IARE '24 DISCORD TEAM

II-II

OS MODULE 5 SOLUTIONS

VISHAL • SYED IKRAM • PRANAV • UJJWAL • ASRITHA

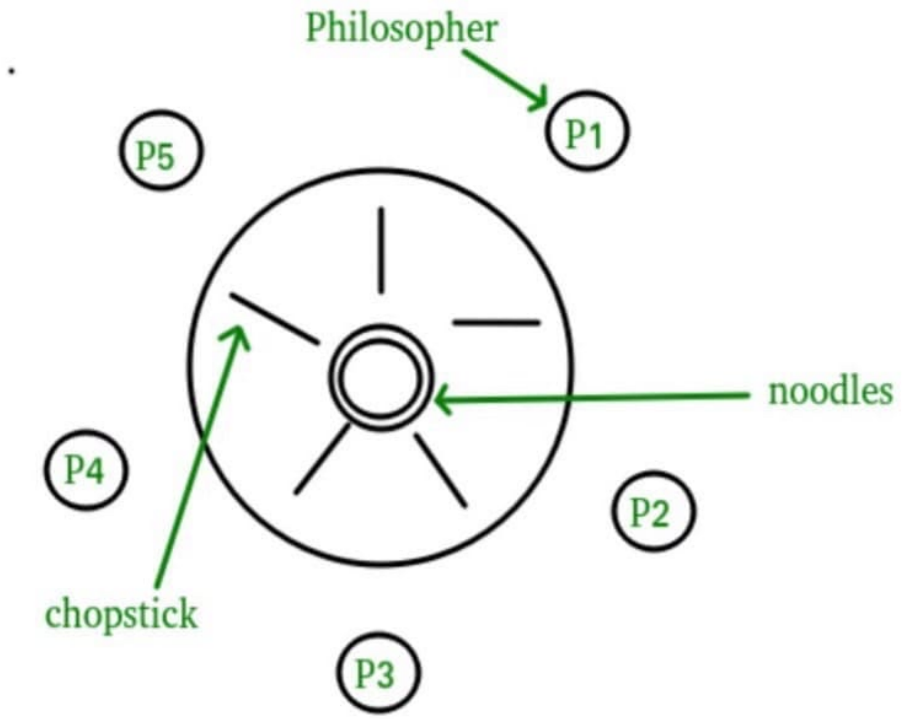
DEADLOCKS, PROTECTION

Completed

OS MODULE 5

PART A

1. Consider the version of the dining-philosophers problem in which the chopsticks are placed at the center of the table and any two of them can be used by a philosopher. Assume that requests for chopsticks are made one at a time. Describe a simple rule for determining whether a particular request can be satisfied without causing deadlock given the current allocation of chopsticks to philosophers



Dining Philosophers Problem

The rule that prevents deadlock philosophers problem is: When one philosopher makes a request for the first chopstick, do not award the

request if there are no other philosophers with two chopsticks and if there is only one chopstick remaining.

2. Consider a system consisting of m resources of the same type being shared by n processes. A process can request or release only one resource at a time. Show that the system is deadlock free if the following two conditions hold: a) The maximum need of each process is between one resource and m resources. b) The sum of all maximum needs is less than m +n.

...Proof:

Suppose N = Sum of all Need(i), A = Sum of all Allocation(i), M = Sum of all Max(i). Use contradiction to prove.

Assume this system is not deadlock free. If there exists a deadlock state, then A = m because there's only one kind of resource and resources can be requested and released only one at a time. From condition b, N + A = M < m + n. So we get N + m < m + n. So we get N < n. It shows that at least one process i that Need(i) = 0. From condition a, Pi can release at least 1 resource. So there are n-1 processes sharing m resources now, condition a and b still hold. Go on the argument, no process will wait permenently, so there's no deadlock.

3. Given 3 processes A,B and C, three resources x,y and z and following events, a. A requests x ii) A requests y iii) B requests y iv) B requests z v) C requests z vi) C requests x vii) C requests y Assume that requested resources should always be allocated to the request process if it is available. Draw the resource allocation graph for the sequences. And also mention whether it is a deadlock? If it is, how to recover the deadlock.

Tutorial - 3

Q1) Given 3 processes A, B & C, three single instance resources X, Y & Z are following events happen in the sequence as given:
1. A requests X, A requests Y & B requests Z, C requests Z, C requests X & C requests Y
2. A requests X, B requests Y, C requests Z, A requests Y, B requests Z, C requests X, C releases Y.

Assuming that initially all resources are free & once requested gets allocated to the requesting process if it is available. Draw the single instance resource allocation graph for the sequence. Add information whether it is a deadlock, justify if yes or no?

⇒ 1)

Here, Allocation is represented in Blue
Requesting is represented in Red

Basically there are mainly four conditions of deadlock:
(i) Mutual Exclusion: (All resources are non-sharable) ✓
(ii) No-Preemption (Process should not be preempted) ✓
(iii) Hold & Wait: (At least one process is holding a resource & waiting for another resource) X (Y)
(iv) Circular Wait ✓ (X)

Here, in this pattern,
• Hold & Wait condition is not there.
Reason: A is not waiting for any resource
• Circular Wait is also not there

[Hence, Deadlock will not happen in this case.]

2)

Here, in this case:
• Mutual Exclusion is there ✓
• No-Preemption ✓
• Hold & Wait ✓
• Circular Wait ✓

Here, All our deadlock condition satisfy.
Also, when cycle is present, in single instance Allocation graph then deadlock will definitely happen.
[Hence, deadlock will occur in this case.]

4. Explain How does the principle of least privilege aid in the creation of protection systems

The principle of least privilege works by allowing only enough access to perform the required job. In an IT environment, adhering to the principle of least privilege reduces the risk of attackers gaining access to critical systems or sensitive data by compromising a low-level user account, device, or application. Implementing the POLO helps contain compromises to their area of origin, stopping them from spreading to the system at large.
The principle of least privilege prevents the spread of malware on your network. An administrator or superuser with access to a lot of other network resources and infrastructure could potentially spread malware to all those other systems.
Five Benefits of the Least Privilege Principle

1. The least privilege principle reduces liability.
2. Least privilege access limits the possibility of catastrophic damage.
3. The principle of least privilege protects against common attacks, like SQL injections.
4. Data classification creates a healthy, secure network.
5. The least privilege principle enables better security and audit capabilities.

5. Describe how the Java protection model would be compromised if a Java program were allowed to directly alter the annotations of its stack frame.

When a thread issues an access request in a doPrivileged() block, the stack frame of the calling thread is annotated and the stack frame can make subsequent method.Thus, the annotation serves to mark a calling thread as being privileged.

6. Describe the Coffman's conditions that lead to a deadlock.

A deadlock situation can arise if and only if all of the following conditions hold simultaneously in a system:

- **Mutual Exclusion:** At least one resource must be non-shareable. Only one process can use the resource at any given instant of time.
- **Hold and Wait or Resource Holding:** A process is currently holding at least one resource and requesting additional resources which are being held by other processes.
- **No Preemption:** The operating system must not de-allocate resources once they have been allocated; they must be released by the holding process voluntarily.
- **Circular Wait:** A process must be waiting for a resource which is being held by another process, which in turn is waiting for the first process to release the resource. In general, there is a set of waiting processes, P = {P1, P2, ..., PN}, such that P1 is waiting for a resource held by P2, P2 is waiting for a resource held by P3 and so on till PN is waiting for a resource held by P1.

These four conditions are known as the Coffman conditions from their first description in a 1971 article by Edward G. Coffman.

7. A system contains three programs and each requires three tape units for its operation. Explain the minimum number of tape units which the system must have such that deadlocks never arise is

Data:

Programs in system = Y = 3

Tapes per program needed = R = 3

Calculation:

Max tape per program to be in deadlock = needed − 1 = 3 − 1 = 2

For Y program, max resource to be in deadlock = Y × 2 = 2Y

Condition for deadlock free

Total tapes > 2Y

$T > 2 \times 3$

$T_{min} = 7$

The minimum number of tape units which the system must have such that deadlocks never arise is 7

8. A system has 6 identical resources and N processes competing for them. Each process can request at most 2 resources. Explain which one of the following values of N could lead to a deadlock.

Data:

Available identical Resources = R = 6

Max needs per process = 2

Concepts:

Deadlock can occur If any process gets available resource < needed (requested) resource

Max resource per process to be in deadlock = needed − 1 = 2 − 1 = 1

For N process, max resource to be in deadlock = N × 1 = N

Condition for deadlock

$N \geq R$

$N \geq 6$

values of N could lead to a deadlock is 6

P_1	P_2	P_3	P_4	P_5	P_6
1	1	1	1	1	1

There can be deadlock

9. Define in detail the technique of deadlock avoidance

In order to avoid deadlocks, the process must tell the OS the maximum number of resources a process can request to complete its execution.
The simplest and most useful approach states that the process should declare the maximum number of resources of each type it may ever need. The Deadlock avoidance algorithm examines the resource allocations so that there can never be a circular wait condition.
Deadlock avoidance can be done with Banker's Algorithm.

Banker's Algorithm
Banker's Algorithm is resource allocation and deadlock avoidance algorithm which test all the request made by processes for resources, it checks for

the safe state, if after granting request system remains in the safe state it allows the request and if there is no safe state it doesn't allow the request made by the process.

10. Explain briefly about the purpose of the banker's algorithm.

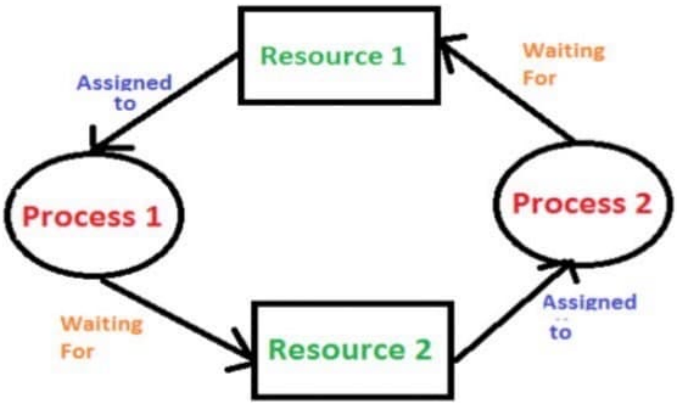
Banker's Algorithm is used majorly in the banking system to avoid deadlock. It helps you to identify whether a loan will be given or not. The banker's algorithm is a resource allocation and deadlock avoidance algorithm that tests for safety by simulating the allocation for predetermined maximum possible amounts of all resources, then makes an "s-state" check to test for possible activities, before deciding whether allocation should be allowed to continue.

Banker's algorithm is named so because it is used in the banking system to check whether a loan can be sanctioned to a person or not. Suppose there are n number of account holders in a bank and the total sum of their money is S. If a person applies for a loan then the bank first subtracts the loan amount from the total money that bank has and if the remaining amount is greater than S then only the loan is sanctioned. It is done because if all the account holders come to withdraw their money then the bank can easily do it.
In other words, the bank would never allocate its money in such a way that it can no longer satisfy the needs of all its customers. The bank would always try to be in a safe state.

PART B

1. What is the deadlock? Explain the necessary conditions for its occurrence

A deadlock is a situation in which more than one process is blocked because it is holding a resource and also requires some resource that is acquired by some other process. Therefore, none of the processes gets executed.



IMG: Example of Deadlock

Four necessary conditions for its occurrence are:

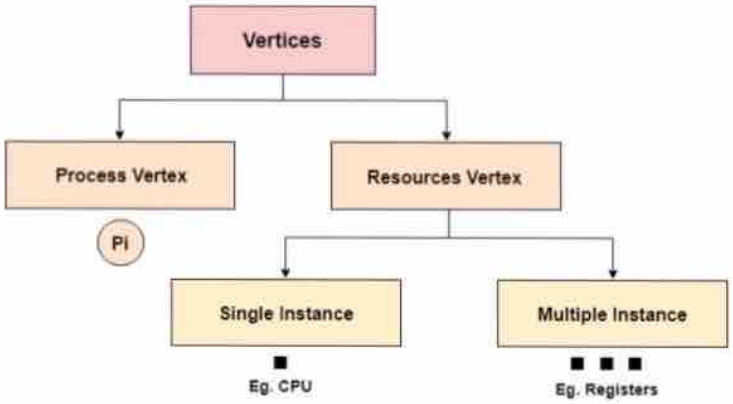
1. Mutual Exclusion: At least one resource must be non-shareable. Only one process can use the resource at any given instant of time.
2. Hold and Wait or Resource Holding: A process is currently holding at least one resource and requesting additional resources which are being held by other processes.
3. No Preemption: The operating system must not de-allocate resources once they have been allocated; they must be released by the holding process voluntarily.
4. Circular Wait: A process must be waiting for a resource which is being held by another process, which in turn is waiting for the first process to release the resource. In general, there is a set of waiting processes, P = {P1, P2, ..., PN}, such that P1 is waiting for a resource held by P2, P2 is waiting for a resource held by P3 and so on till PN is waiting for a resource held by P1.

2. Explain briefly resource allocation graph with examples

The resource allocation graph is the pictorial representation of the state of a system. As its name suggests, the resource allocation graph is the complete information about all the processes which are holding some resources or waiting for some resources.

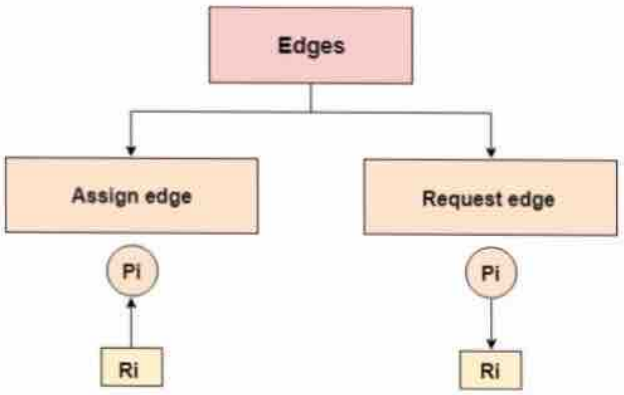
It also contains the information about all the instances of all the resources whether they are available or being used by the processes.

In Resource allocation graph, the process is represented by a Circle while the Resource is represented by a rectangle. Let's see the types of vertices and edges in detail.



Vertices are mainly of two types, Resource and process. Each of them will be represented by a different shape. Circle represents process while rectangle represents resource.

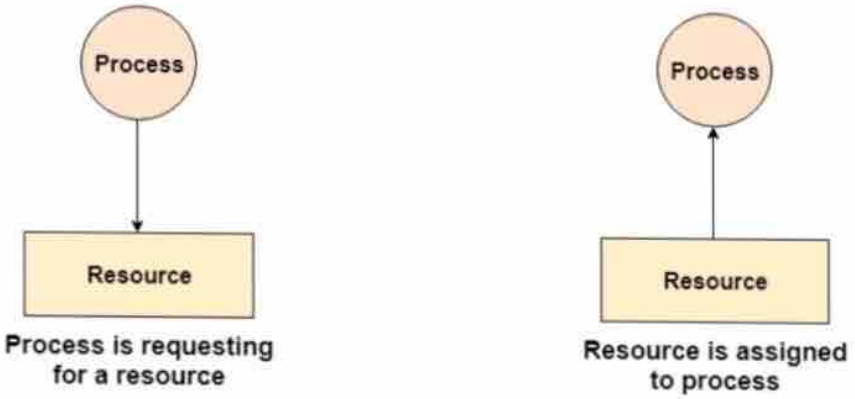
A resource can have more than one instance. Each instance will be represented by a dot inside the rectangle.



Edges in RAG are also of two types, one represents assignment and other represents the wait of a process for a resource. The above image shows each of them.

A resource is shown as assigned to a process if the tail of the arrow is attached to an instance to the resource and the head is attached to a process.

A process is shown as waiting for a resource if the tail of an arrow is attached to the process while the head is pointing towards the resource.

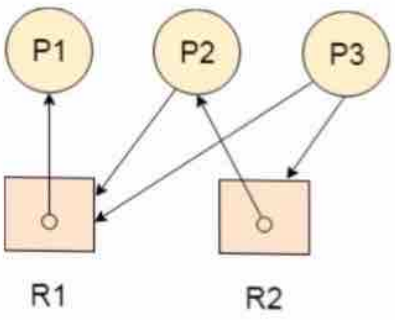


Example

Let's Consider 3 processes P1, P2 and P3, and two types of resources R1 and R2. The resources are having 1 instance each.

According to the graph, R1 is being used by P1, P2 is holding R2 and waiting for R1, P3 is waiting for R1 as well as R2.

The graph is deadlock free since no cycle is being formed in the graph.



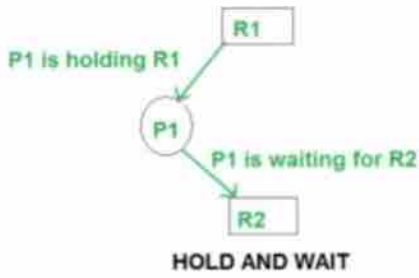
3. Explain the methods for deadlock prevention.

Deadlock prevention is eliminating one of the necessary conditions of deadlock so that only safe requests are made to OS and the possibility of deadlock is excluded before making requests.
As now requests are made carefully, the operating system can grant all requests safely.

Deadlock Prevention Techniques:
Deadlock prevention techniques refer to violating any one of the four necessary conditions. We will see one by one how we can violate each of them to make safe requests and which is the best approach to prevent deadlock.

- Eliminate Mutual Exclusion:
  It is not possible to dis-satisfy the mutual exclusion because some resources, such as the tape drive and printer, are inherently non-shareable.

- Eliminate Hold and wait:
  Allocate all required resources to the process before the start of its execution, this way hold and wait condition is eliminated but it will lead to low device utilisation. For example, if a process requires a printer at a later time and we have allocated a printer before the start of its execution, the printer will remain blocked till it has completed its execution.
  The process will make a new request for resources after releasing the current set of resources. This solution may lead to starvation.



- Eliminate No Preemption:
  Preempt resources from the process when resources are required by other high priority processes.

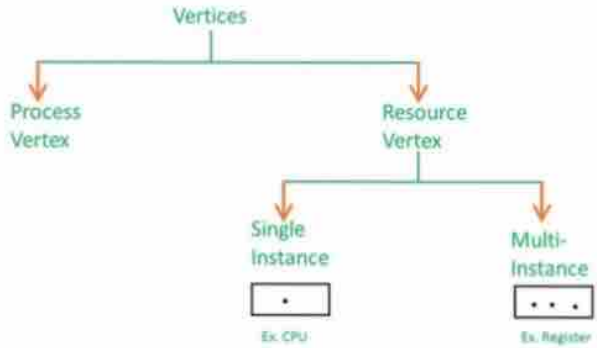
- Eliminate Circular Wait:
  Each resource will be assigned with a numerical number. A process can request the resources to increase/decrease. order of numbering. For Example, if P1 process is allocated R5 resources, now next time if P1 ask for R4, R3 less than R5 such request will not be granted, only request for resources more than R5 will be granted.

4. Explain the resource allocation graph.

As Banker's algorithm using some kind of table like allocation, request, available all that thing to understand what is the state of the system. Similarly, if you want to understand the state of the system instead of using those table, actually tables are very easy to represent and understand it, but then still you could even represent the same information in the graph. That graph is called **Resource Allocation Graph (RAG).**
So, resource allocation graph is explained to us what is the state of the system in terms of processes and resources. Like how many resources are available, how many are allocated and what is the request of each process. Everything can be represented in terms of the diagram. One of the advantages of having a diagram is, sometimes it is possible to see a deadlock directly by using RAG, but then you might not be able to know that by looking at the table. But the tables are better if the system contains lots of process, resource and graph is better if the system contains less number of process and resource. We know that any graph contains vertices and edges. So RAG also contains vertices and edges. In RAG vertices are of two types:

1. **Process vertex -** Every process will be represented as a process vertex. Generally, the process will be represented with a circle.

2. **Resource vertex -** Every resource will be represented as a resource vertex. It is also two types:
    - **Single instance type resource -** It represents as a box, there will be one dot. So the number of dots indicate how many instances are present of each resource type.
    - **Multi - resource instance type resource -** It also represents as a box, inside the box, there will be many dots present.



5. Differentiate the deadlock handling methods

Methods of handling deadlocks : There are three approaches to deal with deadlocks.

1. Deadlock Prevention
2. Deadlock avoidance
3. Deadlock detection

These are explained below.

1. Deadlock Prevention : The strategy of deadlock prevention is to design the system in such a way that the possibility of deadlock is excluded. Indirect methods prevent the occurrence of one of three necessary conditions of deadlock i.e., mutual exclusion, no pre-emption and hold and wait. Direct methods prevent the occurrence of circular wait. Prevention techniques – Mutual exclusion – is supported by the OS. Hold and Wait – condition can be prevented by requiring that a process requests all its

required resources at one time and blocking the process until all of its requests can be granted at a same time simultaneously. But this prevention does not yield good result because :

- long waiting time required
- in efficient use of allocated resource
- A process may not know all the required resources in advance

No pre-emption – techniques for 'no pre-emption are'
If a process that is holding some resource, requests another resource that can not be immediately allocated to it, all resources currently being held are released and if necessary, request them again together with the additional resource.
If a process requests a resource that is currently held by another process, the OS may preempt the second process and require it to release its resources. This works only if both the processes do not have the same priority.
Circular wait One way to ensure that this condition never hold is to impose a total ordering of all resource types and to require that each process requests resource in an increasing order of enumeration, i.e., if a process has been allocated resources of type R, then it may subsequently request only those resources of types following R in ordering.

2. Deadlock Avoidance : This approach allows the three necessary conditions of deadlock but makes judicious choices to assure that deadlock point is never reached. It allows more concurrency than avoidance detection. A decision is made dynamically whether the current resource allocation request will, if granted, potentially lead to deadlock. It requires the knowledge of future process requests. Two techniques to avoid deadlock :

- Process initiation denial

- Resource allocation denial

Advantages of deadlock avoidance techniques :

- Not necessary to pre-empt and rollback processes
- Less restrictive than deadlock prevention

Disadvantages :

- Future resource requirements must be known in advance
- Processes can be blocked for long periods
- Exists fixed number of resources for allocation

3. Deadlock Detection : Deadlock detection is used by employing an algorithm that tracks the circular waiting and killing one or more processes so that deadlock is removed. The system state is examined periodically to determine if a set of processes is deadlocked. A deadlock is resolved by aborting and restarting a process, relinquishing all the resources that the process held.

This technique does not limit resources access or restrict process action.
Requested resources are granted to processes whenever possible.
It never delays the process initiation and facilitates online handling.
The disadvantage is the inherent pre-emption losses.

6. Explain Banker's algorithm for deadlock avoidance with an example

Banker's Algorithm
Banker's Algorithm is resource allocation and deadlock avoidance algorithm which test all the request made by processes for resources, it checks for the safe state, if after granting request system remains in the safe state it allows the request and if there is no safe state it doesn't allow the request made by the process.

Banker's Algorithm is used majorly in the banking system to avoid deadlock. It helps you to identify whether a loan will be given or not.

The banker's algorithm is a resource allocation and deadlock avoidance algorithm that tests for safety by simulating the allocation for predetermined maximum possible amounts of all resources, then makes an "s-state" check to test for possible activities, before deciding whether allocation should be allowed to continue.

Banker's algorithm is named so because it is used in the banking system to check whether a loan can be sanctioned to a person or not. Suppose there are n number of account holders in a bank and the total sum of their money is S. If a person applies for a loan then the bank first subtracts the loan amount from the total money that bank has and if the remaining amount is greater than S then only the loan is sanctioned. It is done because if all the account holders come to withdraw their money then the bank can easily do it.
In other words, the bank would never allocate its money in such a way that it can no longer satisfy the needs of all its customers. The bank would always try to be in a safe state.

7. Discuss the various issues that need to be considered through the process of revocation of access rights.

With an access list scheme revocation is easy, immediate, and can be selective, general, partial, total, temporary, or permanent, as desired.
With capabilities lists the problem is more complicated, because access rights are distributed throughout the system. A few schemes that have been developed include:

- Reacquisition - Capabilities are periodically revoked from each domain, which must then re-acquire them.

- Back-pointers - A list of pointers is maintained from each object to each capability which is held for that object.
- Indirection - Capabilities point to an entry in a global table rather than to the object. Access rights can be revoked by changing or invalidating the table entry, which may affect multiple processes, which must then re-acquire access rights to continue.
- Keys - A unique bit pattern is associated with each capability when created, which can be neither inspected nor modified by the process.
    - A master key is associated with each object.
    - When a capability is created, its key is set to the object's master key.
    - As long as the capability's key matches the object's key, then the capabilities remain valid.

Revocation of Access Rights:

The need to revoke access rights dynamically raises several questions:

- ➔ Immediate versus delayed - If delayed, can we determine when the revocation will take place?
- ➔ Selective versus general - Does revocation of an access right to an object affect all users who have that right, or only some users?
- ➔ Partial versus total - Can a subset of rights for an object be revoked, or are all rights revoked at once?
- ➔ Temporary versus permanent - If rights are revoked, is there a mechanism for processes to re-acquire some or all of the revoked rights?

8. State and explain the methods involved in recovery from deadlocks.

Deadlock recovery performs when a deadlock is detected:
When a deadlock is detected, then our system stops working, and after the recovery of the deadlock, our system starts working again.

Therefore, after the detection of deadlock, a method/way must be required to recover that deadlock to run the system again. The method/way is called deadlock recovery.

1. Deadlock recovery through preemption
2. Deadlock recovery through rollback
3. Deadlock recovery through killing processes

Deadlock Recovery through Preemption

The ability to take a resource away from a process, have another process use it, and then give it back without the process noticing. It is highly dependent on the nature of the resource.

Deadlock recovery through preemption is too difficult or sometimes impossible.

Deadlock Recovery through RollBack

In this case of deadlock recovery through rollback, whenever a deadlock is detected, it is easy to see which resources are needed.

To do the recovery of deadlock, a process that owns a needed resource is rolled back to a point in time before it acquired some other resource just by starting one of its earlier checkpoints.

Deadlock Recovery through Killing Processes

This method of deadlock recovery through killing processes is the simplest way of deadlock recovery.

Sometimes it is best to kill a process that can be returned from the beginning with no ill effects.

9. Describe resource-allocation graph. Explain how resource graph can be used for detecting deadlocks.

The resource allocation graph is the pictorial representation of the state of a system. As its name suggests, the resource allocation graph is the complete information about all the processes which are holding some resources or waiting for some resources.

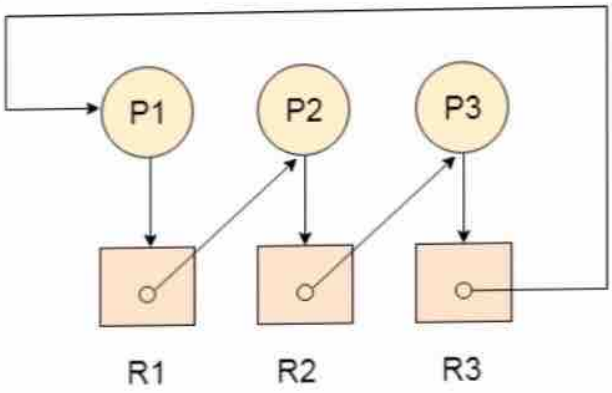
It also contains the information about all the instances of all the resources whether they are available or being used by the processes.

Deadlock Detection using RAG

If a cycle is being formed in a Resource allocation graph where all the resources have the single instance then the system is deadlocked.

In Case of Resource allocation graphs with multi-instanced resource types, Cycle is a necessary condition of deadlock but not the sufficient condition.

The following example contains three processes P1, P2, P3 and three resources R2, R2, R3. All the resources are having single instances each.



If we analyse the graph then we can find out that there is a cycle formed in the graph since the system is satisfying all the four conditions of deadlock.

For more click here

10 Describe the terms. a) Race condition b) Atomic transaction c) Critical section d) Mutual exclusion

a) Race condition

A race condition occurs when two or more threads can access shared data and they try to change it at the same time. Because the thread scheduling algorithm can swap between threads at any time, you don't know the order in which the threads will attempt to access the shared data. Therefore, the result of the change in data is dependent on the thread scheduling algorithm, i.e. both threads are "racing" to access/change the data.
A race condition is a situation that may occur inside a critical section. This happens when the result of multiple thread execution in the critical section differs according to the order in which the threads execute.

b) Atomic transaction
An atomic transaction is an indivisible and irreducible series of database operations such that either all occurs, or nothing occurs. A guarantee of atomicity prevents updates to the database occurring only partially, which can cause greater problems than rejecting the whole series outright. As a consequence, the transaction cannot be observed to be in progress by another database client. At one moment in time, it has not yet happened, and at the next it has already occurred in whole (or nothing happened if the transaction was cancelled in progress).
An example of an atomic transaction is a monetary transfer from bank account A to account B. It consists of two operations, withdrawing the money from account A and saving it to account B. Performing these operations in an atomic transaction ensures that the database remains in a consistent state, that is, money is neither lost nor created if either of those two operations fails.
critical section
The critical section refers to the segment of code where processes access shared resources, such as common variables and files, and perform write operations on them.

Since processes execute concurrently, any process can be interrupted mid-execution. In the case of shared resources, partial execution of processes can lead to data inconsistencies. When two processes access and manipulate the shared resource concurrently, and the resulting execution outcome depends on the order in which processes access the resource; this is called a race condition.

Mutual exclusion

A mutual exclusion (mutex) is a program object that prevents simultaneous access to a shared resource. This concept is used in concurrent programming with a critical section, a piece of code in which processes or threads access a shared resource. Only one thread owns the mutex at a time, thus a mutex with a unique name is created when a program starts

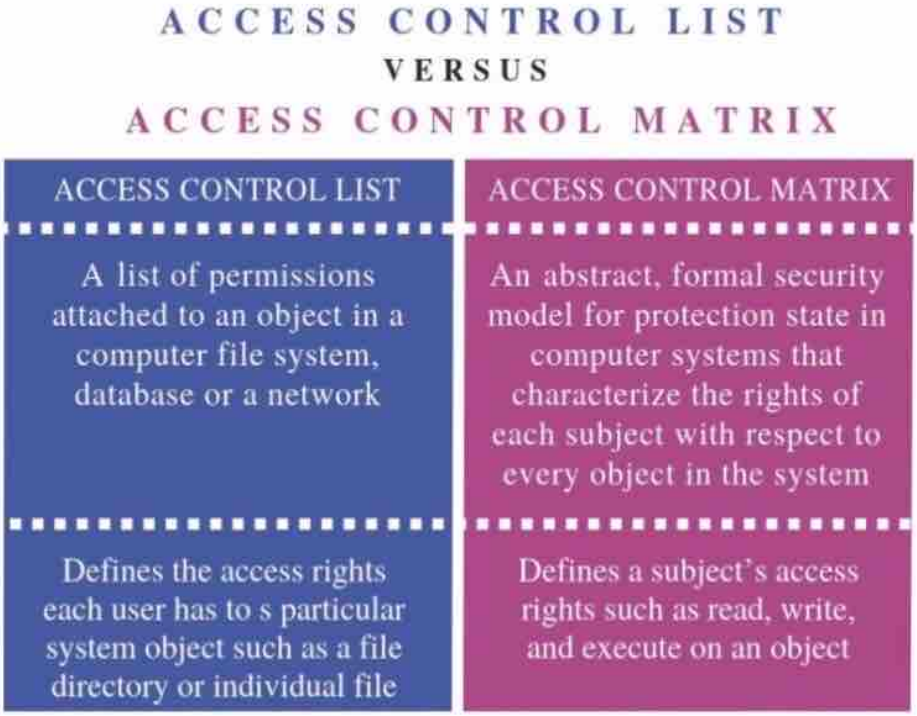
Refer example from Mutual Exclusion in Synchronization - GeeksforGeeks

11. Describe how the access matrix facility and role-based access control facility are similar. How do they differ?

Access Matrix is a security model of protection state in computer systems. It is represented as a matrix. Access matrix is used to define the rights of each process executing in the domain with respect to each object. The rows of the matrix represent domains and columns represent objects.

Role-based access control (RBAC), also known as role-based security, is a mechanism that restricts system access. It involves setting permissions and privileges to enable access to authorised users. Most large organisations use role-based access control to provide their employees with varying

levels of access based on their roles and responsibilities. This protects sensitive data and ensures employees can only access information and perform actions they need to do their jobs.



12. Explain why a capability based system such as Hydra provides greater flexibility than the ring- protection scheme in enforcing protection policies.

The ring-based protection scheme requires the modules to be ordered in a strictly hierarchical fashion. It also enforces the restriction that system code in internal rings cannot invoke operations in the external rings. This restriction limits the flexibility in structuring the code andis unnecessarily restrictive. The capability system provided by Hydra not only allows unstructured interactions between different modules,but also enables the dynamic instantiation of new modules as the need arises.

13. Define Goals of protection

Processes in the system must be protected from one another's activities. Else there may be disruption in the normal working. Protection refers to a mechanism for controlling the access of programs, processes, or users to resources defined by the computer system

Goals of protection

1. Provides a means to distinguish between authorised and unauthorised usage.
2. To prevent mischievously, intentional violation of an access restriction by the user.
3. To ensure that each program component which is active in a system uses system resources only in ways consistent with stated policies. (This gives a reliable system).
4. To detect latent errors at the interfaces between the component subsystems. (This can improve reliability). Early detection helps in preventing malfunctioning of subsystems.
5. To enforce policies governing resource usage.

14. Define Principles of protection.

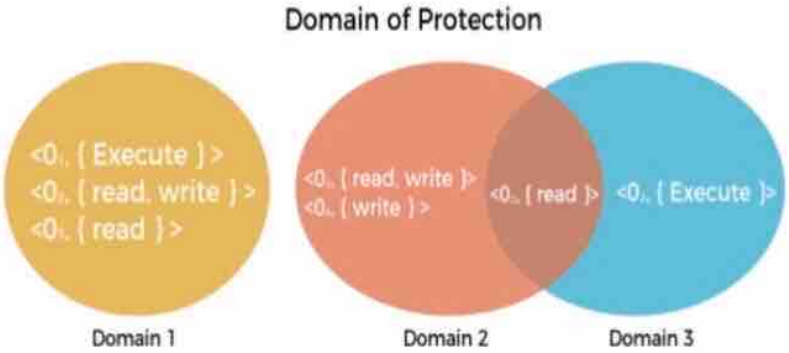
1. The time-tested guiding principle used for protection is called the principle of least privilege. It states that programs, users and even systems be given just enough privileges to perform their tasks.
2. An OS following this principle implements its features, programs, system calls, and data structures so that failure or compromise of a component does the minimum damage and allows minimum damage to be done. Such OS has fine-grained access control.
3. It provides mechanisms to enable privileges when they are needed and to disable them when not needed.
4. Privileged function access have audit trails that enable programmer or systems administrator or law-enforcement officer to trace all protection and security activities of the system.

5. We can create separate accounts for each user with just the privileges that the user needs.

15. Discuss about domain of protection

Various domains of protection in operating system are as follows:

- The protection policies restrict each process's access to its resource handling. A process is obligated to use only the resources necessary to fulfil its task within the time constraints and in the mode in which it is required. It is a process's protected domain.
- Processes and objects are abstract data types in a computer system, and these objects have operations that are unique to them. A domain component is defined as <object, {set of operations on object}>.



System with three protection domains.

- Each domain comprises a collection of objects and the operations that may be implemented on them. A domain could be made up of only one process, procedure, or user. If a domain is linked with a procedure, changing the domain would mean changing the procedure ID. Objects may share one or more common operations.

16. Why do you need to provide protection to the system?

Various needs of protection in the operating system are as follows:

1. There may be security risks like unauthorised reading, writing, modification, or preventing the system from working effectively for authorised users.
2. It helps to ensure data security, process security, and program security against unauthorised user access or program access.
3. It is important to ensure no access rights' breaches, no viruses, no unauthorised access to the existing data.
4. Its purpose is to ensure that only the systems' policies access programs, resources, and data.

17. Discuss the access matrix implementation techniques.

1. Global Table:

- The simplest approach is one big global table with <domain, object, rights> entries.
- Unfortunately this table is very large (even if sparse) and so cannot be kept in memory (without invoking virtual memory techniques.)
- There is also no good way to specify groupings - if everyone has access to some resource, then it still needs a separate entry for every domain.

2. Access Lists for Objects:

- 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.

3. Capability Lists for Domains:

- In a similar fashion, each row of the table can be kept as a list of the capabilities of that domain.
- Capability lists are associated with each domain, but not directly accessible by the domain or any user process.

- Capability lists are themselves protected resources, distinguished from other data in one of two ways:
- A tag, possibly hardware implemented, distinguishing this special type of data. (other types may be floats, pointers, booleans, etc.)
- The address space for a program may be split into multiple segments, at least one of which is inaccessible by the program itself and used by the operating system for maintaining the process's access right capability list.

4. A Lock-Key Mechanism:

- Each resource has a list of unique bit patterns, termed locks.
- Each domain has its own list of unique bit patterns, termed keys.
- Access is granted if one of the domain's keys fits one of the resource's locks.
- Again, a process is not allowed to modify its own keys.
- It is effective and flexible.

Object / Domain	F_1	F_2	F_3	Laser Printer
D_1	read		read	
D_2				Print
D_3		read	exceute	
D_4	read/write		read/write	

Access Matirx

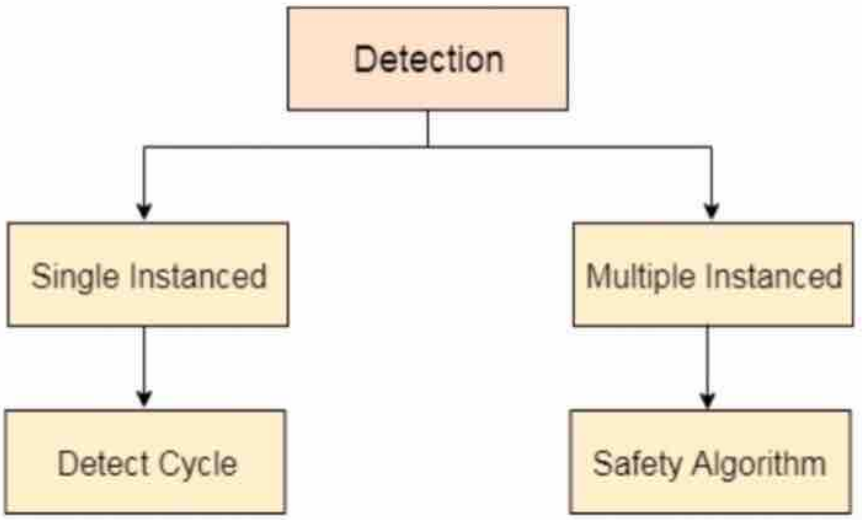
Click here to refer more info

18. Compare the various access matrix implementation techniques

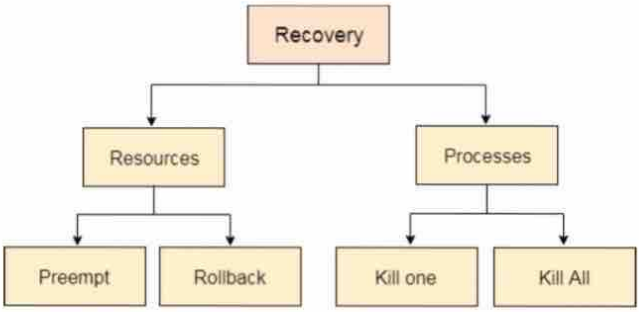
Refer Q.No.17 click here

19. Discuss the deadlock detection method in detail.

In this approach, The OS doesn't apply any mechanism to avoid or prevent the deadlocks. Therefore the system considers that the deadlock will definitely occur. In order to get rid of deadlocks, The OS periodically checks the system for any deadlock. In case, it finds any of the deadlock then the OS will recover the system using some recovery techniques.
The main task of the OS is detecting the deadlocks. The OS can detect the deadlocks with the help of a Resource allocation graph.



In single instanced resource types, if a cycle is being formed in the system then there will definitely be a deadlock.
On the other hand, in multiple instanced resource type graphs, detecting a cycle is not enough. We have to apply the safety algorithm on the system by converting the resource allocation graph into the allocation matrix and request matrix.
In order to recover the system from deadlocks, either OS considers resources or processes.



20. Explain various schemes to implement revocation for capabilities

With an access list scheme revocation is easy, immediate, and can be selective, general, partial, total, temporary, or permanent, as desired.
With capabilities lists the problem is more complicated, because access rights are distributed throughout the system. A few schemes that have been developed include:

- Reacquisition - Capabilities are periodically revoked from each domain, which must then re-acquire them.
- Back-pointers - A list of pointers is maintained from each object to each capability which is held for that object.
- Indirection - Capabilities point to an entry in a global table rather than to the object. Access rights can be revoked by changing or invalidating the table entry, which may affect multiple processes, which must then re-acquire access rights to continue.
- Keys - A unique bit pattern is associated with each capability when created, which can be neither inspected nor modified by the process.
    - A master key is associated with each object.
    - When a capability is created, its key is set to the object's master key.
    - As long as the capability's key matches the object's key, then the capabilities remain valid.

    - The object master key can be changed with the set-key command, thereby invalidating all current capabilities.
    - More flexibility can be added to this scheme by implementing a list of keys for each object, possibly in a global table.