UNIT-VI

6.0 Introduction

- This chapter mainly focuses on deadlock definition, their situation and graphical representation.
- This chapter consists of different methods of deadlock prevention, avoidance and detection.
- In this chapter we also focuses on Resource allocation Graph.
- The most important deadlock avoidance "Bankers" algorithm is discussed.
- It also includes the concept of "recovery".
- This chapter introduces protection Goal, method of protection.
- Security goal cryptography, their technique and threats definitions.
- At the end of chapter we will came across the various numerical on deadlock avoidance with three famous algorithm.

  i.e. Banker, safety, and request response.

6. Deadlock

Q.: What is dead lock. **[W-12, 2M]**

Q.: Dead lock system model. General Definition.

6.1 Background

- A set of all processes is dead locked if each process in the set is waiting for an event that only another process in the set can cause.
- Because all the process are waiting, none of them will ever cause any of the event that could wake up any of the other member of the set, and all the processes continue to wait forever.
- For this model, we assume that processes have only a single thread and that there are no interrupts possible to wake up a blocked process.
- The no-interrupts condition is needed to prevent, otherwise deadlocked process from being awakened by, say an alarm and then causing events that release other process in the set.

- The event that each process is waiting for the release of same resource currently possessed by another member of set.

  In another words, each member of the set of deadlock process is waiting for a resume that is owned by a deadlocked process.
- None of the process can run, none of them can release any resource, and more than can be awakened.
- In multiprogramming environment we need to share some resources even several process are may complete for a finite number of resources, but generally resources are not available that time, then process enters in waiting state, sometimes it never come back,
- Every resources are consist of (Mem CPU, % etc) identical resource instances which leads to satisfied process request.
- If process request an instance (Resource) type the allocation of any instance of the type will satisfy the request.
- So, simply Deadlock is a situation when two or more process get locked and cannot proceed further because of process dependency.
- Deadlock can happen for I/O media or shared resources such as internal table maintained by OS.

Real time Definition

1. **Road:** consider four way crossing in which every user want to cross that road any how. here road is a shared resource to which every user wanted to use and not allowing to give to others then deadlock can occurs.
2. **Population:** The another example of deadlock is a population day to day population is growing rapidly in which every people want to proceed further (i.e. they compete to each others).

   Same solution of dead lock are:

   1. For Road make fly over.
   2. Use of crane
   3. Use of signals (and many more)

   Normal operation on resource

   In order to use any resource uses must follow this convention.

1. **Request:** In order to use resource by any user need to make a request, if resource is free then it can allocate to user as per request priority other wise user have to wait till resource became free by other user.
2. **Use of resource:** After allocating resource to user it need to be used by the user otherwise no sense to allocating the resource.
3. **Release:** After execution of any process on resource it need to release for other user.

   Note: By this three step we can achieve safe deadlock handling.

6.2 Characteristics of Deadlock

Q.: What are the different characteristic of deadlock.

Q.: What are the deadlock prerequisites? (W-12, 6M/W-09, 5M)

- What cause a deadlock? Coffman and shoshani in 1971 have shown that there are four condition all of which must be satisfied for a deadlock to take place.
- A deadlock situation can arise if the following four conditions hold simultaneously in a system.

1. **Mutual exclusion:** At least one resource must be held in a non-sharable mode, that is only one process at time can use the resource.

- If another process request that resource, the requesting process must be delayed until the resource has been released.
- For instance disk drive can be shared by two processes simultaneously. This will not cause a deadlock, But printer, tape drivers, plotters etc. have to be allocated to a process in an exclusive manner. Control the process completely finished its work with it. This leads to trouble.

2. **Hold and wait:** There must exist a process that is holding at least one resource and is waiting to acquire additional resources that are currently being held by other processes.
3. **Non-preemption:** Resource cannot be preempted, i.e. resource can be release only voluntarily by the process holding it. After that process has completed task.

- It means if a process hold certain, resource, no other process should be able to take away from it. Only the process holding then should be able to release then explicitly.

4. **Circular wait:** There must be a set $\{P_0, P_1 - - - P_n\}$ of waiting process such that P_0 is waiting for a resource that is held by P_1, P_1 is waiting for a resource that is held by P_2 --- P_{n-1} is waiting for resource that is held by Pn an Pn is waiting for a resource that is held by P_0.

6.3 Resource Allocation Graph

Q.: Explain the resource allocation Graph algorithm. [W-09, 5M, W-10, 5M]

Q.: Short note on Resource allocation Graph. [W-11, 6M]

- RAG (Resource allocation Graph)
- Deadlock can be described more precisely in terms of a directed group called a system resource allocation graph.
- The graph that shows process and resource relation of their uses.
- As a general graph a RAG is also consist of set of vertices and set of edges (i.e. V, E).
- Where a set of vertices are partitioned into two sets.

i.e. $P = \{P_1 P_2 ..... P_n\}$ is a set of all process and

$R = \{P_1 P_2 --- R_n\}$ is a set of all Resources.

– And set of edges are consist of

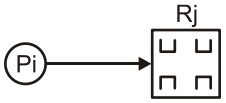
e: {request edge, Assigned edge}

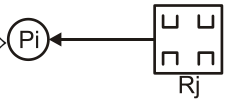
i.e. if $P_i \rightarrow R_j$ (called request edge by process to any resource) and

if $R_j \rightarrow P_j$ (called Assignment edge by resource to any process)

Entities:

In order to represent the RAG we required some Entity

1. Process ⇒ ◯
2. Resource instance (4 instance) ⇒ 
3. Pi request instance of R_j ⇒ 

4. Pi is holding an instance of Rj ⇒ 

Consider are example of Graph.

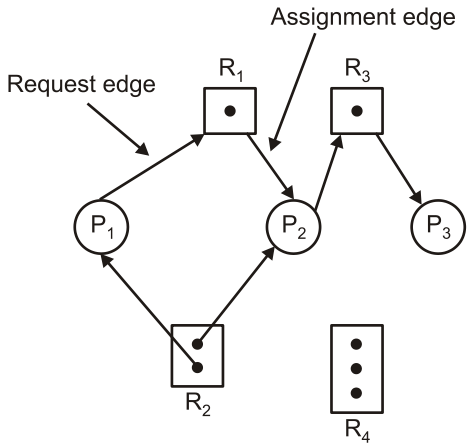


Fig.: RAG

- Here, the Set of P, R and E are

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

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

$E = \{P_1 \rightarrow R_1, R_2 \rightarrow R_3, R_1 \rightarrow P_2, R_2 \rightarrow P_2, R_2 \rightarrow P_1, R_3 \rightarrow P_3\}$

- Resource instances are:
  - one instance of resource type R_1
  - Two instance of resource type R_2
  - One instance of resource type R_3
  - Three instance of resource type R_4

Process State

- Process P_1 is holding an instance of resource type R_2 and is waiting for an instance of resource type R_1.
- Process P_2 is holding an instance of R_1 and instance of R_2 and is waiting for an instance of R_3.
- Process P_3 is holding an instance of R_3.

  Now, use of RAG is to show dead lock.

- Some facts of RAG Deadlock are

  1. If Graph contains no cycle → No deadlock in system.
  2. If Graph contains a cycle so.

  i) If only one instance per resource type then deadlock is possible.

  ii) If several instance per resource type then possibility of deadlock is may or may not be present [if that process is not waiting for other resource instance].

- Take an example of cycled Graph

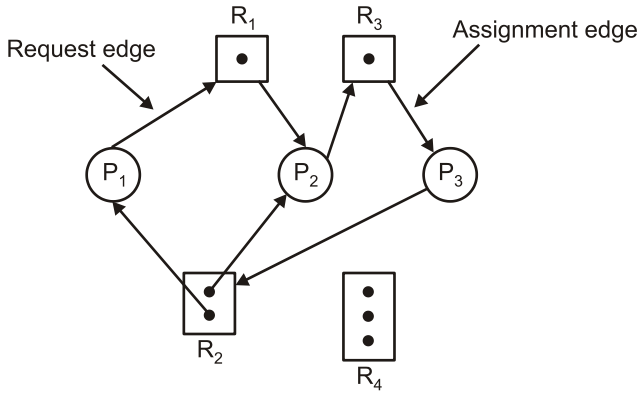


Fig.: RAG with a dead lock

Description

- In above diagram (Graph) there are some cycle presents i.e.

  $P_1 \rightarrow R_1 \rightarrow P_2 \rightarrow R_3 \rightarrow P_3 \rightarrow R_2 \rightarrow P_1$

  $P_2 \rightarrow R_3 \rightarrow P_3 \rightarrow R_2 \rightarrow P_2$

  At this point we can say that process P_1, P_2 and and P_3 are deadlock because they are hold same devices and waiting for another which are held by another process. They will never get that resource.

Take as example of cycled Graph without deadlock.

Description:

- Now, see a RAG which has one cycle i.e.

  $P_1 \rightarrow R_1 \rightarrow P_3 \rightarrow R_2 \rightarrow P_1$

- However there is no deadlock How?
- Consider a process P_4 may release it's instance of resource type R_2. That resource can then be allocated to P_3 and cycle can break.

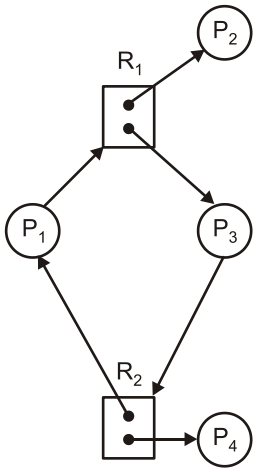
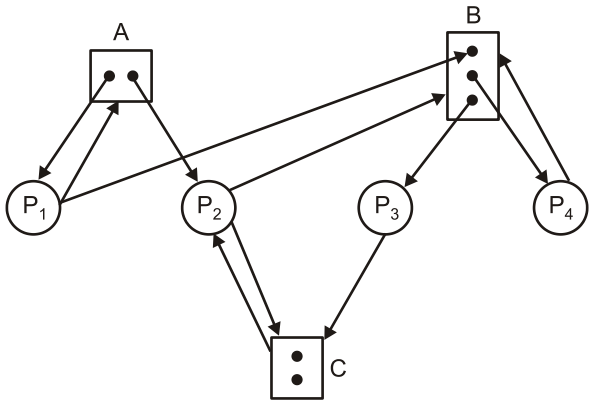


Fig. RAG with cycle but no deadlock

- It is only, possible because P_4 does not wait any other resource instance.
- So we can say that even if cycle is present in RAG but their is no dead lock.

Q.: Consider the following RAG. **[S-11, 5M]**

1. **State the situation when the system is in a deadlock state.**
2. **Otherwise find a safe sequence.**



1. Resource allocation graph with a deadlock.

   $P_1 \rightarrow A \rightarrow P_2 \rightarrow B \rightarrow P_1$

   $P_2 \rightarrow B \rightarrow P_3 \rightarrow C \rightarrow P_2$

   $P_1 \rightarrow A \rightarrow P_1$

   $P_2 \rightarrow C \rightarrow P_2$

   $P_4 \rightarrow B \rightarrow P_4$

2. Safe sequence.

(i) Graph with a cycle but no deadlock

$P_1 \rightarrow A \rightarrow P_2 \rightarrow B \rightarrow P_1$

No dead lock because P_4 and P_3 may release its instance of resource B, then it can be allocated to P_2.

(ii) $P_2 \rightarrow B \rightarrow P_3 \rightarrow C \rightarrow P_2$

No deadlock, because P_4 may release its instance of resource B and then it can be allocated to P_3.

Deadlock Handling

1. Deadlock can be handle by numbers of way.

– Prevent the system from hanging when two or more processes are independent upon each other.

– Deadlock are a challenging problem to correct as they result in data loss are difficult to

isolate, create unexpected problem and time consume to fix.

– For deadlock handling most of current OS cannot prevent a deadlock from occurring.

– When a deadlock occurs, different OS respond to them in different non-standard manners. Most approaches work by preventing one of the four coffman condition from occurring, especially the fourth one.

The three main consideration to deadlock handling that are as follows:

1. We can use a protocol to prevent or avoid deadlock, ensuring that the system will never enter into a deadlock state.
2. We can allow the system to enter into a deadlock state, detect it and recover it.
3. We can also ignore the problem altogether, and pretend that dead lock never occurs in the system.

• Above three way can also be subclassified in deadlock Ignorance, Detection, Recover, Prevent and Avoid.

– Deadlock Ignore and Recover strategies are not that much effective, our main concern is all about deadlock prevention, avoidance and deadlock detection.

6.4.1 Deadlock Prevention

Q.: How can you prevent deadlock by preventing circular wait condition. [W-08, 5M]

Q.: Explain the protocol to be used to break the circular wait condition in deadlock. [S-12, 3M]

• As we seen all four condition (ME, No-preemption, circular wait and Hold and wait) are necessary for deadlock to occur, if follows that deadlock might be prevented by denying any one of the condition.

• Deadlock prevention is the process of making it logically impossible for one of the 4 deadlock condition to hold.

• The prevention goal is to ensure that at least one of the necessary condition for deadlock can never hold.

• Preventing deadlock by constraining how request for resource can be made in the system and how they are handled (system design).

1. ME (Mutual Exclusion)

– As we all know that the definition of ME in which only one activity can possible at a time.

– In order to prevent a deadlock the ME can not be guarantee because not every resources are shared resource, generally almost all resources are non-shared mode.

– Consider printer as a resource which state can not be capture, printer (state) is not a shared resource.

- Once we gave command to print we can not control/save their state so we can not say that deadlock can be preempted using printer.
- Now, Take another example of CPU which is inherited shared device or resource. On CPU we can save and shared their state and can be possible to execute other processes.
- But the drawback of CPU sharing and saving their state its too difficult as we seen in time sharing OS.
- So, Generally we can not prevent deadlock by just denying the mutual exclusion condition because same resource are intrinsically non-sharable.
- Finally we can say that, we can not eliminate the ME.
- Note that sharable resource like read only file do not require ME access and thus cannot be involved in deadlock.

2. Hold and wait

- We must guarantees that whenever a process request a resource it does not hold by any other resource.
- It means if any process has a hold of one resource so don't allow to wait for other resource.
- In order to achieve above criteria we need to allocate all resources to process at initial state.
- Otherwise allow a process to request resource only when the process has not any resource with itself.
- But if we assign resource at initial stage so it leads to some drawback i.e.

  1. Less resource utilization because if we allocate all resource to process so they may not be useful to produce O/P by other process.

  2. Second problem is starvation because if any process need resource which is holded by other process so they have to wait infinite time.

3. No-preemption

- If a process that is holding some resource, then another request cannot be immediately served.
- Preempted resource are added to the list of resource for which the process is waiting.
- Process will be restarted only when it can regain its old resource as well as the new ones that it is requesting.
- Otherwise if a process request some resources we first check whether they are available, if they are, we can allocate them. If they are not available we check whether they are allocated to some other process that is waiting for additional resources.

4. Circular wait

- "Impose a total ordering of all resource types, and require that each process request resources in an increasing order of enumeration".
- Let R = $\{R_1, R_2 ..... R_n\}$ is a set of resource that, and assign a unique number, which can be

useful while comparison.

- Consider an example with tape drive, disk and printer with 2, 5, and 12 as a identification number respectively.
- So at the time of assignment to any process to printer it needed to hold type drive but it can not request because the identification number of printer is 12 and tape drive is 5 so it can not.
- Where as if any process hold tape drive and want to hold printer so it can because the identification number of printer is 12 and tape drive is 5. So the number of printer is high than tape drive [As per the 1st statement of order enumeration].

6.4.2 Deadlock Avoidance

Q. Explain Deadlock Avoidance. [S-11, 6M]

Ans.: Deadlock avoidance means, make sure that systems never enter into unsafe state.

- Require that the system has some additional a prior information available (like all resource value information, type, used, etc.).
- Simplest and most useful model requires that each process declares the maximum number of resource of each type that it may need.
- A deadlock avoidance also dynamically examine the RAG state to ensure that there can never be a circular wait condition.
- RAG state is defined by the number of available and allocated resource and the maximum demands of the process.

Q.: What is safe state and state sequence?

- A state is safe if the system can allocate resource to each process in some order and still avoid a dead lock.
- System is in safe state if there exist a safe sequence of all processes.
- Safe sequence is a sequence which show or indicate the order of process execution, its help to keep system in safe state.
- There may be a possibility to have more than one safe sequence in system.

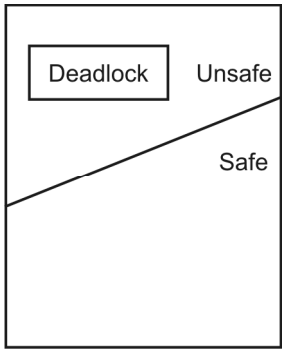


Fig.: Safe, Unsafe and Deadlock State Space

- **Unsafe State:** is a state which leads to dead lock.
- Basic Facts are :
  1. If a system in safe state so there is no deadlock.
  2. If system in unsafe state, then there is possibility to have deadlock in system.
- IF system generate a safe sequence if means. System is in safe state.
- Example of safe sequence, consider 12 tape drives and 3 processes and current system state is.

Table:

Process	Max Need	Allocated
P0	10	5
P1	4	2
P2	9	2

Description: process P_0 required 10 tape drives and it has allocated 5 drives, P_1 required 4 and it has 2 allocated and P_2 required 9 and allocated 2 drives.

- Now total allocation of drives are 9 (5 + 2 + 2) and initially it was 12 available so allocated-available (9 – 12) = 3.
- 3 Type drives are available to satisfied process request.
- Above system is safe state the answer is yes because it having safe sequence $<P_1, P_0, P_2>$
- From available 3 type driven we can satisfy process P_1 request because it required only 2 tape drives after execution of P_1 it release 2 Tap drives is became 5 available and available 5 can serve the request of P_0 and P_2.
- So here is a $<P_1, P_0, P_2>$ is a safe sequence so we can say given system is in safe state and deadlock free.

How RAG can used to Deadlock avoidance?

- RAG can used to deadlock avoidance in single instance of resource type.
- In this algorithm claim edge: When process is $P_i \rightarrow Rj$ indicated that process P_j may request resource R_j, and that represented by dashed line.
- Claim edge converts to request edge when a process request a resource.
- Request edge converted to an assignment edge when the resource is allocated to the process.

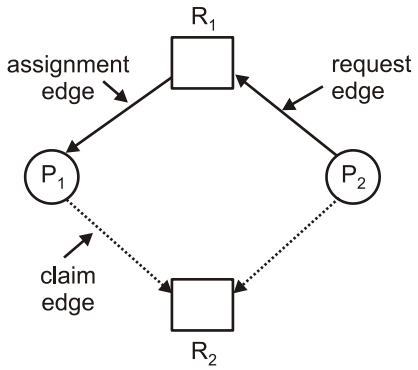


Fig.: RAG for deadlock avoidance

Here:

$P_1 \rightarrow R_1$ is request edge

$R_1 \rightarrow P_1$ is Assignment edge

$P_1 \rightarrow R_2$ is claim edge and

$P_2 \rightarrow R_2 \rightarrow$ is also claim edge

- Actually claim edge is used for future benefit purpose.

– When resource is release by a process claim edge is converts into request edge and assignment edges as well.

– Here, claim edge $P_2 \rightarrow R_2$ is now converted into request edge immediately it assigned to that process as $R_2 \rightarrow P_2$.

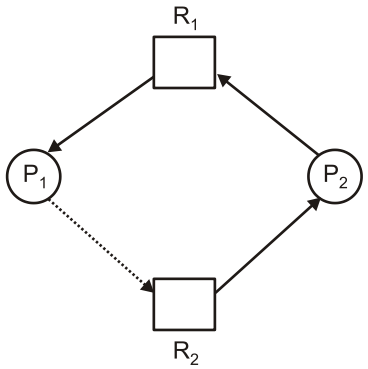


Fig.: After conversion of edge (Converted RAG)

– If a cycle so that edge does not converts.

Q.: What is Bankers Algorithm. [S-09, 3M] [W–11, 7M]

- Bankers algorithm used for multiple instance of resource.
- Because RAG is not applicable for multiple instance type.
- When a new process enter the system, it must declare the maximum number of each resource type that it is needed. This number may not exceed the total number of resource in the system.

- Because of multiple instance, banker algorithm required more data structure that are as follows, in which n is it the number of processes in the system and m is the number of resource type.

1. **Available:** available vector indicate how many available resource instance are available.

– A vector length in indicates the number of available resource of each type

If Available [j] equal K, then K instance of resource type R_j are available.

2. **Max:** Max vector indicates how many instances are required.

– An M × n matrix define the maximum demand of each process if Max [i] [j] equal K, then process Pi may request at most K instance of resource type R_j

3. **Allocation:** allocation matrix indicates the allocated resume instances.

– An n×m matrix defines the number of resource of each type currently allocated to each process. if Allocation [i] [j] equal k, then process P_1 is currently allocated K instance of resource type R_j.

4. **Need:** An n × m matrix indicates the remaining resource need of each process. If need [i][j] equal K. Then process P_j may need K more instance of resource type R_j to complete its task. Note that Need [i] [j] equal Max [i] [j] – Allocation [i] [j].

Q.: What is safety Algorithm [S–09, 3M]

- While execution of banker algorithm we need to call safety algorithm, to check whether system is in safe state or not.
- **Algo:**

(1) Let work and finish be vector of length m and n, respectively Initialise work = Available and finish [i] = false for i = 0, i .... n - 1

(2) Find on index i such that both

a. finish [i] = = false

b. Need[i] ≤ work

if no such i exist, go to step 4

(3) work = work + Allocation i

finish [i] = true

Go to step = 2

(4) If finish [i] = true for all ; then the system is in a safe state.

What is Resource Request Algorithm

- This algorithm determining whether request can be safety granted or not.

Algorithm

let Request i be the request vector for process P_i.

If Request [j] = K, then

process P_i want K instance of resource type R_j,

when a request for resources is made by process P_i, then following action are taken.

1. If Request [i] ≤ Need[i] go to step 2, otherwise, raise an non condition, since the process has exceeded its maximum claim.
2. It Request ≤ Available, go to step 3, otherwise Pi must wait, since the resource are not available.
3. System pretend to have allocated the request resource to process Pi by modifying the state as follows.

   Available = Available – Request

   Allocation = Allocation + Request

   Need[i] = Need[i] – Request[i]

- if safe = the resource are allocated to P_i
- if unsafe = P_i must wait, and the old resource-allocation state is restored.

Q.: Examples of Banker algorithm. **[W-08, 8M]**

- Consider a system with the processes P_0 through P_4 and three instance type A, B and C, Resource type A has 10 instance, B has 5 instance and C has 7 instance and suppose at the time t1 following snap shot is.

Process	Allocation	Max	Available
	ABC	ABC	ABC
P_0	0 1 0	753	332
P_1	200	322	
P_2	302	902	
P_3	211	222	
P_4	0 0 2	433	

Step 1 calculate need matrix content as

Need = Max - allocation

Process	Need		
	A	B	C
P_0	7	4	3
P_1	1	2	2
P_2	6	0	0
P_3	0	1	1
P_4	4	3	1

Step 2 update Need matrix with Allocation and work matrix (work and available same).

	Allocation			Need			Available/work		
	A	B	C	A	B	C	A	B	C
P_0	0	1	0	7	4	3	3	3	2
P_1	2	0	0	1	2	2			
P_2	3	0	2	6	0	0			
P_3	2	1	1	0	1	1			
P_4	0	0	2	4	3	1			

Step 3

Now, check for satisfaction and calculation of safe sequence.

i) check for process P_0;

whether, Po Need [i] ≤ Available [i]

i.e. 743 ≤ 3 3 2

No. Jump to next process

ii) Check for P_1:

i.e. 1 2 2 ≤ 3 3 2

Yes, P_1 can finish their execution

So, Add P_1 into safe sequence and

update available matrix

Safe sequence of process $<P_i>$

update matrix i.e. Allocation of Pi + work

∴ 200 + 332

= 532

∴ New work available is 532

iii) Check for P_2 Need matrix with updated work matrix contents

i.e. $600 \leq 532$

Ans = No, jump to next process

iv) check for P_3

i.e. $011 \leq 532$

Ans = Yes, so add this process to safe sequence

i.e. safe sequence processor $<P_1, P_3>$

and update work matrix with P_3 allocation

i.e. $211 + 532$

$= 743$

∴ New work matrix is 743

v) Check for P_4

i.e. $431 \leq 743$

Ans. Yes, so Add this process to safe sequence

i.e. Safe sequence processes $<P_1, P_3, P_4>$

and update work matrix value with P_4 allocation

i.e. $= 0\ 0\ 2 + 743$

$= 745$

∴ New work matrix is 745

vi) check for remaining process i.e. P_0

i.e. $= 743 \leq 745$

Ans. = Yes, so add this process to safe sequence.

i.e. safe sequence processors $<P_1, P_3, P_4, P_0>$

and update work matrix value with P_0 allocation.

i.e. $= 0.10 + 745$

$= 755$

∴ New work matrix value is 755

viii) Now last check for process P_2

i.e. $600 \leq 755$

Ans. : Yes, so add this to state sequence

i.e. safe sequence processes <P_1, P_3, P_4, P_0, P_2>

- Final safe sequence is <P_1, P_3, P_4, P_0, P_2>
- If safe sequence generated so there is no any deadlock and hence system is deadlock free.

Numerical and bankers Algorithm

Q.: Consider th following snapshot. **[S-09,5M]**

Process	Allocation			Max		
	A	B	C	A	B	C
P_0	0	0	1	1	1	1
P_1	2	0	0	3	2	3
P_2	1	3	2	4	3	1
P_3	1	0	1	0	0	1
P_4	0	0	1	3	2	1

Let A, B, C be resource with instance of A is 7, B is 3 and C has 6 instance.

Now 1. What is the content of the NEED matrix?

2. Whether system is in safe state or not ?

Solution: 1. Calculation of Need matrix

Need = Max – Allocation

so, we will get the matrix is

	A	B	C
P_0	1	1	0
P_1	1	2	3
P_2	3	0	1
P_3	0	0	0
P_4	3	2	0

2. Check system is in safe state or not.

Solution: In order to calculate the safe sequence we need available matrix value that is not given but total instances valve is given so we can easily calculate the available matrix content.

i.e. A = 7, B = 3 and C = 6 instances

So we can simply calculate available matrix valve by minusing all allocated instances.

* Total allocation of A instance by each processes (P_0 ---- P_4) is 4.
* So allocation – Total instance = available

  i.e. available = Total instances – allocation

  $\therefore$ 7 – 4 = 3
* Similar for the B instance is

  available = 3 – 3 = 0
* Similar for the C instance is

  available = 6 – 5 = 1
* Finally available is A = 3, B = 0, C = 1

  Final snapshot on which we can calculate safe sequence is

Process	Allocation			Need			Available		
	A	B	C	A	B	C	A	B	C
P_0	0	0	1	1	1	0	3	0	1
P_1	2	0	0	1	2	3			
P_2	1	3	2	3	0	1			
P_3	1	0	1	0	0	0			
P_4	0	0	1	3	2	0			

Now, apply the standard procedure to calculate safe sequence we will get

$\therefore$ safe sequence is $<P_2, P_3, P_4, P_0, P_1>$

$\therefore$ System is in safe state.

Q.: It is proposed to use a Bankers algorithm for handling deadlock. The total number of resources available for allocation is 7, 7 and 10 respectively.

The current resource allocation state is shown as given below. (S-12)

Process	Allocation			Max			Available		
	R_1	R_2	R_3	R_1	R_2	R_3	R_1	R_2	R_3
P_1	2	2	3	3	6	8	7	7	10
P_2	2	0	3	4	3	3			
P_3	1	2	4	3	4	4			

1. **Is the current allocation state safe ?**
2. **Would the following request be granted in the current state?**

   P_1 request (1, 1, 0)

Solution: Safe state given system having safe state because if generate a safe sequence.

∴ By applying the standard procedure of banker algorithm.

consider, 7, 7 , 10 is available matrix, it can satisfied the request.

Calculate Need matrix

Need = Max – allocation

Process	Allocation			Need			available		
	R_1	R_2	R_3	R_1	R_2	R_3	R_1	R_2	R_3
P_0	2	2	3	1	4	5	7	7	10
P_1	2	0	3	2	3	0			
P_2	1	2	4	2	2	0			

1. Check for process P_0 need is (1, 4, 5) is less than available so add is to safe sequence.
2. Check for process P_1 Need is (2, 3, 0) is less than available so add it to the safe sequence.
3. Check for process P_2 also satisfied so add into safe sequence.

So, Final safe sequence is $<P_1, P_2, P_3>$

hence given system is in safe state.

2. Would following request be granted P_1 request <1, 1, 0>

Ans.: Yes, because the Need of process P_1 is <1, 1, 0> is less than available matrix.

Q.: Consider the following snapshot of a system. [S-13, 8 M]

Process	Allocation				Max				Available			
	A	B	C	D	A	B	C	D	A	B	C	D
P_0	0	0	1	2	0	0	1	2	1	5	2	0
P_1	1	0	0	0	1	7	5	0				
P_2	1	3	5	4	2	3	5	6				
P_3	0	6	3	2	0	6	5	2				
P_4	0	0	1	4	0	6	5	6				

Calculate 1 Content of Need matrix

Need = Max – Allocation

So, final table is

Process	Allocation				Max				Available			
	A	B	C	D	A	B	C	D	A	B	C	D
P_0	0	0	1	2	0	0	0	0	1	5	2	0
P_1	1	0	0	0	0	7	5	0				
P_2	1	3	5	4	1	0	0	2				
P_3	0	6	3	2	0	0	2	0				
P_4	0	0	1	4	0	6	4	2				

(ii) Is the system in safe state prove it.

Solution: To say system in safe state we need to calculate the safe sequence.

$\therefore$ Apply standard procedure of banker algo for each process.

(i) Check P_0 Need $\leq$ available.

$\therefore$ 0 0 0 0 $\leq$ 1520

Ans = Yes

$\therefore$ Update available matrix and add process to safe sequence.

New available = allocation + available

i.e. 0 0 1 2 + 1 5 2 0

available = 1 5 3 2

(ii) Check for process P_1

i.e. 1750 $\leq$ 1 5 3 2

Ans. No, Jump to Next process

(iii) Check for process P_2

i.e. 1 0 0 2 $\leq$ 1 5 3 2

Ans.= Yes, so update available matrix and add to safe sequence.

new available = allocation + available.

i.e. allocation of P_2 + available

$\therefore$ 1 3 5 4 + 1 5 3 2

= 2886

(iv) check for process P_3

i.e. 0 0 2 0< 2 8 8 6

Ans. : Yes, So update available matrix and add process to safe sequence.

i.e. 0 6 3 2 + 2 8 8 6

= 2 14 11 8

(v) Check for process P_4

i.e. 0 6 4 2 ≤ 2 8 8 6

= 2 14 11 8

Ans.: Yes, so update available matrix.

i.e. 0 0 0 1 4 + 2 14 11 8

= 2 14 1 2 12

(vi) Check for process P_1

i.e. 1750 ≤ 2 14 12 12

Ans. Yes, so update available matrix A add process to safe sequence.

∴ 0 0 12 + 2, 14 12, 12 = 2 14 13 14

(vii) So final safe sequence is

$<P_0, P_2, P_3, P_4, P_1>$

so, finally system is in safe state.

(iii) If a request from process P_1 arrives for (0, 4, 2, 0) can the request be granted immediately and Why?

Solution: When request (0, 4, 2, 0) comes from P_1 then check the request ≤ available. (Here we are considering the final available matrix value i.e. 2 14 13 14)

0, 4 2 0 ≤ 2 14 13 14

∴ yes it can be granted

and if we consider the I[st] available matrix ∴ i.e. 1520

then 04 20 ≤ 15 20

it also can be satisfied

so, finally in both cause it is possible to grant the request from P_1.

6.4.3 Deadlock Recovery

Q.: What is deadlock recovery explain in detail?

- When a detection algorithm determine that a deadlock exist, several alternative are available.
- One possibility is to inform the operator that a deadlock has occurred and let the operator deal with the deadlock manually.
- Another possibility is let the system recover from the deadlock automatically.

There are two options for breaking a deadlock.

1. One is simple to abort one or more processes to break the circular wait.
2. The other is to preempt some resource from one or more deadlock process.

– Above to solution are categories into two types.

(A) Process Termination

(B) Resource Preemption

(A) Process Termination

In this step same of the possibilities are possible that are as follows:

(i) Abort all, deadlock processes- In process termination the abort of process execution for deadlock recovery is possible because if we abort all running processes so the resources which holded by running process made available to others.

(ii) Abort one process at a time: Releasing resources until no deadlock.

– How do we determine which process to about first?

– Priority ordering process which has done least work.

(iii) Selectively restart

– The processes from a previous check point i.e. before it claimed any resource difficult to achieve sometimes impossible.

– Check points are the mechanism which can help us to restart the process from previous normal stage.

(iv) Successively withdraw resources

– From a process and give a another process until deadlock is broken.

– How to choose which process and which resources?

To do above active we need some of the skills.

1. Complex decision due to large number of process present within a system.
2. Difficult to automate
3. Use operator to resolve conflicts, but this requires the operator to have skill and understanding of what process are actually doing.

(B) Resources Preemption:

Note: Separate question is possible and asked

Q.: What are the three issues needed to be address when pre-emption is required to deal with deadlock. [W-10, 6M]

– In order to recover the deadlock resource preemption is another way/method. In which some issues are needed and is to be considered while recovery.

– List of issues

1. Selecting a victim
2. Roll back
3. Starvation.

Selecting a victim

- In resource preemption we need to find a resource/process suffered from deadlock.
- Now select a proper process which is to be preempted.
- As in process termination we must determine the order of preemption to minimize cost.

ii) Roll back

- After preempts a resource from process, what should be done with that process?
- It can not continue with its normal executions. It is missing some needed resource. We must roll back the process to some safe state and restart it from that state.
- However, it is more effective to roll back the process only as for a necessary to break the deadlock. On the other hand, this method requires the system to keep more information about the state of all the running processes.
- Rollback is an operation which returns the database to same previous state.
- Simply a rollback is operation removes all changes made since the previous commit or rollback operation.

iii) Starvation

- How do we ensure that starvation will not occurs?
- i.e. How we can give guarantee that resource will not always be preempted from the same process.
- Starvation is a resource management problem where a process does not get the resource it needs for a long time because the resource are being allocated to other process.

Note: There are above issues are need to be address while deadlock recovery.

6.5 Security

- Security is the degree of resistance to, or protection from, harm.
- It applies to any vulnerable and valuable asset, such as a person, dwelling, community, nation or organisation.
- Definition of system security control of access to a computer system resources, specially its data and os itself.

Q. Explain the Goal of protection. [W-11, 3M]

- A computer system have became more sophisticated and pervasive in their application and operation, that need to be protect their integration.

- So we need to provide protection for several vulnerable reasons.
- It ensure that all resource are accessed in a correct way and only by those who are authorized.
- Protection refer to a mechanism for controlling the access of program, process or user to the resources define by the computer system.
- The most obvious is the need to prevent mischievous, intentional violation by a user.
- Protection can improve reliability by detaching latent error at the interferes between component subsystem.
- The role of protection in a computer system is to provide a mechanism for the enforcement of the policies governing resource uses
- Obviously to prevent malicious misuse of the system by user programs.
- To ensure that each shared resource is used only in accordance with system policies which may be set either by system designer or by system administrators.

6.5.1 Access Matrix

Q.: What is access matrix.

- Access matrix is an array of cells with row's and columns.
- The rows are for per subject and columns are for object.
- Each column is equivalent to access matrix (control) list for the object and each row is equivalent to an access profile for the subject.
- The access control matrix can be used as a model of the static access permission in any type of access control system.

  Simply access matrix is an abstract model of permission.
- Access matrix show the protection level across several domain.
- Concrete definition of access matrix is a method of representing discretionary authorization information, with rows representing subject or user of system, columns corresponding to objects or resources of the system and cells composed of allowable operation that a subject may apply to an object.
- **Domain structure**
- A protection domain specifies the resources that a process may access.
- Each domain defines a set of object and the type of operation that may be invoked on each object.
- An access eights is the ability to execute an operation on an object.
- A domain is define as a set of <object, {access right set}> pair

– Consider a Table matrix there exist two processes a file and a device the 1st process has the ability to execute. The second have read the file and write same information to the device. While the second process can any send information to the first.

	Asset 1	Asset 2	File	Device
Role 1	R/W E/0	E/R	R	W
Role 2	R	R/W/E0		

– Operation Access matrix.

(1) Copy (2) Owner (3) Control.

1. Copy

– The ability to copy right's it denoted by an asterisk, indicating that processes in that domain have the right to copy that access within the same column, i.e. for the same object.

2. Owner

– The owner right adds the privilege of adding new right remaining existing.

3. Control

– Copy and owner rights only allow the modification of rights within a column. The additional control rights, which only apply to domain object, allow process operating in one domain to affect the rights available in other domain.

Access matrix Implementation

Q.: Write a short note on access matrix implementation.[S-13, 5M/W-12, 3M].

There are five way to implement access matrix these are as follows.

(A) Global Table.

– The simplest approach is one big global table with <domain object, right> entries.

– Unfortunately this table is very large and so cannot be kept in memory.

– There is also no good way to specify grouping, if everyone has access to same resources, than it still needs a separate entry for every domain.

(B) Access list for object

– Each column of the table can be kept as a list of the access rights for that particular object discarding blank entries.

– For efficiency a separate list of default access rights can also be kept and checked first.

(C) Capability list for domains

– In a similar fashion each row the table can be kept as a list the capabilities of that domain.

– Capability list are associated with each domain, but not directly accessible by the domain or any user process.

– Capability list are themselves is a protected resources.

(D) A lock–key mechanism

- Each resource has a list of unique bit pattern, termed as lock.
- Each domain has its own list of unique bit pattern, terms as keys.
- Access is granted if one of the domain's key fits one of the resource locks.
- Again a process is not allowed to modify its own keys.

(E) Comparison

- Each of the methods here has certain advantages or disadvantages depending on the particular situation and task at hand.
- Many system employ some combination of the listed methods.

Q.: What are the differences between ACCESS and CAPABILITY list.

[S-09, 8M S-9 8M. S11, 8M/ S12, 9M]

ACL (Access list control)	CAPABILITY LIST
1. An ACL is a list of permissions attached to an object. An ACL specifies which user or system processes are granted access to objects.	1. Capability list is a design of secure computing system. A capability is a communicate, unforgeable taken by authority.
2. ACL having at least five namespaces – File name – Unique identifier of device – User identifier – Operation on object – Identification of process.	2. Capability having four namespaces – unique object identifier – operation an object – process identifier – namespace of capacity.
3. When computer shutdown and all the processes disappears. Then in ACL this is not a problem because the login session disappears too.	3. But in capability list there is definite problem.
4. ACL allow courser grain protection.	4. Capability list allow finer grain protection.
5. In all we can remove a user from the list.	5. In capability list we can not remove a user from the list.
6. All does not allow rights transfer.	6. In capability list capabilities can be transferred.

6.5.2 Security Goal's

Q. What is security and its goals. [W-08, 4M]

- Security is important aspect of a operating system.
- In General, secure system will control, through use of specific security feature, access to information that only properly authorized individuals or processes operating on their behalfs

will have access to read, write, create or delete.

- Operating system security is the process of ensuring OS integrity confidentially and availability.
- OS security encompasses all preventive-control technique, which safe guard any computer assets capable of being stolen edited or detected if OS security is compromised.

Goals

- Provides proper and unique access to each other.
- Provides integrity and non-repudiation with supporting user who forget their passwords or who let their password expire.
- Provides confidentiality to all. This is a process to controlling access to files either in storage or in transit.
- Availability is the cinderella of security as it is rarely discussed but however safe from hackers to the system.
- So, four goals are confidentially, Integrity, Availability and Non-repudiation.

6.5.3 System threats

Q.: Write a short note on system threats. [S-12, 3M]

- In computer system many resource are sharable to more users.
- Every user shared as per their requirement.
- So, more sharing gives, raise to more possibility of security threats or protection, thus it required higher protection.
- Even sharing of resources itself creates a contracting environment if proper assignment of resources not done.
- System threats are serious matter by many way most people wanted to hack or access the resource. Some of the major threats are.

1. Unauthorized disclosure of information called disclosure.
2. Unauthorized attention a detection of information called amendment.
3. Unauthorized fabrication of information called fabrication.
4. Denial of service to authorized user called Denial.
5. Unauthorized use of service called Tapping.

Note: Security threats can arise from unintentional or deliberate reasons.

6.5.4 Cryptography

1. **What is cryptography. [W-08, 3M, S-09 5M, W-09, 4M, S-11, 5M, S-12, 6M]**
2. **Write short note on cryptography.**

Ans.: In todays computer environment every users are communicate with the electronic mail, simple chat, messaging, etc.

- The message transfer from one end (sender) to another end (Receiver), it goes from many intermediate nodes and it leads to dangers.
- Because many opponent or hacker are sitting and listening the n/w traffic.
- so to avoid such fraud the message quality should be secure, even if any hacker gets message he/she can not be able to read it.
- Now, the cryptography is a technique of encoding (i.e. encryption) and decoding (i.e. decryption) message. So that in between user or opponent can not read that message only those having rights (key) to access that message they can.

Some component of cryptography.

1. **Plain Text/Message:** Plain Text is a text (data or message) that user wanted to send over the network.
2. **Sender:** Is one of the component that sends the message i.e. its a originator of communication.
3. **Receiver:** It exact reverse of sender.
4. **Cipher Text:** is a encrypted message i.e. conversion of plain text into encoded (non-readable) format.
5. **Encryption:** This is the process of converting message into non reliable format by using some encryption algorithm.
6. **Decryption:** This is the process of converting encoded message into readable format using different algorithms.

Diagram

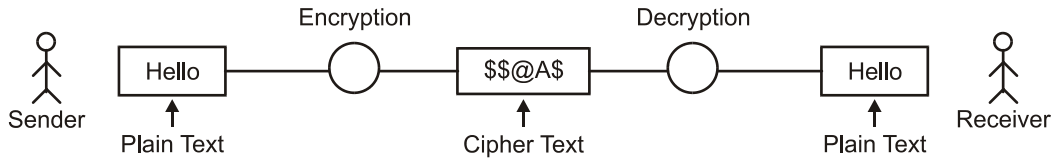


Fig. Process of cryptography

6. **Key:** in simple term key is the access rights to the message, sender and receiver have its own public private key to encrypt and decrypt the messages.
7. **Algorithm:** This categories into encryption and decryption algorithm of message.

Q.: Explain the cryptography Technique. [S-10, 4M W-11, 4M]

Note: Cryptography terms is a tremendous and wide in security it is out of scope of operating system syllabus.

- There are so many cryptography tech that are categorized into two crypto system.

1. Symmetric cryptosystem.
2. Asymmetric cryptosystem.

1. Symmetric cryptosystem

– Symmetric key cryptography uses same key for both encryption and decryption.
– The key need to be kept private hence symmetric cryptosystem also called private key cryptography.
– The secure distribution of key is the major challenge that is associated with this system.
– Data Encryption standard a Advance Encryption standard are the algorithms commonly uses to this cryptosystem. (DES, AES).
– Reliability of security of exchange is based on the security and symmetric key.
– The new ciphertext can be created by an attackers interpretation which is uses the symmetric key reading the cipher text.

2. Asymmetric cryptosystem

– In asymmetric cryptosystem both private and public key are used in operation.
– One key is used for the data encryption and another key is used for data decryption.
– Asymmetric key system is used in solving the challenge of secure distribution of the secret key.
– Authentication is another feature of a symmetric cryptosystem.