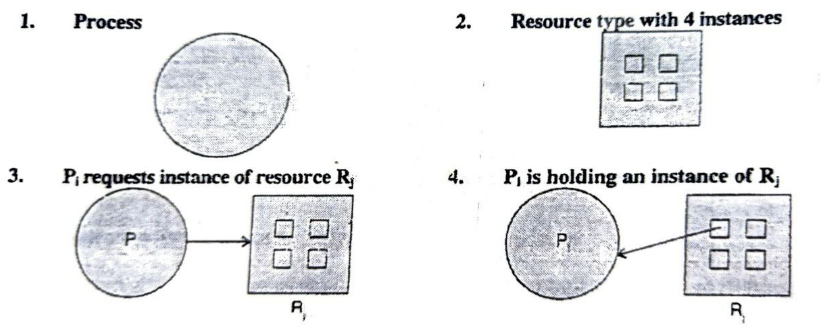
Process Deadlock

- **Deadlock State**: A set of processes are in a deadlock if all of them are waiting for some resources acquired by other processes in the system.
- **System Table**: The OS maintains a table to maintain the state of each resource available on the system free or allocated.
- Necessary condition for Deadlock to occur
    1. **Mutual Exclusion**: When resources are not shared among different processes then it is called as mutual exclusion.
    2. **Hold and Wait**: A process holding one resource and waiting for other resources acquired by some other process.
    3. **No Preemption**: A resource can only be release if the process holding it completed its task.
    4. **Circular Wait**: When process P0 is waiting for resources held by process P1, P1 is waiting for resources held by process P2 and so on and at last process Pn is waiting for resources held by process P0 and none of them are able to complete their tasks then this condition is known as circular wait.
- **Resource Allocation Graph**: Graph which is used to describe deadlock is called as Resource Allocation Graph. It contains set of edges and vertices.

  Pictorial representation:



1. Process
2. Resource type with 4 instances
3. P_i requests instance of resource R_j
P
R_j
4. P_i is holding an instance of R_j
P_i
R_j

💡 Solve Resource Allocation Graph Questions

- **Starvation**: If a process needs several popular resources than it may need to wait indefinitely because at least one resource may be allocated to some other process.

Deadlock Prevention

1. **Mutual Exclusion**: Shareable resources can be immediately allocated to any number of process that request it. A process do not have to wait for the shareable resource and thus mutual exclusion can be avoided.

   Other way to avoid mutual exclusion is to avoid assigning of resources to processes if its not necessary and try to make sure that only a few processes claim the resource.

2. **Hold and Wait**: To avoid hold and wait condition we have to make sure that if a process request for a resource it does not hold any other resource.
3. **No Preemption**: If a process is holding some resources and request some resources and the requested resources cannot be allocated to the process immediately and all held resources must be preempted. In this way we can preempt held process and allocate them to requesting processes.
4. **Circular Wait**: To create a rule saying that a process is entitled to a single resource at any given time.

Deadlock Avoidance

- One way is that each process declares different resource types and maximum number of each type of resources it will need during its execution. This leads to lower resource utilization.
- Another way is that deadlock avoidance algorithm(Banker’s Algorithm) dynamically examines and ensure that there can never be any circular wait condition.
- **Resource-Allocation State**: It is defined by the number of available and allocated resource and the maximum demands of the processes.
- **Safe State**: A state is safe if system can allocate maximum number of resources to processes and still avoid deadlock.
- **Safe Sequence**: For current allocation state a sequence of processes is a safe sequence if for each process Pj can be still satisfied by the currently available resources plus the resources held by all other process pi, with j < i.

  A system is in safe state only if there exists a safe sequence.

💡 Solve Banker’s Algorithm Questions

Deadlock Detection

- If the system contains a single instance of each resource type, a wait-for-graph is used for deadlock detection.
- It is similar to resource allocation graph and is obtained from the resource allocation graph by removing the resources nodes and collapsing the appropriate edges.

Deadlock Recovery

1. Process Termination
    a. Abort all deadlocked process
        - It is very expensive as the process which are terminated might have been running for a long period of time.
        - The partial results calculated by process will have to be discarded and recalculated again.

    b. Abort one process at a time until the deadlock cycle is eliminated
        - This adds additional overhead as after aborting each process we have to check if deadlock is present or not and for this a deadlock detection algorithm needs to be run.
        - Another disadvantage is if the aborted process is in the middle of some file updation then it might leave file in some inconsistent manner.
2. Resource Preemption
    a. **Selecting a Victim**: To recover from a deadlock the resources needs to be preempted and its done by selecting a victim. Victim is selected by
        i. Number of resources a deadlocked process is holding.
        ii. Amount of time a process has consumed in its execution.

    b. **Rollback**: The process must be rollbacked to a safe state and start its execution from there but sometimes it is hard to determine what a safe state is so we do a total rollback i.e. terminate a process and than restart it.

c. **Starvation**: It may happen that the same process is getting selected every time for abortion and the same resource is getting selected for preemption which can lead to starvation, to avoid this we must ensure that a process can be picked only a finite number of time.

File System Management

- **What is a file?**
    - A file is an abstract data type.
    - It is logical storage unit defined by OS.
    - It is a sequence of bits, bytes, lines or records.
    - It can be text file, an executable file or a source file.

File Attributes

- **Name**: This is the only information kept in human readable form.
- **Identifiers**: It is a number which is used to identify the file in the file system.
- **Type**: It is used to support different types.
- **Location**: This is pointer to file location on device.
- **Size**: It gives current size of the file.
- **Protection**: Provides access on who can read, write and execute certain function.

File Operations

- **Creating a File**: First we find space in the file system and then we create an entry for the new file.
- **Writing a File**: We need to pass file name and information to be written.
- **Reading a File**: We use file name and buffer location to read from a file.
- **Repositioning within File**: Current file pointer position is repositioned to a given value.
- **Deleting a File**: By giving a name the directory is searched and if file is found all file space is erased.
- **Truncating a File**: Keeping all the attributes of the file and just erasing content of the file.

Some other file operations

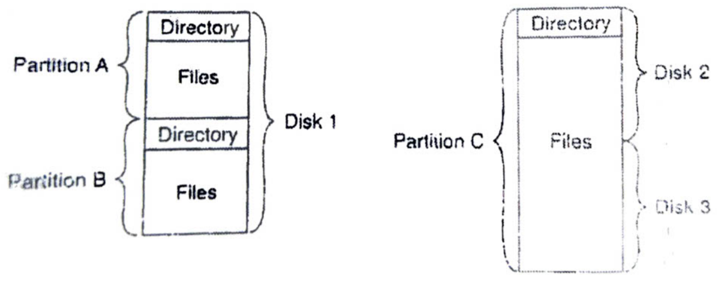
- **Open**: To open a file to read or write to the file.
- **Close**: When file is no longer being actively used it is closed by the process.
- **Append**: To add some new information at the end of the existing file.
- **Rename**: To rename existing file.
- **Copy**: We can create a copy of the file.

Access Methods

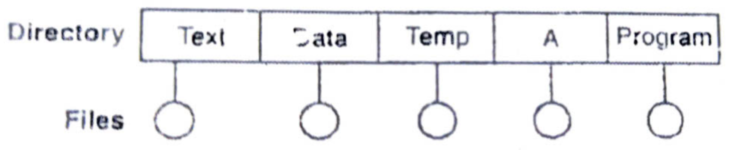
1. Sequential Access
    - Entire file is read or written from beginning to end sequentially.
    - Simplest access method.
    - Read operation reads the next block of the file and automatically advances a file pointer.
    - Write operation appends to the end of the file and the pointer points to the new end of the file.
2. Direct Access
    - Known as relative access.
    - It allows records i.e. information to read/write faster in no particular order.
    - We may read block 20 then block 13 and may write block 53.

Directory Structure

1. Storage Structure
    - A disk can be partitioned into parts which are known as partitions or minidisks. These are used for file system and have other uses.
    - A part of storage that holds file system is known as volume, these are also known as virtual disks.

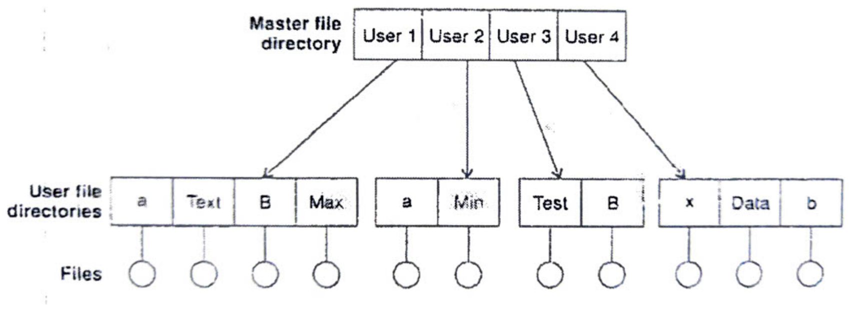


2. Directory Overview
   - Directory view translates file names into their directory entries.
   - Some operations which can be performed to the directory:
     1. **Search for a file**: We have to search the directory for the entry.
     2. **Create a file**: Newly created file are added to the directory.
     3. **Delete a file**: When we delete a file its entry must also be deleted from the directory.
     4. **List a directory**: List the entry for each file in a directory.
     5. **Rename a file**: Renaming a file may allow its position to change in the directory.
     6. **Traverse a file system**: We can access every directory and every file within a directory structure.
3. Single Level Directory
   - All files are kept in the same directory.
   - This type of structure is easy to support and understand.
   - Some limitations are:
     - When the number of files increases they all must have unique names.
     - Difficult to remember the names.



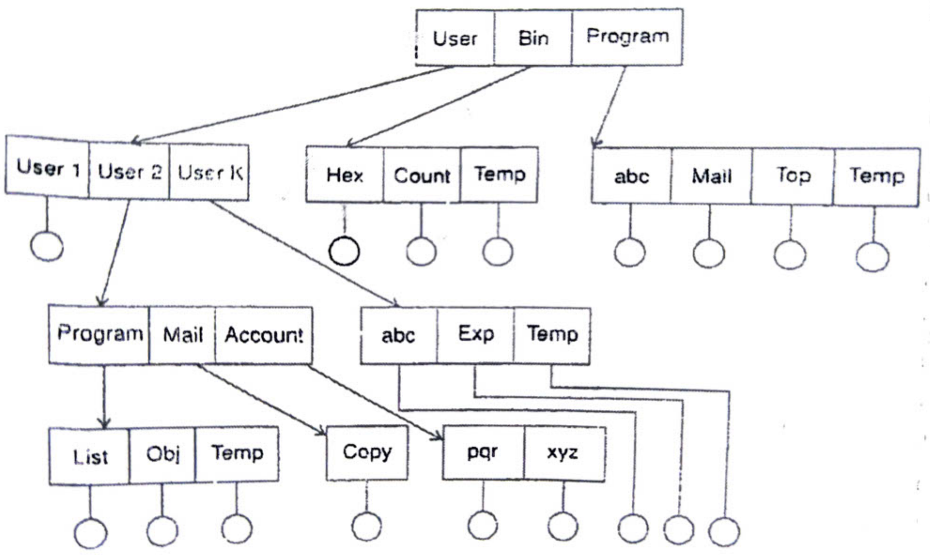
4. Two Level Directory

- Each user has separate directory.
- Two levels are : Master File Directory(MFD) and User File Directory(UFD). Where each user has their own UFD.
- Two level directory can be thought as tree with height 2 where MFD is root and children are UFD.



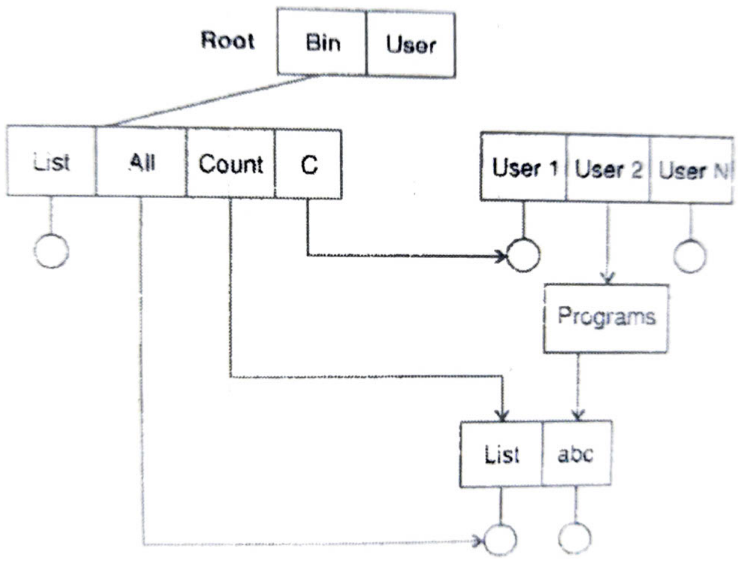
5. Tree Structure Directory

- This tree structure allows users to crate their directories and subdirectories to organize their files. This structure can have arbitrary tree height.
- A tree is most common directory structure. A tree has a root directory and every file in the system has a unique path name.
- Limitation:
    - For deletion of directory in a tree structure, special policy is required. The directory which is to be deleted is emptied then simply its entry in the directory is deleted.

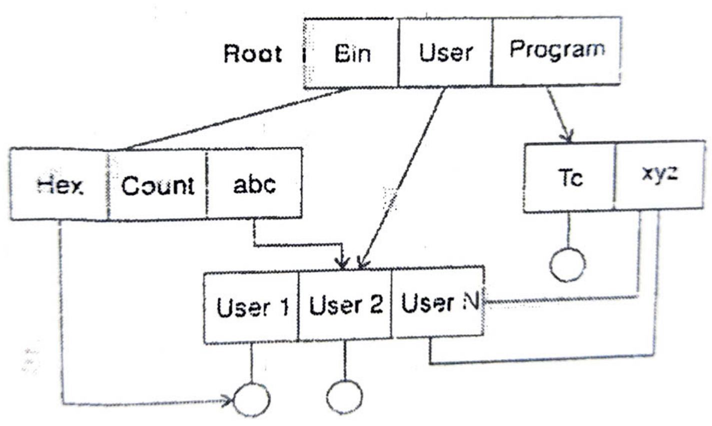


6. Acyclic Graph Directory

- This directory allows directories to share subdirectories and files i.e. A file or a subdirectory may be in a two different directories.
- Any change made by a user immediately appears to the other users. If a file is created in a shared subdirectory it automatically appears in all other shared subdirectories.
- This approach is useful when more than one programmer works on a joint project.
- Implementation:
    - In UNIX system, it creates a new directory entry called a link.
    - A link is a pointer to another file or subdirectory.
    - An absolute or relative path is used in link implementation.
- Issues:
    - More complex than a tree structured directory.
    - A file may have multiple absolute path names.
    - If a file is removed it may lead to dangling pointers to the now non-existent file.
    - If the space is reused then these dangling pointers may point to the middle of some other files or may point to wrong file.



7. General Graph Directory
    - Major drawback of acyclic graph structure is ensuring that there are no cycles and this is removed by general graph directory.
    - Cycles are allowed in general graph directory.
    - While searching we wish to avoid any component twice for correctness and performance of result.
    - If algorithm is not effectively designed it might result in an infinite loop and searching may never terminates. We can limit directories that can be accessed during search to avoid this.



Allocation Methods

- **External fragmentation**: It exists whenever free space is broken into parts of chunks. Storage is fragmented into a num er of holes, no hole of which is large enough to store the data.

1. Contiguous Allocation
    - Each file occupies a set of contiguous blocks on the disk. Disk addresses are defined linearly. Here assumption made is only one job is accessing the disk. when job wants to access data it accesses disk contiguously.
    - Both sequential and direct access can be supported by it.
    - Disadvantages:
        - Finding space for a new file. Various strategies are used to deal with this but all of them suffers from external fragmentation.
        - Space required for a newly created file should be known in advance.

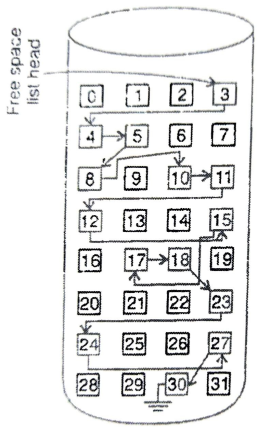
2. Linked Allocation
    - Each file is a linked list of disk blocks which ay be scattered anywhere on the disk.
    - The direction entry for the file contains a pointer to the first and last block of the file.
    - Each block contains a pointer to the next block. These pointers are hidden from the user.
    - Advantages:
        - There is no external fragmentation.
        - File can grow as much as user wants.
        - File size doesn’t need to be known in advance.
    - Disadvantages:
        - Only used for sequential access of files.
        - Space is required for the pointers.
        - Reliability. If a pointer is lost or damaged, we may pick up wrong pointer or no pointer could be available while accessing the file.
3. Indexed Allocation
    - Linked allocation solves the problem of external fragmentation and size declaration but without a FAT, linked allocation cannot support efficient direct access.
    - In index allocation, all pointers are kept together into the index table.
    - Each file has its own index block. An index block is an array of disk block addresses. The 1st entry in the index block points to the 1st block of the file and so on.
    - Advantages:
        - It does not suffers from external fragmentation.
        - It does not require initial size declaration.
        - It supports direct access to the file.
    - Disadvantage:

    - Memory required to store index block is more than that of linked allocation.

Free Space Management

1. Bit Vector
    - Each block is represented by one bit. If block is free then bit 1 is used and if it is allocate bit 0 is used.
    - Example: Consider a disk where 3, 4, 5, 8, 10 are free and the rest blocks till 12 are allocated.

      The bit vector would be 0001110010100
2. Linked List
    - All the free blocks are linked together. Pointer to the first free block is kept in a special location and cached in the memory.
    - First block contains a pointer to the 2nd free block and so no, works just like a pointer.
    - Whenever operating system needs to allocate free block it uses the first block in the free list.



3. Grouping
    - Address of the n free blocks are stored in the first free block. The last nth block contains the addresses of another n free blocks and so on.
    - Traversing a list is avoided and large number of free blocks can be found quickly.
4. Counting

- When space is allocated with contiguous memory allocation several contiguous blocks may be allocated or freed simultaneously.
- Address of the first free block and number of free contiguous block that follow the first block are stored.

5. Space Maps
    - ZFS creates metaslabs to divide the space on the device into manageable chunks. A volume may contain hundreds of metaslabs. Each metaslab has an associated space map.
    - ZFS uses counting algorithm to store information about free blocks.
    - Space map is a log of all block activity in time order and counting format.
    - When ZFS has to allocate memory it loads the associated space map into the memory.
    - Space maps uses balanced tree structure.

Disk Scheduling

- **Disk Bandwidth**: It is total number of bytes transferred divided by time difference between first request and the completion of the last transfer.

▼ Working of Disk Schedular

- Access time and bandwidth can be improved by managing the order in which the disk I/O requests are serviced. Whenever a process needs I/O to or from the disk, it issues a system call to the OS.
- Request can be serviced immediately if desired disk drive and controller are available.
- If the drive or controller is busy, any new requests for will be placed in pending queue.
- For multiprogramming, disk often have several pending requests. System choice to service a new request is based on several disk scheduling algorithm.

▼ Converting logical block number to disk address

- Logical blocks are mapped onto the sectors of the disk sequentially. Sector 0 is the first sector of the track and on the outermost cylinder.
- Mapping proceeds through the rest of the track in that cylinder, and then through the rest of the cylinders from outermost to innermost.
- This mapping provides the disk address which consists of a cylinder number, a track number and a sector number.

Disk Performance Parameters

- **Disk Partitioning**: Management of disk partitions affects the overall performance of the disk. Creating too many partitions may cause a decrease in overall disk performance. It increases performance when data types are segregated in different partitions.
- **Drive Form Factor**: It specifies the disk’s total storage capacity and the corresponding number of platters used within it. It also determines the disk’s transfer rate.

- **Spindle Speed**: Spindle speed on which the disk rotate determines the speed of the hard drive in RPMs, this determines data access and retrieval rates.
- **Platter Diameter**: Diameter of the platter affects the drive's data transfer rates. Larger the diameter of the platter higher the transfer rates.
- **Access Times**: It is the time needed for magnetic head to re-position itself to the appropriate platter for any read or write request. It depends upon write seek time, read seek time, full stroke time and track to track.
- **Seek Time**: Time it takes to moves the disk arm to the required cylinder.
- **Rotational Latency**: Time it takes the disk to rotate so that the required sector is above the disk head.

Disk Scheduling Algorithms

1. FCFS
    - Simplest and easy in computation.
    - **Working**:
        1. Identify the current head position.
        2. Select the request which have arrived first.
        3. Move the disk head to selected positioning.
        4. Repeat steps 2 and 3 until disk queue is empty.
        5. When disk queue is empty, calculate total head movements.
    - **Problems**:
        1. Does not provide the fastest service.
        2. If disk head is at the one end and next request is at the other end, the total seek time is increased and thereby performance is decreased
        3. If requests arrive non-consecutively, they will have to be traversed separately.
2. SSTF
    - It chooses the pending request closest to the current head position. This is called as Shortest Seek Time First(SSTF) algorithm.
    - **Working**:

    1. Identify the current head position.
    2. Select the request which is at minimum seek time from the current head position.
    3. Move the disk head to selected positioning.
    4. Repeat steps 2 and 3 until disk queue is empty.
    5. When disk queue is empty, calculate total head movements.
- **Problems**:
    1. It may cause starvation of some requests.
    2. Not optimal compared to some other algorithms.

3. SCAN
    - Disk arm starts at one end of the disk and it moves toward the other end.
    - While moving it services the requests until it get to the other end of the disk.
    - At the other end, the direction of the head is reversed and servicing continuous.
    - Head continuously scans back and forth until all the requests are serviced.
    - SCAN algorithm is also known as elevator algorithm.
    - Initial direction of the disk arm is important factor.
    - **Working**:
        1. Identify the current head position and current direction of the disk arm.
        2. Select the requests which arrives immediately in the direction of the disk arm.
        3. Move the head to the selected position.
        4. Move the head in the same direction till all it reaches disk end then reverse the direction.
        5. Repeat steps 2, 3 and 4 till the disk queue is empty.
        6. When disk queue is empty, calculate total head movements.
    - **Problems**:
        1. A request arriving just behind the head will have to wait until the arm moves to the end of the disk, reverses direction and comes back.

2. Relatively few requests will arrived in front of the head, since these cylinders have recently been serviced. The heaviest density of requests is at the other end of the disk. These requests will have to wait the longest.

4. C-SCAN
    - It provides a more uniform wait time.
    - When head reaches the end, it immediately returns to the beginning of the disk without servicing any request on the return trip.
    - This algorithm treats the cylinders as a circular list.
    - Here, the direction is always from 0 to the end of the cylinder.
    - **Working**:
        1. Identify the current head position.
        2. Select the requests which arrives immediately in the direction of the disk arm.
        3. Move the head to the selected position.
        4. Move the head in the same direction till all it reaches disk end then immediately return the head to the beginning of the disk.
        5. Repeat steps 2, 3 and 4 till the disk queue is empty.
        6. When disk queue is empty, calculate total head movements.
5. LOOK
    - LOOK scheduling is a version of SCAN that follows this pattern, because it *looks* for a request before continuing to move in a given direction
    - The difference between SCAN and LOOK is that SCAN traverses till the end in both the directions even if there are no requests at the end and reverses back. LOOK traverses in either direction till there are requests in the direction it is traversing. If there are no further requests, it reverses back and travers the requests in the opposite direction.
    - Direction of the disk arm is important.
    - **Working**:
        1. Identify the current head position and current direction of the disk arm.

2. Select the requests which arrives immediately in the direction of the disk arm.
3. Move the head to the selected position.
4. Move the head in the same direction till there are requests in the queue in the same direction.
5. For each request found repeat steps 2, 3 and 4.
6. If there are no further requests in that direction, reverse the direction of the head and repeat steps 2, 3, 4 and 5.
7. When disk queue is empty, calculate total head movements.

6. C-LOOK
    - LOOK scheduling is a version of SCAN that follows this pattern, because it *looks* for a request before continuing to move in a given direction
    - The difference between C-SCAN and C-LOOK is that C-SCAN traverses till the end in both the directions even if there are no requests at the end and goes to the starting of the disk. C-LOOK traverses in either direction till there are requests in the direction it is traversing. If there are no further requests, it goes straight to the starting of the disk.
    - This algorithm treats the cylinders as a circular list.
    - Here, the direction is always from 0 to the end of the cylinder.
    - **Working**:
        1. Identify the current head position.
        2. Select the requests which arrives immediately in the direction of the disk arm.
        3. Move the head to the selected position.
        4. Move the head in the same direction till there are requests in the queue in the same direction.
        5. For each request found repeat steps 2, 3 and 4.
        6. If there are no further requests in that direction, immediately return to the beginning of the disk and repeat steps 2, 3, 4 and 5.
        7. When disk queue is empty, calculate total head movements.

💡 Solve Disk Scheduling Algorithm’s Questions

Introduction to Distributed Operating Systems and Architecture

- **Distributed System**: It is a collection of autonomous computing elements that appears to its users as a single coherent system.
- Benefits of Distributed Systems:
    1. **Scaling**: Distributed system can scale horizontally and can account for more traffic.
    2. **Modular Growth**: No cap on how much modules can be added.
    3. **Fault Tolerance**: More fault tolerant than a single machine.
    4. **Cost Effective**: Because of scalability, it becomes more cost effective.
    5. **Low Latency**: Traffic will hit nearest node providing low latency system.
    6. **Efficiency**: It breaks complex data into smaller pieces.
    7. **Parallelism**: Multiple processes divide complex problem into pieces.

Design Goals

1. **Resource Sharing**:
    - An important goal of distributed system is to make it easy for users to access and share the remote resources.
    - Motivation behind sharing a resource is to make the system economically cheaper.
2. **Distribution Transparency**:
    - Main goal is to hide the fact that its resources and processes are physically distributed. This is achieved by making distribution invisible to the end users and applications.

    a. **Access Transparency**: There should be no apparent difference between local and remote access methods i.e. explicit communication may be hidden.

b. **Location Transparency**: Topology of the system is of no concern to the user. Location of an object in the system may not be visible to the user.

c. **Concurrency Transparency**: Users and applications should be able to access shared data without interference between each other.

d. **Replication Transparency**: If the system provides replication it should not concern the user.

e. **Fault Transparency**: If a system failure occurs then these should be hidden from the user. Partial failure of the system is possible and this may not be reported.

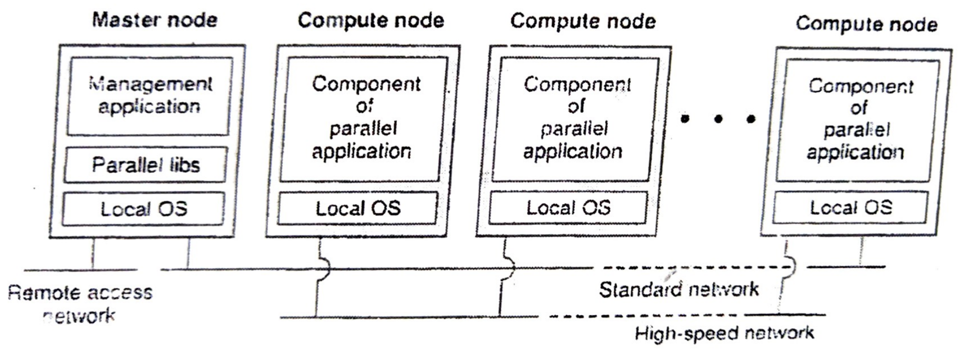
f. **Migration Transparency**: If data is migrated to provide better performance or reliability then this should be hidden from the user.

3. **Openness**:
    - In open distributed system the components can easily be used by or integrated into other systems and they can easily interact with other open systems.
    - Another goal is that it should be easy to configure the system out of different components.
    - An open distributed system should also be **extensible**.
4. **Scalability**:
    - We can measure the scalability of a distributes system in three main ways. These three forms are often referred to as **scalability dimensions**.

    1. **Size Scalability**: A system can be scalable with respect to its size, meaning that we can easily add more users and resources to the system
    2. **Geographical Scalability**: No matter the distance between user and the resources the system should function efficiently.
    3. **Administrative Scalability**: Adding more nodes into a system shouldn’t require a significant uptick in terms of managing those new nodes.

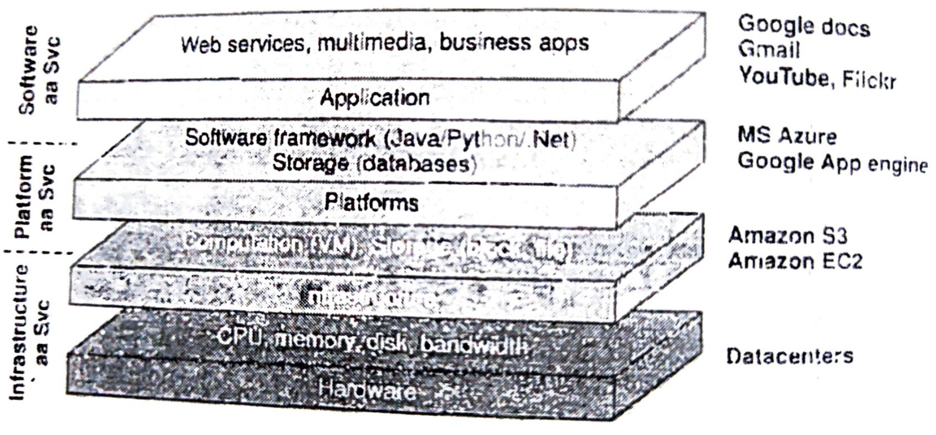
Distributed Computing Systems

1. **Cluster Computing System**:



- A cluster computer refers to a network of the same type of computers whose target is to work as the same unit.
- Two or more same types of computers are clubbed together to make a cluster. In this computing system nodes or computers has to be of same types, like same CPU, same OS.
- It is used when a resource hungry task requires high computing power or memory.
- Computers of cluster computing are dedicated to a single task.
- They are co-located and are connected by high speed network bus cables.
- They are prepared using centralized network topology.

2. **Cloud Computing System**:



- It is a parallel computing system consists of inter-connected computers.
- Clouds are a large pool of resources such as hardware, platforms and services.

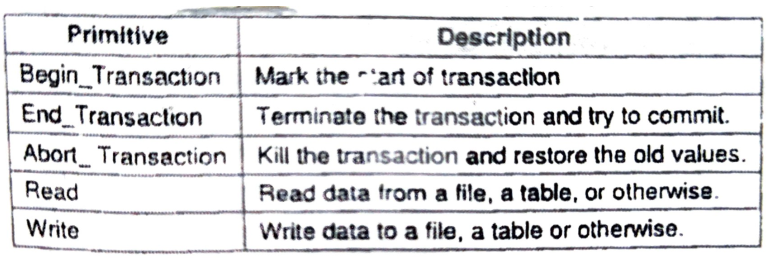
- Cloud computing refers to a client-server computing where resources are managed in a centralized fashion.
- It is more flexible than Grid computing.
- Users pay for using the cloud computing resources. They need not set up anything. They use the platform as a service.

3. **Grid Computing System**:
    - Grid computing refers to a network of the same or different types of computer whose target is to perform a task by multiple computers together as per the need.
    - In this system each computer may belong from a different domain and may be heterogeneous in nature.
    - In this system each computer can work independently. Grid computer can have homogeneous or heterogeneous network.
    - Grid computing network is distributed and has a de-centralized network topology.
    - In grid computing multiple servers can exist. Each node behaves independently without need of any centralized scheduling server.

Distributed Information Systems

- It is another class of distributed systems used in organizations which are taking advantage of networked applications but for which inter-operability becomes difficult to achieve.
- Clients can wrap a number of requests into a single larger request and have it executed as distributed transaction.

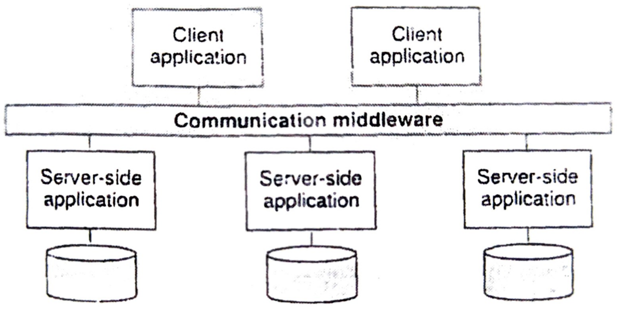
1. **Transaction Processing Systems**:
    - Database operations are carried out as **transactions**.
    - Special primitives are required for the execution of transactions. Typical examples of transaction primitives are shown in the below table:



Primitive	Description
Begin_Transaction	Mark the start of transaction
End_Transaction	Terminate the transaction and try to commit.
Abort_ Transaction	Kill the transaction and restore the old values.
Read	Read data from a file, a table, or otherwise.
Write	Write data to a file, a table or otherwise.

- A key feature of a transaction is either all of these operations are executed or none. All transactions stick to the ACID properties.

2. **Enterprise Application Integration**:
   - Application components should be able to communicate independently and directly with each other.



   - The above figure shows that how existing applications can exchange information. There are many different communication models available to fulfill the need for inter-application communication.

Distributed Pervasive Systems

- A pervasive system domes with many **sensors** to pick up various aspects of a user's behavior.
- To provide information and feedback pervasive systems may have large number of **actuators**.
- Pervasive devices are smaller, mostly battery powered, connected with the wireless connection, mobile.

1. **Ubiquitous Computing Systems**:
   - This type of system is continuously present and its elements are spread through many parts of the environment and user is continuously interacting with the system knowingly or unknowingly.

- Core requirements for this system:
    - Distribution
    - Interaction
    - Context awareness
    - Autonomy
    - Intelligence

2. **Mobile Computing Systems**:
    - Mobility is an important component of the this type of system. Mobile device vary widely, such as smart phones, tablet computers, remote controls, pagers, car equipment, GPS-enabled devices as so on.
    - The location of these devices keeps changing over the period of time. These changes in location may lead to many issues like services available at that location.
3. **Sensors Network**:
    - Sensors network consists of tens to hundreds or thousands of relatively small nodes. Each node is equipped with one or more sensing devices. These nodes can often act as actuators.
    - Many sensor networks are battery powered and use wireless communication.
    - It has a software layer which offers services like low level network access, access to sensors and actuators, memory management and so on.

Architectural Styles

- Distributed systems are complex in design. It is necessary that these systems need to be properly organized.
- The software architectures deal with how the various components are to be organized and how they should interact.
- A component is a modular unit with a well defined and provided replaceable interface within its environment.
- Style defines the way that component are connected to each other, how data is exchanged between them and how they are configured into the system.

1. **Layered Architecture**:
    - It is based on the idea that components are organized in a layered way where a component at a layer can make a **downcall** to a component at lower level and expect a response. An **upcall** is made to higher level component in some exceptional cases.

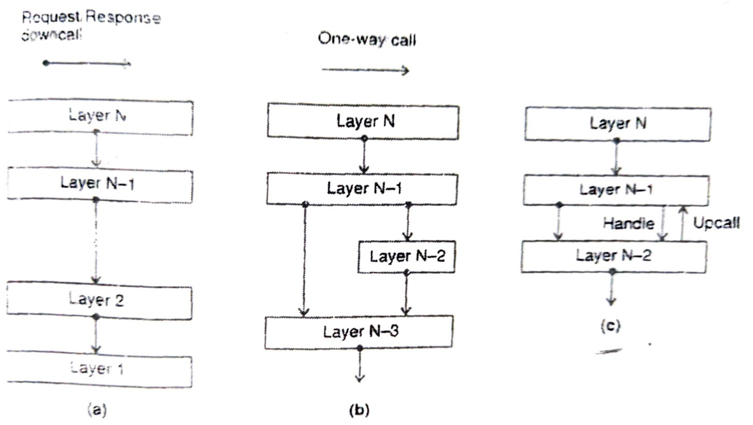
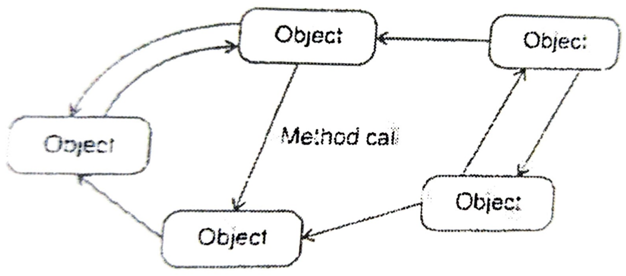


Figure 4.7: (a)Pure layered architecture (b) Mixed layer architecture (c) Layered architecture with upcalls (Adopted from [Krakowlak, 2009])

    - Applied layered architecture which is found everywhere is the communication protocol stack.
2. **Object-Based Architecture**:
    - Each object is defined as a component that is connected through a procedure call mechanism.
    - Below figure shows the organization followed in object based architecture.



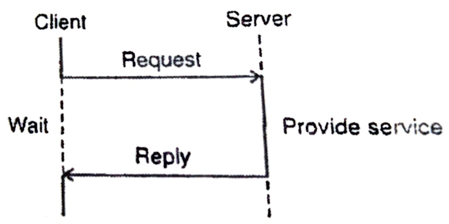
    - Object based architectures provide an object’s state that is the encapsulation of data and operations which can be performed on that data into a single entity.
3. **Resource Based Architecture**:

- Allowing the distributed system to view as a huge collection of resources that are individually managed by components. Remote applications add, remove or even modify the resources. this approach is know as Representational State Transfer (REST).
- Characteristics of RESTful architecture:
    - Resources are identical through a single naming scheme.
    - All services offer the same interface.
    - Messages sent or receive are fully self-described.
    - After executing an operation to a service, that component doesn't keep information about the caller. This is also known as stateless execution.

System Architecture

- System architecture decided about software components, their interaction and their placement.

1. **Centralized Organization**:
    - This model processes in a distributed system that are divided into two groups namely Clients and Servers.
    - A server is a process of implementing a specific service. A client is a process that requests a service from the server. A client sends requests and waits for the response from the server.
    - This **request-response** behavior is shown in the figure.



    - This communication can use connectionless or connection oriented protocol as per the application.

2. **Decentralized Organization**:
    - More often the distribution of the clients and the servers may be split up into logically equivalent parts. Each part is operating on its own of the complete

data set to balance the load. This is known as peer to peer system. In peer to peer system, all processes are equal.

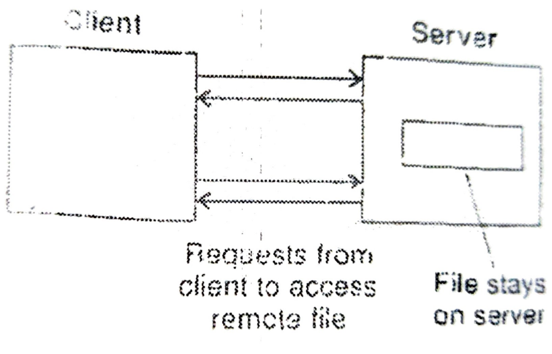
- Each process will act as a client and a server at the same time resulting in interaction between processes is symmetric.

3. **Hybrid Organization**:
    - Many distributed systems combine the features of client server, and peer to peer organizations.
    - **Edge-server System**: These systems are deployed on the internet where servers are placed "at the edge" of the network. This forms a boundary between an enterprise network and the actual internet.

Network File System(NFS)

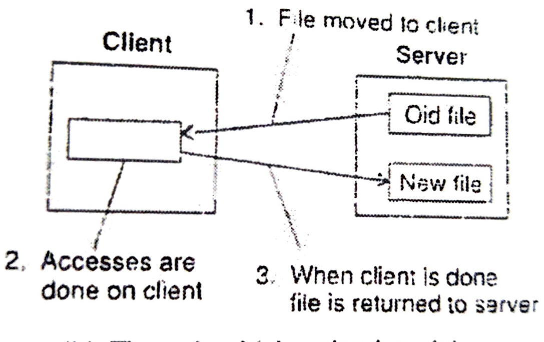
- In this system, each file server provides a standardized view of its local file system. In other words each NFS server supports the same mode, irrespective of the implementation of local file.
- NFS uses a protocol that allows clients to access the files stored on a server. It also allows heterogenous collection of processes, hardware to share common file systems.

1. **Remote File Service**: Without knowing its actual location clients can access files through an interface contain various file operations. The server is responsible for implementing those operations.



(a) The remote access model

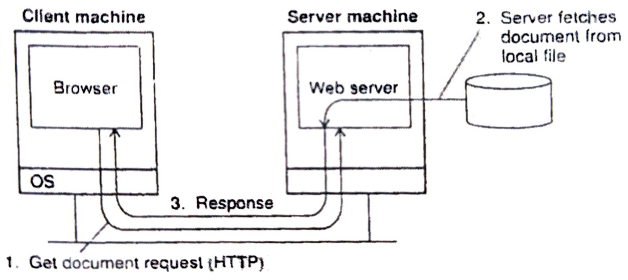
2. **Upload/Download Model**: In this model, clients access a file locally after having downloaded it form the server. Once the use of the file is done, it is uploaded back to the server again for other clients to use it.


1. File moved to client
Client
Server
Old file
New file
2. Accesses are done on client
3. When client is done file is returned to server

(b) The upload / download model

The Web-Based Systems

- Web site is formed b y a process that has access to the documents stored in at local file system. Documents are referred by giving a reference called Uniform Resource Locator (URL).
- Browser is used for interaction with clients and displaying the contents of the documents.


Client machine
Server machine
2. Server fetches document from local file
Browser
Web server
OS
3. Response
1. Get document request (HTTP)