints result to printer.

- Disadvt - By ~~1st~~ methods 1) All resources allocated but unused for longer time.

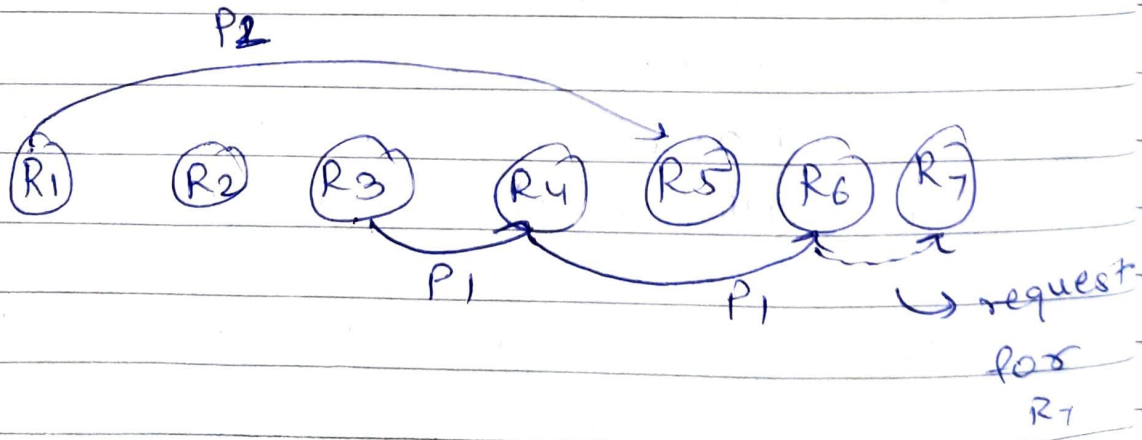
(2) Starvation is possible - A process who needs several resources may have to wait indefinitely.

[3.] No preemption (Denying the "No preemption")

- When a process holding resources is denied a request for additional resources (i.e the process must wait that cannot be immediately allocated)
- Process must release the resources (all) it holds and if necessary request them again together with old and new resources. This strategy effectively denies the no-preemption condition. The process will be re-started only ~~if~~ when it can regain its old + new resources.
- Alternatively, if process request some resources, first check whether they are available, if available allocate them. If they are not available, check whether they are allocated to some other process that is waiting for additional resources. If so we preempt desired resource from waiting process and allocate them to the requesting process.
- This protocol is often applied to resources whoes state can be easily saved and restored later such as CPU registers and mem space.

[4] Circular wait (Denying the "circular wait")

- This strategy denies the possibility of circular wait.
- In this assign a unique num to each resource that the sys manages and we create a linear ordering of resource.
- A process must then request a resource in a strictly ascending order.
- e.g if P1 request R3 then the process can request only resources with a no. greater than 3.
- Because all resources are uniquely numbered and because process must request resource in ascending order a circular wait cannot develop



Disadvt :- 1) If resources are added/deleted it will affect the numbering
2) Provided ordering must be lived with Sys for longer time
3) If some process requires in diff order than specified, results in Poor util.

5 * Deadlock Avoidance

- A method for avoiding deadlocks is to require an additional information about how resources are to be requested.
- with this knowledge of the complete sequence of requests and release for each process, the sys can decide for each request whether or not the process ~~sche~~ should wait in order to avoid a possible future deadlock. Each request requires that in making this decision the system considers the resources currently available, the resources currently allocated to each process, and the future requests and release of each process.

↳ The Banker's Algorithm

- The banker's algo defines how a partiala sys can prevent deadlock by carefully controlling how resources are distributed to users.
- The Bankes algo prevents deadlock in o.s that exhibit the following properties.

(A) The o.s shares a fixed no. of resources t, among fixed no. of process n.

(B) Each process specifies in advance the max^m no. of resources that it requires to complete its task.

(c) The O.S accepts a process's request if that process's max^m need does not exceed the total no. of resources available in the sys,t.

(d) Sometimes a process may have to wait to obtain an additional resource but the O.S guarantees a finite wait time

(e) If the O.S is able to satisfy process's max need for resources, then the process guarantees that the resource will be used and released to the O.S within a finite time.

- Safe state :- The sys is said to be in safe state if the O.S can guarantee that all current processes can complete their work within a finite time. If not, the sys is said to be in unsafe state.

* following terms describes the distribution of resources among process →

PAGE NO.
DATE

- Let max(P_i) :- max^m no. of resources required by P_i
  e.g max(P_3) = 2, P_3 never require > 2,

- Let loan(P_i) :- no. of resources allocated to P_i, e.g loan(P_5) = 4

- Let claim(P_i) :- current claim of process, where a process's claim is equal to its max^m need minus its current loan.
  for e.g if P_7 has max^m need of 6 resource and current loan of 4 ~~pro~~ resources then we have

$$\text{claim}(P_7) = \max(P_7) - \text{loan}(P_7)$$
$$= 6 - 4$$
$$= 2.$$

- Let 'a' be the no. of resources still available for allocation, it is equivalent to t − (sum of loans to all process in sys)

$$a = t - \sum_{i=1}^{N} \text{loan}(P_i)$$

* Example of Safe state

Process	max(P_i) (max need)	loan(P_i) (current loan)	claim(P_i) (current claim)
P1	4	1	3
P2	6	4	2
P3	8	5	3

Total resource $t = 12$

Available resource $a = 2$

a = is computed by, Substracting the sum of current ~~claims~~ loan from the total no. of resources, $t = 12$

$12 - (1 + 4 + 5) = 2$.

- Above sys is in safe sate. If sys allocate these 2 available resources to P2, fullfilling P2's max need. P2 will complete its exe releases 6 resources enabling sys to immediately fullfill max need of P1(3) and P3(3).

* Example of unsafe state

Process	max (P_i)	loan(P_i)	claim(P_i)
P1	10	8	2
P2	5	2	3
P3	3	1	2

Total resources

$t = 12$

available resource $a = 1$

$(12 - (8 + 2 + 1) = 1)$

- If P1 requests and is granted the last available resource. A three way deadlock could occur.
- If Unsafe state does imply that some unfortunate sequence of events might lead to a deadlock.
- Unsafe state doesn't guarantee deadlock but it indicates possibility of deadlock occurences due to improper sequence of resource allocation.

↳ Procedure in Bankers algorithm for resource allocation

- The "mutual exclusion", "wait-for" & "no preemption" conditions are allowed- Processes are indeed allowed to hold resource while requesting and waiting for additional resources and resources may not be preempted from a process holding those resources.
- Processes ease onto the sys by requesting one resource at a time; the sys may either grant or deny each request.
- If a request is denied that process holds any allocated resources and waits for a finite time until that

request is eventually granted.

- The sys grants only requests that result in safe states.
- Resources requests that would result in unsafe states are repeatedly denied.
- Bcoz the sys is always maintained in a safe state, sooner or later (i.e in a finite time) all requests can be satisfied and all users can finish.

↳ Weaknesses in Banker Algo

1) fixed no. of resources are not guaranteed in the sys (e.g USB)
2) Process population can not be fixed in todays interactive & multiprogrammed sys. (Process population is constantly changing)
3) Algo requires that, grant all resources in "finite time" and each process should return all resources within "finite time" it is actually difficult in real-time system.
4) In dynamic resource allocation, it is very difficult to know in advance, processes max^m need of resources

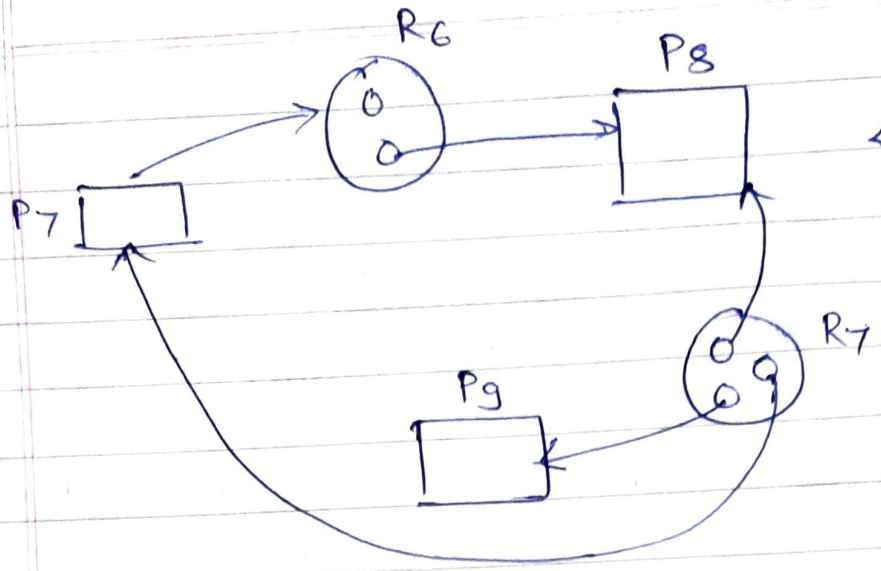
6] * Deadlock Detection

- Recovery can be carried out by detecting the deadlock.
- The Algorithm examines the state of sys to determine whether deadlock has occured, and then to recover from deadlock
- Deadlock detection is a process of determining that a deadlock exists and identifying the process and resources involved in deadlock

* Reduction of resource allocation graph

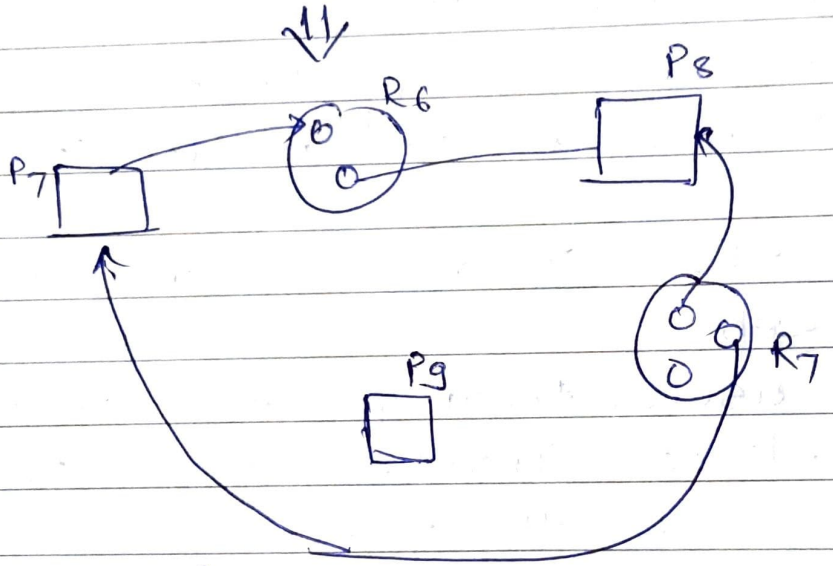
- A useful technique for detecting deadlocks involves graph reductions in which the process that may complete their exe?
- If a processes's resource request may be granted, then we can say that a graph may be reduced by that process.
- We reduce a graph by removing arrows
  (1) from resource to process
  (2) from process to resource.
- If a graph can be reduced by all its process then there is no deadlock

1)

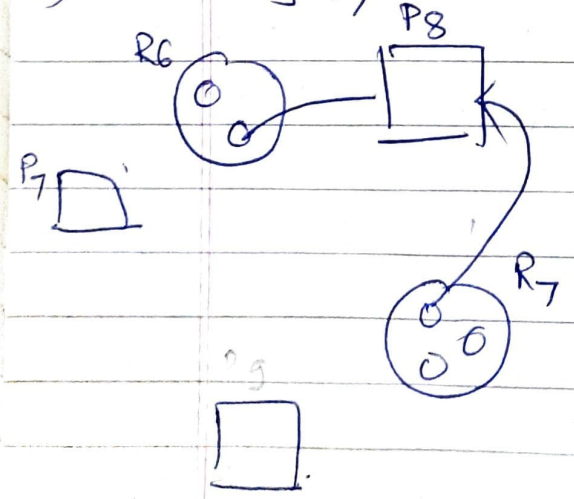


← original

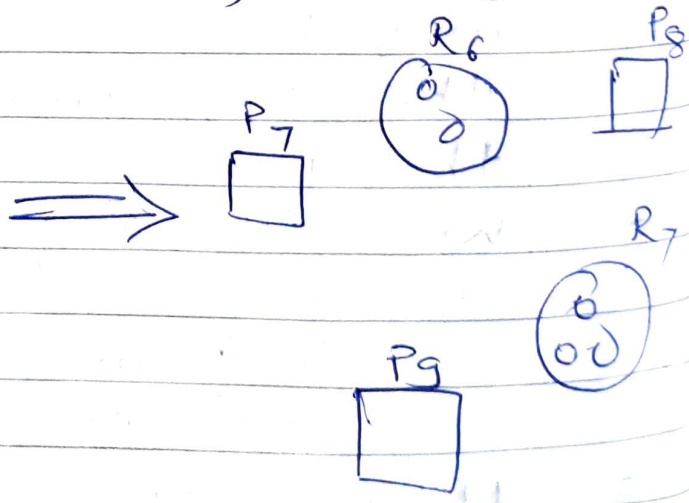
2) Reducing by P9



3) Reducing P7



4) Reducing P8



// Multiple resources of same type are represented by nested circles

7] * Deadlock Recovery

- After detection of deadlock, there are two possibilities for breaking of deadlock
  1) To abort one/more process to break circular wait
  2) To preempt some resources from one or more of the deadlocked process.

1) Process Termination
- To eliminate deadlocks by aborting process, the sys reclaims all resources allocated to the terminated process.

(A) Abort all deadlocked process—
This will clearly break deadlock cycle

(B) Abort one process at a time until deadlock cycle is eliminated.

- For this many factors may affect which process is chosen including
- Process may be removed by —— considering Priority of process, Suspend Process partially, considering remaining time of process, How many Resources are hold by Process etc.

PAGE NO:
DATE:

2) Resource Pre-emption

- Here we successively preempt some resources from processess and give these resources to other processes until the deadlock cycle is broken.
- If preemption is required to deal with deadlocks then three issues need to be address.

(A) Selecting a victim :- Select which resource and which process are to be pre-empted.

(B) Rollback :- If we preempt the resource what should be done with that process? Abort the process and then restart it. Also one more effective to roll-back the process only as far as necessary to break the deadlock.

(C) Starvation :- we should ensure that same resource should not be pre-empted always for same process