Unit – 5

Part-A

1 What is Deadlock

A deadlock is a situation in which two computer programs sharing the same resource are effectively preventing each other from accessing the resource, resulting in both programs ceasing to function.

2 Define resource. List some resources that a process might need for its execution?

3 Explain the sequence in which a process may utilize the resources in normal mode of operation?

- Request: If the request cannot be granted immediately,then the requesting process must wait until it can acquire the resource.
- Use: The process can operate on the resource.
- Release: The process releases the resource.

4 Describe the conditions under which a deadlock situation may arise

A deadlock situation can arise if the following four conditions hold simultaneously in a system:
a. Mutual exclusion
b. Hold and wait
c. No pre-emption
d. Circular wait

5 Explain safe state and unsafe state?

Safe state: if the system can allocate resources to the process in such a way that it can avoid deadlock. Then the system is in a safe state.

Unsafe state: if the system can't allocate resources to the process in such a way that it can avoid deadlock. Then the system is in the unsafe state.

6 Describe the representation of a resource-allocation graph?

A) 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).

7 Distinguish between deadlock avoidance and prevention strategies?

A) Avoidance:

• The goal for deadlock avoidance is to the system must not enter an unsafe state.

• Deadlock avoidance is often impossible to implement.

Prevention:

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

• Deadlock prevention is often impossible to implement.

8 Describe the purpose of banker's algorithm

Banker's algorithm is one form of deadlock-avoidance in a system. It gets its name from a banking system wherein the bank never allocates available cash in such a way that it can no longer satisfy the needs of all of its customers.

9 List the four data structures (matrices) that must be maintained to implement banker's algorithm?

A)

- Available
- Max
- Allocation
- Need

10 Describe the techniques for recovery from deadlock?

A)

- Killing the process.
- Resource Preemption

11 List the goals of protection?

- Obviously to prevent malicious misuse of the system by users or programs.
- To ensure that each shared resource is used only in accordance with system *policies,* which may be set either by system designers or by system administrators.
- To ensure that errant programs cause the minimal amount of damage possible.
- Note that protection systems only provide the *mechanisms* for enforcing policies and ensuring reliable systems. It is up to administrators and users to implement those mechanisms effectively.

12 Define the terms – object, domain, access right?

13 Write the format of an access matrix?

14 List the implementation techniques of access matrix?

- Network-wide capabilities
- Replicated access lists
- Application-Defined Objects
- Access Proxies
- Stack Check/Modified Name Space

15 Describe role-based access control?

Role-based access control (RBAC) is a method of regulating access to computer or network resources based on the roles of individual users within an enterprise. In this context, access is the ability of an individual user to perform a specific task, such as view, create, or modify a file. Roles are defined according to job competency, authority, and responsibility within the enterprise.

16 List the schemes that implement revocation of capabilities?

17 List any two example systems that implement capability-based protection?

A)

- Hydra
- Cambridge CAP System

18 Describe any one language-based protection schemes.

19 Write the main differences between capability lists and access lists?

An access list is a list for each object consisting of the domains with a nonempty set of access rights for that object. A capability list is a list of objects and the operations allowed on those objects for each domain**.**

20 State the protection problems that may arise if a shared stack is used for parameter passing?

The contents of the stack could be compromised by other process(es) sharing the stack.

21 State principle of least privilege?

The principle of least privilege (POLP), an important concept in computer security, is the practice of limiting access rights for users to the bare minimum permissions they need to perform their work. Under POLP, users are granted permission to read, write or execute only the files or resources they need to do their jobs: In other words, the least amount of privilege necessary.

Part –B

1 Define deadlock? what are the four conditions necessary for a deadlock situation to arise? how it can be prevented?

Deadlock:

- The computer system uses may types of resource which are then used by various processes to carry out their individual functions.
- But problem is that the amount of resources available is limited and many process needs to use it.
- A set of process is said to be in a deadlocked state when every process in the set is waiting for an event that can be caused only by another process in the set. The event can be resource acquisition, resource release etc. The resource can be physical (printers, memory space) or logical (semaphores, files)

The necessary and sufficient conditions for deadlock to occur are:

- Mutual Exclusion
    - A resource at a time can only be used by one process.

    - If another process is requesting for the same resource, then it must be delayed until that resource is released.

- Hold and Wait
    - A process is holding a resource and waiting to acquire additional resources that are currently being held by other processes.
- No Pre-emption:
    - Resources cannot be pre-empted
    - Resource can be released only by the process currently holding it based on its voluntary decision after completing the task
- Circular wait
    - A set of processes { P0,P1,….,Pn-1,Pn } such that the process P0 is waiting for resource held by P1,P1 is waiting for P2 ,and Pn is waiting for P0 to release its resources.
    - Every process holds a resource needed by the next process.

All the four above mentioned conditions should occur for a deadlock to occur

It ensures Deadlock Prevention:

- That the system never enters a deadlock state.
- It provides a set of methods to make sure that at least one of the four necessary conditions for a deadlock is never satisfied.
- Mutual Exclusion ⊇ this condition is needed to be checked for non-sharable resources (e.g. Printer)
- Hold and Wait ⊇ It requires a process to request a resource and get allocated before execution or allow process to request resources when the process has none.
- No preemption ⊇ If a process that is holding some resources requests another resource that cannot be immediately allocated to it, then all resources currently being held are released
- Circular Wait⊇we can impose a total ordering of all resource types, and ask that each process requests resources in an increasing order of enumeration.

- Disadvantage is that it can lead to low device utilization and reduced system throughput.

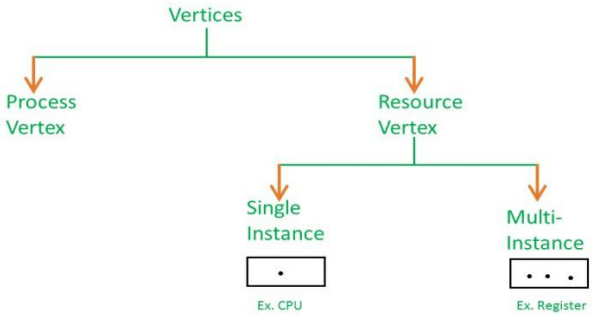
2 Explain briefly resource allocation graph with examples?

- 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 explains us 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 and 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 two type –

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 type –

- Single 7 instance type resource – It represents as a box, inside the 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.



3 Differentiate the deadlock handling methods?

Prevention:

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

• Deadlock prevention is often impossible to implement.

• The system doesnot require additional apriori information regarding the overall potential use of each resource for each process.

• In order for the system to prevent the deadlock condition it does not need to know all the details of all resources in existence, available and requested.

• Deadlock prevention techniques include non-blocking synchronization algorithms, serializing tokens, Dijkstras algorithm etc.

• Resource allocation strategy for deadlock prevention is conservative, it undercommits the resources.

• All resources are requested at once.

• In some cases preempts more than often necessary.

Avoidance:

• The goal for deadlock avoidance is to the system must not enter an unsafe state.

• Deadlock avoidance is often impossible to implement.

• The system requires additional apriori information regarding the overall potential use of each resource for each process.

• In order for the system to be able to figure out whether the next state will be safe or unsafe, it must know in advance at any time the number and type of all resources in existence, available, and requested.

• Deadlock avoidance techniques include Banker's algorithm, Wait/Die, Wound/Wait etc.

• Resource allocation strategy for deadlock avoidance selects midway between that of detection and prevention.

• Needs to be manipulated until atleast one safe path is found.

• There is no preemption.

Detection:

• The goal is to detect the deadlock after it occurs or before it occurs.

• Detecting the possibility of a deadlock before it occurs is much more difficult and is, in fact, generally undecidable. However, in specific environments, using specific means of locking resources, deadlock detection may be decidable.

• The system doesnot requires additional apriori information regarding the overall potential use of each resource for each process in all cases.

• In order for the system to detect the deadlock condition it does not need to know all the details of all resources in existence, available and requested.

• A deadlock detection technique includes, but is not limited to, Model checking. This approach constructs a Finite State-model on which it performs a progress analysis and finds all possible terminal sets in the model.

• Resource allocation strategy for deadlock detection is very liberal. Resources are granted as requested.

• Needs to be invoked periodically to test for deadlock.

• Preemption is seen.

4 Discuss in detail the technique of deadlock avoidance?

Deadlock Avoidance

Deadlock avoidance can be done with Banker's Algorithm.

Banker's Algorithm

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

Inputs to Banker's Algorithm
1. Max need of resources by each process.
2. Currently allocated resources by each process.
3. Max free available resources in the system.

Request will only be granted under below condition.
1. If request made by process is less than equal to max need to that process.

2. If request made by process is less than equal to freely availbale resource in the system.

Example

Total resources in system:

A B C D

6 5 7 6

Available system resources are:

A B C D

3 1 1 2

Processes (currently allocated resources):

A B C D

P1 1 2 2 1

P2 1 0 3 3

P3 1 2 1 0

Processes (maximum resources):

A B C D

P1 3 3 2 2

P2 1 2 3 4

P3 1 3 5 0

Need = maximum resources - currently allocated resources.

Processes (need resources):

A B C D

P1 2 1 0 1

P2 0 2 0 1

P3 0 1 4 0

5 Explain Banker's algorithm for deadlock avoidance with an example?

Deadlock Avoidance

Deadlock avoidance can be done with Banker's Algorithm.

Banker's Algorithm

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

Inputs to Banker's Algorithm
1. Max need of resources by each process.
2. Currently allocated resources by each process.
3. Max free available resources in the system.

Request will only be granted under below condition.
1. If request made by process is less than equal to max need to that process.

2. If request made by process is less than equal to freely availbale resource in the system.

Example

Total resources in system:

A B C D

6 5 7 6

Available system resources are:

A B C D

3 1 1 2

Processes (currently allocated resources):

A B C D

P1 1 2 2 1

P2 1 0 3 3

P3 1 2 1 0

Processes (maximum resources):

A B C D

P1 3 3 2 2

P2 1 2 3 4

P3 1 3 5 0

Need = maximum resources - currently allocated resources.

Processes (need resources):

A B C D

P1 2 1 0 1

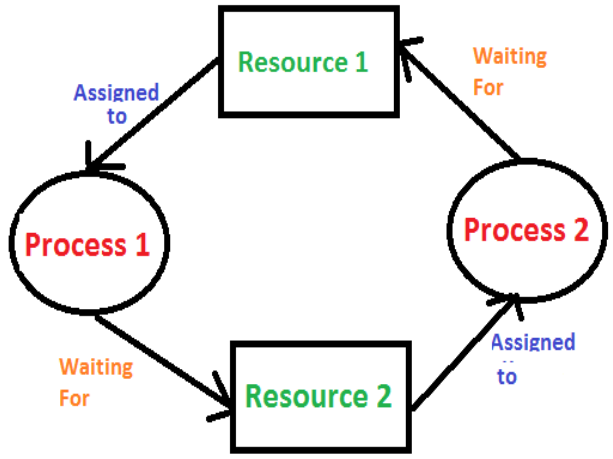
P2 0 2 0 1

P3 0 1 4 0

6 Discuss deadlock detection method in detail?

Deadlock Detection

1. If resources have single instance:
In this case for Deadlock detection we can run an algorithm to check for cycle in the Resource Allocation Graph. Presence of cycle in the graph is the sufficient condition for deadlock.



In the above diagram, resource 1 and resource 2 have single instances. There is a cycle R1–>P1–>R2–>P2. So Deadlock is Confirmed.

2. If there are multiple instances of resources:
Detection of cycle is necessary but not sufficient condition for deadlock detection, in this case system may or may not be in deadlock varies according to different situations.

7 State and explain the methods involved in recovery from deadlocks?

1. Killing the process.

killing all the process involved in deadlock.

Killing process one by one. After killing each

process check for deadlock again keep repeating

process till system recover from deadlock.

2. Resource Preemption
Resources are preempted from the processes involved in deadlock, preempted resources

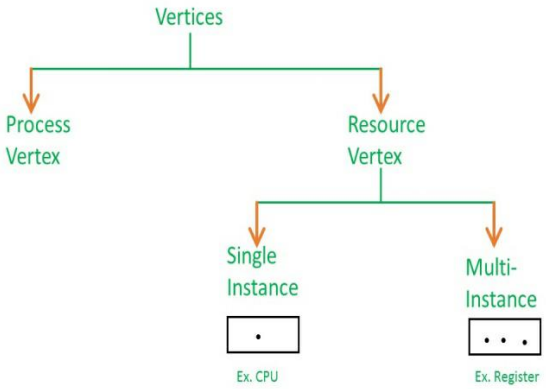
are allocated to other processes, so that their is a possibility of recovering the system from deadlock. In this case system go into starvation.

3. race condition:B. results when several threads try to access and modify the same data concurrently

8 Describe resource-allocation graph? Explain how resource graph can be used for detecting deadlocks?

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 and 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 two type –
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 type –

- Single instance type resource – It represents as a box, inside the 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.



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

A)

a) A race condition is an undesirable situation that occurs when a device or system attempts to perform two or more operations at the same time, but because of the nature of the device or system, the operations must be done in the proper sequence to be done correctly.

b) All the synchronization techniques we have studied so far are essentially low level, like semaphores. They require the programmer to be intimately involved with all the details of mutual exclusion, critical region management, deadlock prevention, and crash recovery. What we would really like is a much higher-level abstraction, one that hides these technical issues and allows the programmer to concentrate on the algorithms and how the processes work together in parallel. Such an abstraction exists and is widely used in distributed systems. We will call it an atomic transaction, or simply transaction. The term atomic action is also widely used. In this section we will examine the use, design, and implementation of atomic transactions.

c) In simple terms a critical section is group of instructions/statements or region of code that need to be executed atomically (read this post for atomicity), such as accessing a resource (file, input or output port, global data, etc.).

d) 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.

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

Answer:

- The roles in a role-based access control facility are similar to the domain in the access-matrix facility.
- Just as a domain is granted access to certain resources, a role is also granted access to the appropriate resources.
- The two approaches differ in the amount of flexibility and the kind of access privileges that are granted to the entities.
- Certain access-control facilities allow modules to perform a switch operation that allows them to assume the privileges of a different module, and this operation can be performed in a transparent manner. Such switches are less transparent in role-based systems where the ability to switch roles is not a privilege that is granted through a mechanism that is part of the access-control system, but instead requires the explicit use of passwords.

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

Answer:

- 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 and is 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.

12 Explain the following. a) Goals of protection b) Principles of protection

Goals of Protection

- Obviously to prevent malicious misuse of the system by users or programs. See chapter 15 for a more thorough coverage of this goal.
- To ensure that each shared resource is used only in accordance with system *policies,* which may be set either by system designers or by system administrators.
- To ensure that errant programs cause the minimal amount of damage possible.
- Note that protection systems only provide the *mechanisms* for enforcing policies and ensuring reliable systems. It is up to administrators and users to implement those mechanisms effectively.

Principles of Protection

- The *principle of least privilege* dictates that programs, users, and systems be given just enough privileges to perform their tasks.
- This ensures that failures do the least amount of harm and allow the least of harm to be done.
- For example, if a program needs special privileges to perform a task, it is better to make it a SGID program with group ownership of "network" or "backup" or some other pseudo group, rather than SUID with root ownership. This limits the amount of damage that can occur if something goes wrong.
- Typically each user is given their own account, and has only enough privilege to modify their own files.
- The root account should not be used for normal day to day activities - The System Administrator should also have an ordinary account, and reserve use of the root account for only those tasks which need the root privileges

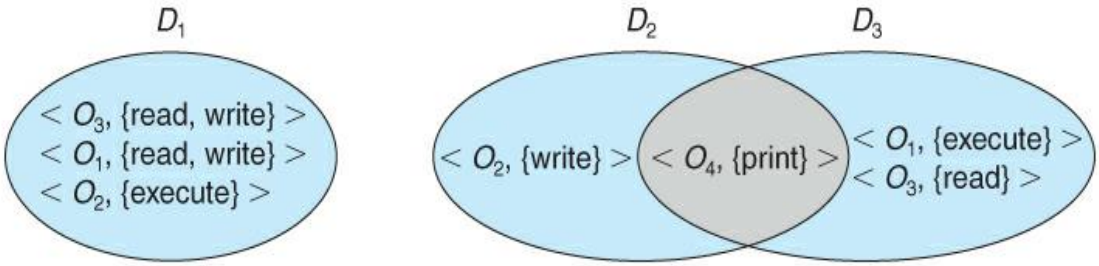
13 Discuss about domain of protection?

Domain of Protection

- A process operates within a protection domain, which specifies the resources that the process may access.
- Each domain defines a set of objects and the types of operations that may be invoked on each object.
- The ability to execute an operation on an object is an access right. „
- A domain is a collection of access rights, each of which is an ordered pair: „ Example: If domain D has the access right: , then a process executing in domain D can only read and write file F.

Domain structure:

- Access-right = where rights-set is a subset of all valid operations that can be performed on the object. „
- Domain = a collection of access-rights „
- A protection domain specifies the resources that the process may access



- Domains may share access rights.
    - A process executing in either D2 or D3 can print O4
    - A process must be executing in D1 to read and write O1. Also, only process in D3 may execute O1

14 Why do you need to provide protection to the system? Explain how access matrix can be used for the purpose?

Access Matrix

- The model of protection that we have been discussing can be viewed as an ***access matrix,*** in which columns represent different system resources and rows represent different protection domains. Entries within the matrix indicate what access that domain has to that resource.

object / domain	F_1	F_2	F_3	printer
D_1	read		read	
D_2				print
D_3		read	execute	
D_4	read write		read write	

Figure 14.3 - Access matrix.

15 Discuss the access matrix implementation techniques?

Implementation of Access Matrix

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.

16 Compare the various access matrix implementation techniques?

Many trade-offs to consider

- Global table is simple, but can be large
- Access lists correspond to needs of users
    - Determining set of access rights for domain non-localized so difficult
    - Every access to an object must be checked – Many objects and access rights -> slow
- Capability lists useful for localizing information for a given process.
    - But revocation capabilities can be inefficient
- Lock-key effective and flexible, keys can be passed freely from domain to domain, easy revocation

17 Discuss the various issues that need to be considered through the process of revocation of access rights?

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?

18 Explain various schemes to implement revocation for capabilities?

- 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.

19) Explain how language-based protection scheme can be used for providing system protection at kernel level?

In a language-based protection system, the kernel will only allow code to execute that has been produced by a trusted language compiler. The language may then be designed such that it is impossible for the programmer to instruct it to do something that will violate a security requirement.[14]

Advantages of this approach include:

No need for separate address spaces. Switching between address spaces is a slow operation that causes a great deal of overhead, and a lot of optimization work is currently performed in order to prevent unnecessary switches in current operating systems. Switching is completely unnecessary in a language-based protection system, as all code can safely operate in the same address space.

Flexibility. Any protection scheme that can be designed to be expressed via a programming language can be implemented using this method. Changes to the protection scheme (e.g. from a hierarchical system to a capability-based one) do not require new hardware.

Disadvantages include:

Longer application start up time. Applications must be verified when they are started to ensure they have been compiled by the correct compiler, or may need recompiling either from source code or from bytecode.

Inflexible type systems. On traditional systems, applications frequently perform operations that are not type safe. Such operations cannot be permitted in a language-based protection system, which means that applications may need to be rewritten and may, in some cases, lose performance.

Examples of systems with language-based protection include JX and Microsoft's Singularity.

20) Explain relative merits of compiler-based enforcement based solely on a kernel, as opposed to enforcement provided largely by a compiler?

Compiler-Based Enforcement:

- In a compiler-based approach to protection enforcement, programmers directly specify the protection needed for different resources at the time the resources are declared.

This approach has several advantages:

- Protection needs are simply declared, as opposed to a complex series of procedure calls.
- Protection requirements can be stated independently of the support provided by a particular OS.
- The means of enforcement need not be provided directly by the developer.
- Declarative notation is natural, because access privileges are closely related to the concept of data types.

PART-C

1)

2) 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.

Answer: The following rule prevents deadlock: when a philosopher makes a request for the first chopstick, do not grant the request if there is no other philosopher with two chopsticks and if there is only one chopstick remaining.

3) Consider a system consisting of m resources of the same type beingshared by n processes. A process can request or release only one resourceat a time. Show that the system is deadlock free if the following twoconditions hold:

Let:N = summation of all Needi

A = summation of Allocationi

M = summation of all Maxi

a.
The maximum need of each process is between one resource and m resources. If the system is assumed to not be deadlock free and there exists a deadlock state, then A=M be cause there is only one resource which can be requested/released one at a time.

b. The sum of all maximum needs is less than m + n.

In this condition, M < m+n = N+A,

which is the sanme as N+m < m+n

so N < n.

From condition a, this process can release at least one resource, so there are n-1 processes sharing n resources at this point, and both conditions a and b still hold true. N o processes will wait permanently so there is no deadlock

4) How does the principle of least privilege aid in the creation of protection systems?

Answer: The principle of least privilege allows users to be given just enough privileges to perform their tasks. A system implemented within the framework of this principle has the property that a failure or compromise of a component does the minimum damage to the

system since the failed or compromised component has the least set of privileges required to support its normal mode of operation

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

Answer: When a Java thread issues an access request in adoPrivileged() block, the stack frame of the calling thread is annotated according to the calling thread's protection domain. A thread with an annotated stack frame can make subsequent method calls that require certain privileges. Thus, the annotation serves to mark a calling thread as being privileged. By allowing a Java program to directly alter the annotations of a stack frame, a program could potentially perform an operation for which it does not have the necessary permissions, thus violating the security model of Java.

7) A system has n resources R0,...,Rn-1,and k processes P0,....Pk-1.The implementation of the resource request logic of each process Pi is as follows:

```
 if (i % 2 == 0)
{
 if (i < n) request Ri
if (i+2 < n) request Ri+2
 }
else
{
 if (i < n) request Rn-i
if (i+2 < n) request Rn-i-2
}
```

Answer:No. of resources, n = 21

No. of processes, k = 12

Processes {P0, P1....P11} make the following Resource requests:

{R0, R20, R2, R18, R4, R16, R6, R14, R8, R12, R10, R10}

For example P0 will request R0 (0%2 is = 0 and 0< n=21).

Similarly, P10 will request R10.

P11 will request R10 as n - i = 21 - 11 = 10.

As different processes are requesting the same resource, deadlock

may occur.

8) 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?

if 6 resources are there then it may be possible that all three process have 2 resources and waiting for 1 more resource. Therefore, all of them will have to wait indefinitely. If 7 resources are there, then atleast one must have 3 resources so deadlock can never occur.

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

- We have 6 Resources
- Allocate 1 resourse to each
- 2 Resoueses will be available

	MAX	Allocated	Need
p1	2	1	1
p2	2	1	1
p3	2	1	1
p4	2	1	1

Available (2)

Using Bankers Algorithm , This Requests can be satisfied in any order . The lowest value of N for which the system will be in deadlock is 6 .

10) Two shared resources R1 and R2 are used by processes P1 and P2. Each process has a certain priority for accessing each resource. Let Tij denote the priority of Pi for accessing Rj. A process Pican snatch a resource Rh from process Pj if Tik is greater than Tjk. Given the following :

1. T11 > T21

2. T12 > T22

3. T11 < T21

4. T12 < T22

Explain which of the following conditions ensures that P1 and P2 can never deadlock?

Answer : T11>T21 and T12>T22