T. Y. B. Sc.
COMPUTER SCIENCE
SEMESTER-VI

NEW SYLLABUS
CBCS PATTERN

OPERATING SYSTEMS-II

Dr. Ms. MANISHA BHARAMBE
Mrs. VEENA GANDHI

NIRALI PRAKASHAN
ADVANCEMENT OF KNOWLEDGE

SPPU New Syllabus

A Book Of

OPERATING SYSTEMS-II

For T.Y.B.Sc. Computer Science : Semester – VI

[Course Code CS 361 : Credits - 2]

CBCS Pattern

As Per New Syllabus, Effective from June 2021

Dr. Ms. Manisha Bharambe

M.Sc. (Comp. Sci.), M.Phil. Ph.D. (Comp. Sci.)

Vice Principal, Associate Professor, Department of Computer Science

MES's Abasaheb Garware College

Karve Road, Pune

Mrs. Veena Gandhi

M. C. S, M.Phil (Computer Science), UGC-NET

Head, Department of BCA Science

Abeda Inamdar Senior College

Pune

Price ₹ 230.00

NIRALI PRAKASHAN

ADVANCEMENT OF KNOWLEDGE

N5946

OPERATING SYSTEMS - II **ISBN 978-93-5451-253-7**

First Edition : January 2022

Published By :
NIRALI PRAKASHAN
Abhyudaya Pragati, 1312, Shivaji Nagar,
Off J.M. Road, Pune – 411005
Tel - (020) 25512336/37/39
Email : niralipune@pragationline.com

Polyplate

Printed By :
YOGIRAJ PRINTERS AND BINDERS
Survey No. 10/1A, Ghule Industrial Estate
Nanded Gaon Road
Nanded, Pune - 411041

DISTRIBUTION CENTRES

PUNE

Nirali Prakashan
(For orders outside Pune)
S. No. 28/27, Dhayari Narhe Road, Near Asian College
Pune 411041, Maharashtra
Tel : (020) 24690204; Mobile : 9657703143
Email : bookorder@pragationline.com

Nirali Prakashan
(For orders within Pune)
119, Budhwar Peth, Jogeshwari Mandir Lane
Pune 411002, Maharashtra
Tel : (020) 2445 2044; Mobile : 9657703145
Email : niralilocal@pragationline.com

MUMBAI

Nirali Prakashan
Rasdhara Co-op. Hsg. Society Ltd., 'D' Wing Ground Floor, 385 S.V.P. Road
Girgaum, Mumbai 400004, Maharashtra
Mobile : 7045821020, Tel : (022) 2385 6339 / 2386 9976
Email : niralimumbai@pragationline.com

DISTRIBUTION BRANCHES

DELHI

Nirali Prakashan
Room No. 2 Ground Floor
4575/15 Omkar Tower, Agarwal Road
Darya Ganj, New Delhi 110002
Mobile : 9555778814/9818561840
Email : delhi@niralibooks.com

BENGALURU

Nirali Prakashan
Maitri Ground Floor, Jaya Apartments,
No. 99, 6th Cross, 6th Main,
Malleswaram, Bengaluru 560003
Karnataka; Mob : 9686821074
Email : bengaluru@niralibooks.com

NAGPUR

Nirali Prakashan
Above Maratha Mandir, Shop No. 3,
First Floor, Rani Jhanshi Square,
Sitabuldi Nagpur 440012 (MAH)
Tel : (0712) 254 7129
Email : nagpur@niralibooks.com

KOLHAPUR

Nirali Prakashan
438/2, Bhosale Plaza, Ground Floor
Khasbag, Opp. Balgopal Talim
Kolhapur 416 012, Maharashtra
Mob : 9850046155
Email : kolhapur@niralibooks.com

JALGAON

Nirali Prakashan
34, V. V. Golani Market, Navi Peth,
Jalgaon 425001, Maharashtra
Tel : (0257) 222 0395
Mob : 94234 91860
Email : jalgaon@niralibooks.com

SOLAPUR

Nirali Prakashan
R-158/2, Avanti Nagar, Near Golden
Gate, Pune Naka Chowk
Solapur 413001, Maharashtra
Mobile 9890918687
Email : solapur@niralibooks.com

marketing@pragationline.com | www.pragationline.com

Also find us on www.facebook.com/niralibooks

Preface ...

We take an opportunity to present this Text Book on **"Operating Systems-II"** to the students of Third Year B.Sc. (Computer Science) Semester-VI as per the New Syllabus, June 2021.

The book has its own unique features. It brings out the subject in a very simple and lucid manner for easy and comprehensive understanding of the basic concepts. The book covers theory of Process Deadlocks, File System Management, Disk Scheduling, Introduction to Distributed Operating Systems and Architecture and Mobile Operating Systems.

A special word of thank to Shri. Dineshbhai Furia, and Mr. Jignesh Furia for showing full faith in us to write this text book. We also thank to Mr. Amar Salunkhe and Mr. Akbar Shaikh of M/s Nirali Prakashan for their excellent co-operation.

We also thank Ms. Chaitali Takle, Mr. Ravindra Walodare, Mr. Sachin Shinde, Mr. Ashok Bodke, Mr. Moshin Sayyed and Mr. Nitin Thorat.

Although every care has been taken to check mistakes and misprints, any errors, omission and suggestions from teachers and students for the improvement of this text book shall be most welcome.

Authors

■■■

Syllabus ...

1. Process Deadlocks (7 Lectures)

- System Model
- Deadlock Characterization - Necessary Conditions, Resource Allocation Graph
- Deadlock Methods- Prevention and Deadlock Avoidance - Safe State, Resource Allocation Graph Algorithm, Banker's Algorithm
- Deadlock Detection
- Recovery from Deadlock - Process Termination, Resource Preemption

2. File System Management (6 Lectures)

- File Concept, File Attributes, File Operations
- Access Methods - Sequential, Direct, Other Access Methods
- Directory Overview, Single Level Directory, Two Level Directory, Tree Structure Directory, Acyclic Graph Directory, General Graph Directory
- Allocation Methods - Contiguous Allocation, Linked Allocation, Indexed Allocation
- Free Space Management - Bit Vector, Linked list, Grouping, Counting, Space Maps

3. Disk Scheduling (4 Lectures)

- Overview, Disk Structure
- Disk Scheduling, FCFS Scheduling, SSTF Scheduling, Scan Scheduling - Scan Scheduling, Look Scheduling, Disk Management

4. Introduction to Distributed Operating Systems and Architecture (11 Lectures)

- What is a Distributed System, Design Goals
- Types of Distributed Systems
- Architectural Styles - Layered Architectures, Object-based Architectures, Resource-Centered Architectures
- System Architecture - Centralized Organization, Decentralized Organizations, Peer-to- Peer Systems, Hybrid Architectures.
- Example Architectures - Network File System(NFS), Web-based Distributed Systems

5. Mobile Operating Systems (7 Lectures)

- Introduction
- Features
- Special Constraints and Requirements of Mobile Operating System
- Special Service Requirements
- ARM and Intel architectures - Power Management
- Mobile OS Architectures - Underlying OS, Kernel Structure & Native Level Programming, Runtime Issues, Approaches to Power Management
- Commercial Mobile Operating Systems - Windows Mobile, iPhone OS (iOS), Android
- A Comparative Study of Mobile Operating Systems (Palm OS, Android, Symbian OS, Blackberry OS, Apple iOS)

■■■

Contents ...

■■■

CHAPTER 1

Process Deadlocks

Objectives...

- To understand Concept of Deadlock and Conditions for Deadlock
- To learn Methods for Handling Deadlocks like Deadlock Prevention, Deadlock Avoidance and Deadlock Detection
- To know Recovery from Deadlock

1.0 INTRODUCTION

- A process is a program in execution. A process may need multiple resources types, such as processors, memory space, disks and I/O devices, to accomplish its task.
- These resources are allocated either when the program is created or when it is executing.
- However, to use any resource type in system, it must follow some steps which are as follows:
  1. Request for the required resource.
  2. Use the allocated resource.
  3. Release the resource after completing the task.
- If the requested resource is not available, the requesting process enters a waiting state until it acquires the resource.
- For example, consider a computer system with a printer and a disk drive and two processes namely, Process P_1 and Process P_2 are executing simultaneously on this system. During execution, the Process P_1 requests for the printer and Process P_2 requests for the disk drive and both the requests are granted. Further, the Process P_2 requests for the printer held by Process P_1 and Process P_1 requests for the disk drive held by the Process P_2. Here, both processes will enter a waiting state. Since, each process is waiting for the release of resource held by other; they will remain in waiting state forever. This situation is called deadlock.
- In a multiprogramming system, numerous processes get competed for a finite number of resources. Any process requests resources, and as the resources aren't available at

that time, the process goes into a waiting state. At times, a waiting process is not at all able again to change its state as other waiting processes detain the resources it has requested. That condition is termed as deadlock.

- A deadlock happens in operating system when two or more processes need some resource to complete their execution that is held by the other process.
- Deadlock situation is common in multi-processing OS where multiple processes share a specific type of resource.
- Deadlock is a situation where a set of processes are blocked because each process is holding a resource and waiting for another resource acquired by some other process.
- In this chapter, we would discuss the technique for dealing with deadlocks. These techniques are deadlock prevention, deadlock avoidance and deadlock detection and recovery.

1.1 SYSTEM MODEL [April 17]

- We are going to study deadlock using a formal model to represent the resource allocation status of the system's components/elements. The system model will represent which resources are allocated to each process.
- Any system consists of a finite number of resources which are distributed among the processes. For example, if a system has five printers, then the resource is a printer and it has five instances.
- If a Process P requires an instance of a resource type printer, then it is allocated to Process P (if free) to satisfy the request.
- If the instances of the printer are not free; means all the printers are allocated to other processes, then Process P has to wait until any other process releases it.
- So a process must request a resource before using it, and must always release the resource after using it. A process may request any number of resources to carry out its task.
- A process cannot request the number of resources greater than the number of resources present in the system.
- For example, if the system has three printers and the process requests four printers, then the request is not granted.
- Under the normal mode of operation, any resource is utilized by a process in following manner:
  - **Request:** A process request for a resource. If it is available, it is immediately granted; otherwise the process must wait for it.
  - **Use:** After a request is granted, the process uses the resource.
  - **Release:** The Process releases the resource after it is used.

- Fig. 1.1 shows the above sequence.

1. Process P_1 is requesting resource R_1.

2. If Resource R_1 is free, then request made by P_1 is granted. Process P_1 uses resource R_1.

3. Process P_1 releases resource R_1 and may be used by P_2.

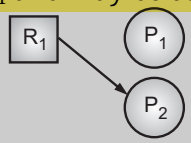


Fig. 1.1: System Model

1.1.1 Definition and Concept of Deadlock [April 17]

- Deadlocks are a set of blocked processes each holding a resource and waiting to acquire a resource held by another process.
- Deadlock occurs when every process in a set of processes are in a simultaneous wait state and each of them is waiting for the release of a resource held exclusively by one of the waiting processes in the set.

Definition of Deadlock: **[April 16, Oct. 16]**

- A set of processes is deadlocked, if each of them waits for an event that can be caused only by a process (or processes) in the set. Thus, each process waits for an event that cannot occur.
- Events related to resource allocation are:
  1. **Request:** A process requests a resource through a system call.
  2. **Allocation:** The process becomes the holder of the resource allocated to it.
  3. **Release:** A process releases a resource through a system call.
- For example, there are two tape drives, in some systems. There are two processes, each holding one more tape.
- Now each of the processes needs one more tape drive. Now both the processes are waiting for each other to release a tape drive. This is a deadlock situation.

Concept of Deadlock:

- Operating systems handle only deadlocks caused by sharing of resources in the system. Such deadlocks arise when some conditions concerning resource requests and resource allocations hold simultaneously.
- Deadlock handling approaches use these conditions as the basis for their actions. Deadlock detection detects a deadlock by checking whether all conditions necessary for a deadlock hold simultaneously.

- The deadlock prevention and deadlock avoidance approaches ensure that deadlocks cannot occur, by not allowing the conditions for deadlocks to hold simultaneously.
- A simple example of a resource deadlock is shown in Fig. 1.2. This resource allocation graph shows two processes as rectangles and two resources as ellipses.
- An arrow from a resource to a process indicates that the resource belongs to or has been allocated to the process.
- An arrow from a process to resources indicates that the process is requesting, but has not yet been allocated, the resource.
- Fig. 1.2 shows a deadlocked system with Process P_1 holds Resource R_1 and needs Resource R_2.
- The system is deadlocked because each process holds a resource being requested by the other process and neither process is willing to release the resource it holds.
- Process P_2 holds Resource R_2 and needs Resource R_1 to continue. Each process is waiting for the other to free a resource that it will not free until the other process releases its resource that it will not do until the first process frees its resource.

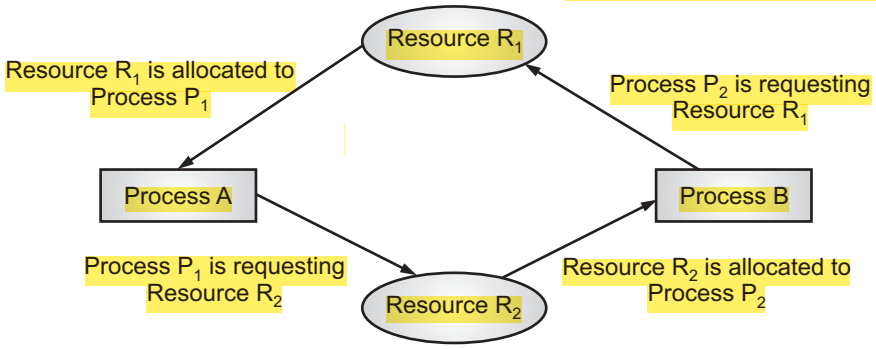


Fig. 1.2: A Simple Deadlock

1.2 DEADLOCK CHARACTERIZATION

- Before discussing the methods to handle a deadlock, we will discuss the conditions that cause a deadlock.

1.2.1 Necessary Conditions [Oct. 18]

- Edward G. Coffman (1971) identified four conditions that must hold simultaneously for there to be a deadlock. These four conditions are explained below:

1. **Mutual Exclusion Condition**: The resources involved are non-shareable. Only one process at a time can use a resource.

   Explanation: At least one resource (thread) must be held in a non-shareable mode, that is, only one process at a time claims exclusive control of the resource. If another process requests that resource, the requesting process must be delayed until the resource has been released.

2. **Hold and Wait Condition:** A process holding at least one resource is waiting to acquire additional resources held by other processes.
   Explanation: There must be a process that is holding a resource already allocated to it; while waiting for additional resources that are currently being held by other processes.
3. **No-Preemption Condition:** Resources already allocated to a process cannot be preempted.
   Explanation: Resources cannot be removed from the processes until its completion or released voluntarily by the process holding it.
4. **Circular Wait Condition:** The processes in the system form a circular list or chain; where each process in the list is waiting for a resource held by the next process in the list.
   Explanation: A set $\{P_0, P_1, ..., P_n\}$ of waiting processes must exist such that P_0 is waiting for resource held by P_1, P_1 is waiting for a resource held by P_2,.....P_{n-1} is waiting for a resource held by P_n and P_n is waiting for a resource held by P_0.

- It is necessary to understand that all four conditions must be satisfied simultaneously for a deadlock to occur. If any one of them does not occur, a deadlock can be avoided.
- Fig. 1.3 shows a kind of deadlock that occasionally occurs/develops in a city in the traffic system.
- A number of automobiles are attempting to drive through a busy section of the city, but the traffic becomes completely jumbled.
- Traffic comes to a halt, and it is necessary for the police to remove the jam by slowly and carefully backing cars out of the congested area.
- Eventually the traffic begins to flow normally, without any annoyance, and effort now, but of course with loss of considerable time.

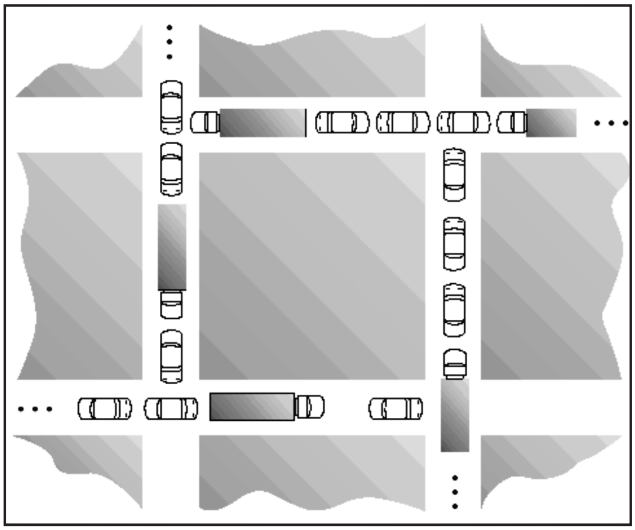


Fig. 1.3: Traffic Deadlock in City

- Consider each section of the street as a resource.
  1. **Mutual exclusion** condition applies, since only one vehicle can be on a section of the street at a time.
  2. **Hold-and-wait** condition applies, since each vehicle is occupying a section of the street, and waiting to move on to the next section of the street.
  3. **No-preemption** condition applies, since a section of the street that is occupied by a vehicle cannot be taken away from it.
  4. **Circular wait** condition applies, since each vehicle is waiting on the next vehicle to move. That is, each vehicle in the traffic is waiting for a section of street held by the next vehicle in the traffic.
- The simple rule to avoid traffic deadlock is that a vehicle should only enter an intersection if it is assured that it will not have to stop inside the intersection.
- It is not possible to have a deadlock involving only one single process. The deadlock involves a circular "hold-and-wait" condition between two or more processes, so "one" process cannot hold a resource, yet be waiting for another resource that it is holding.
- In addition, deadlock is not possible between two threads in a process, because it is the process that holds resources, not the thread, that is each thread has access to the resources held by the process.

1.2.2 Resource Allocation Graph [Oct. 17]

- Deadlocks can be illustrated more precisely in terms of a resource allocation graph. Resource allocation graph is nothing but a directed graph, which is used to describe resources allocated to a particular process in a system.
- The graph is made up of:
  - A set of processes P = {P_1, P_2, ..., P_n}. This set contains all the processes in the system.
  - A set of resources R = {R_1, R_2, ..., R_n}. This contains all the resources present in the system.
  - A set of directed edges.
  - A directed edge from a process P_i to a resource R_j. ($P_i \rightarrow R_j$) is called request edge.
  - A directed edge from a resource R_j to a process P_i ($R_j \rightarrow P_i$) is called an allocation edge or an assignment edge.

Conventions:

1. In resource allocation the graph process is represented by a circle.
2. Resource is represented by a rectangle in the resource allocation graph.
3. Number of resources of similar type (instances) is represented by dots.
4. Process may be holding a resource or may be requesting for a resource.

5. When process P_i requests an instance of resource type R_j, a request edge is inserted in the resource-allocation graph.
6. When this request can be fulfilled, the request edge is instantly transformed to an assignment edge.
7. When the process no longer needs access to the resource, it releases the resource; as result, the assignment edge is removed.
8. If the graph contains no cycle, then no process in the system is deadlocked.
9. If graph contains cycles, then deadlock may exist.

- The resource allocation graph as shown in Fig. 1.4 depicts the situation.

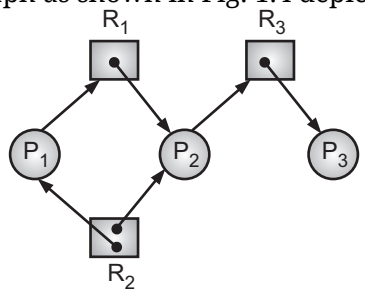


Fig. 1.4: Resource Allocation Graph

- $P = (P_1, P_2, P_3)$
- $R = (R_1, R_2, R_3)$
- E = Request Edge = $\{P_1 \rightarrow R_1, P_2 \rightarrow R_3\}$
  Assignment Edge = $\{R_1 \rightarrow P_2, R_3 \rightarrow P_3, R_2 \rightarrow P_1, R_2 \rightarrow P_2\}$

Process States:

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

- As the graph contains no cycle, the system is deadlock free.
- Consider Fig. 1.5 in which Process P_3 requests an instance of resource type R_2. At this point, two minimal cycles exist in the system:
  $P_1 \rightarrow R_1 \rightarrow P_2 \rightarrow R_3 \rightarrow P_3 \rightarrow R_2 \rightarrow P_1$
  $P_2 \rightarrow R_3 \rightarrow P_3 \rightarrow R_2 \rightarrow P_2$

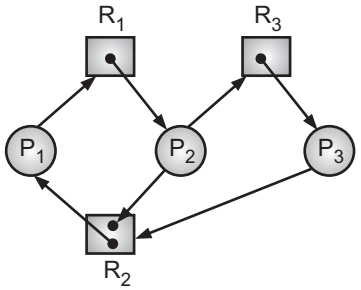


Fig. 1.5: Resource Allocation Graph with Deadlock

- Processes P_1, P_2 and P_3 are deadlocked. Process P_2 is waiting for the resource R_3, which is held by process P_3.
- Process P_3 is waiting for either process P_1 or P_2 to release resource R_2. Also process P_1 is waiting for process P_2 to release resource R_1. So the system is in deadlock.
- Consider resource allocation graph shown in Fig. 1.6 which contains cycle, $P_1 \rightarrow R_1 \rightarrow P_2 \rightarrow R_2 \rightarrow P_1$. However there is no deadlock.
- Process P_3 may release the resource R_2. Then that resource R_2 can be allocated to P_2, breaking the cycle.

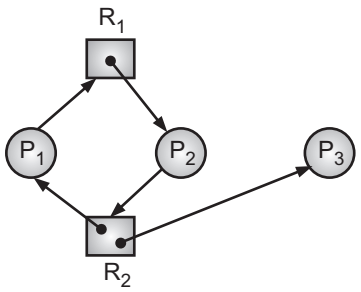


Fig. 1.6: Resource Allocation Graph with Cycle but no Deadlock

- If graph contains cycle, then check the following:
  - If there is one instance per resource type, then there is a deadlock.
  - If resource type has multiple instances, then there may or may not be deadlock.

1.3 DEADLOCK HANDLING METHODS [Oct. 16, April 19]

- The common methods for dealing with the deadlock situation are as follows:
  1. Prevent the deadlock from occurring.
  2. Adopt methods for avoiding the deadlock.
  3. Allow the deadlock to occur, detect it and recover from it.
  4. Ignore the deadlock.
- The deadlock handling techniques are deadlock prevention, deadlock avoidance, ignoring deadlock, deadlock detection and recovery.
- Deadlock prevention (or deadlock avoidance) techniques can be used to ensure that deadlocks never occur in a system.
- If any of these two techniques is not used, a deadlock may occur. In this case, an algorithm can be provided for detecting the deadlock and then using the algorithm to recover the system from the deadlock.
- Deadlock prevention algorithm disallows one of four necessary conditions for deadlock.
- Deadlock avoidance algorithms do not grant resource requests if resource-allocation has potential have lead to deadlock.

- The other method must be provided to either prevent the deadlock from occurring or detecting the deadlock and takes an appropriate action if a deadlock occurs.
- However, if in a system, deadlock occurs less frequently (say, once in two years) then it is better to ignore the deadlocks instead of adopting expensive techniques for deadlock prevention, deadlock avoidance or deadlock detection and recovery.
- Deadlock detection and recovery algorithm always grants resource requests when possible. It periodically checks for a deadlock. If the deadlock exists, recover from it.

1.3.1 Deadlock Prevention [April 17, 18, Oct. 17]

- Deadlock prevention is very important to prevent a deadlock before it can occur. So, the system checks each process before it is executed to make sure it does not lead to deadlock. If there is even a slight chance that a process may lead to deadlock in the future, it is never allowed to execute.
- A deadlock can be prevented if either of four conditions (mutual exclusion, hold-and-wait, no-preemption and circular wait) is prevented from taking place.
- By ensuring that at least one of these conditions cannot hold, we can prevent the occurrence of a deadlock. In simple words, a deadlock can be prevented by eliminating any one of the four necessary conditions of the deadlock.
- Preventing deadlock using this method results in the inefficient use of resources. Let us consider each of them separately:

1. **Eliminating Mutual Exclusion Condition:**
   - The mutual exclusion condition must hold for non-sharable resources. That is, several processes cannot simultaneously share a single resource.
   - This condition is difficult to eliminate because some resources, such as the tape drive and printer, are inherently non-shareable.
   - The printer can work for only one process at a time. It cannot print data being sent as output from more than one process simultaneously. Hence, the condition of mutual exclusion cannot be eliminated in case of all the resources.
2. **Eliminating Hold-and-Wait Condition:**
   - In order to prevent this condition to hold, we must guarantee that the process holding some resource does not request for more resources.
   - The hold-and-wait condition can be eliminated by not allowing any process to request for a resource until it releases the resources held by it, which is impractical as processes may require the resources simultaneously.
   - Another way to prevent this condition is by allocating all the required resources to the process before starting the execution of that process.
   - The disadvantage associated with it is that a process may not know in advance about the resources that will be required during its execution.

- Even if it knows in advance, it may unnecessarily hold the resources which may be required at the end of its execution. Thus, the system resources are not utilized optimally.

3. Eliminating No Preemption Condition:

- The elimination of the no-preemption condition means a process can release the resource held by it.
- If a process requests for a resource held by some other process then instead of making it wait, all the resources currently held by this process can be preempted.
- The process will be restarted only when it is allocated the requested as well as the preempted resources.
- Note that only those resources can be preempted whose current working state can be saved and can be later restored.
- For example, the system resources like printer and disk drives cannot be preempted.

4. Eliminating Circular Wait Condition:

- This condition can be eliminated by assigning a priority number to each available resource and a process can request system resources only in increasing order.
- Whenever a process requests for a system resource, the priority number of the required resource is compared with the priority numbers of the resources already held by it.
- If the priority number of a requested resource is greater than that of all the currently held system resources, the request is granted.
- If the priority number of a requested resource is less than that of the currently held resources, all the resources with greater priority number must be released first, before acquiring the new system resource.

1.3.2 Deadlock Avoidance

- If detail information about the processes and resources is available, then it is possible to avoid deadlock.
- For example, which process will require which resources, possibly in what sequence etc. This information may help to decide the sequence in which the processes can be executed to avoid deadlock.
- Each request can be analyzed on the basis of the number of resources currently available, currently allocated and future requests which may come from other processes. From this information, the system can decide whether or not a process should wait.
- The deadlock avoidance algorithm dynamically examines the resource- allocation state from the available information to ensure that there is no circular wait.

- The resource-allocation state is defined by the number of available and allocated resources and the maximum demands of the processes.
- In deadlock avoidance, the request for any resource will be granted if the resulting state of the system doesn't cause deadlock in the system. The state of the system will continuously be checked for safe and unsafe states.
- Deadlock avoidance requires the information about the future process resource requests. A safe state is not a deadlock state and a deadlock state is an unsafe state.
- Not all unsafe states are deadlocks; however an unsafe state may lead to a deadlock. As long as the state is safe, the operating system can avoid unsafe (and deadlock) states (See Fig. 1.7).

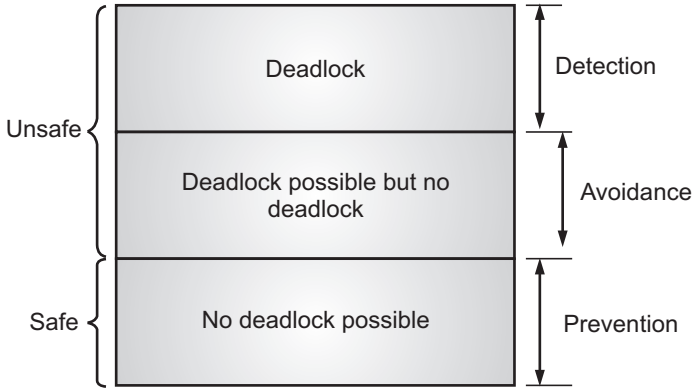


Fig. 1.7: Scope of Policies in Deadlock

1.3.2.1 Safe State [April 19]

- The state of resource allocation can be either safe or unsafe. A state is safe if the system can allocate resources to each process in some order and still avoid a deadlock.
- In other words, a state is said to be safe if allocation of resources to processes does not lead to the deadlock.
- A system is in a safe state; if there exists a safe sequence. A safe sequence is a sequence of process execution such that each and every process executes till its completion.
- Safe sequence is a sequence of processes $<P_1, P_2, ..., P_n>$ is a safe sequence for current allocation state if, for each process Pi, the resource requests that P_i can still make can be satisfied by the currently available resources plus the resources held by all P_j have finished with j < i.
  - If resources that P_i needs are not available, then P_i can wait until all P_j have finished.
  - When P_j terminates, P_i will obtain its resources and complete the task.
  - When P_i terminates, P_{i+1} obtains its resources and so on.
  - If no such sequence exists, then the system is in unsafe state.
  - A system without safe sequence is unsafe.

- A safe state is not a deadlocked state. Conversely, a deadlocked state is an unsafe state. Thus, unsafe systems may lead to deadlock as shown in Fig. 1.8.

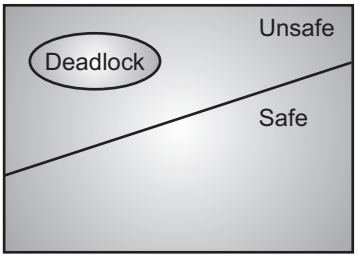


Fig. 1.8: Safe, Unsafe and Deadlock State Spaces

1.3.3 Resource Allocation Graph Algorithm [April 18]

- Resource allocation graph algorithm uses a variant of resource allocation graph for a single instance of resource type. 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.
- The resource allocation graph consists of two types of edges namely, request edge and assignment edge.
- In the resource allocation graph the processes are depicted as circles and resources as squares.
- In a resource allocation graph, a directed arc from a process to a resource (known as request edge) indicates that the process has requested for the resource and is waiting for it to be allocated. Whereas, a directed arc from a resource to a process (known as assignment edge) indicates that the resource has been allocated to the process.
- In this algorithm, apart from all sets and edges from the resource allocation graph, one more edge is used. This edge is called claim edge.
- A claim edge (P_1, R_j) indicates that process P_i may request resource R_j, sometime in future.
- A claim edge is represented by a dashed or dotted line. All the claim edges must appear in the graph before P_i starts execution.
- When process P_i requests for a resource R_j, then the claim edge $<P_i, R_j>$ is converted to request edge $<P_i, R_j>$ (dashed line converted to regular line).
- A resource is granted only when conversion of request edge $<P_i, R_j>$ to assignment edge $<R_j, P_i>$ does not form any cycle in the graph.
- If graph contains a cycle, then the system is in an unsafe state.

- Consider the following example, in Fig. 1.9, Processes P_1 and P_2 claim for resources before execution.
  - In Fig. 1.9 (a), Process P_1 requests Resource R_A . So claim edge $<P_1, R_A>$ is converted into request edge $<P_1, R_A>$.
  - In Fig. 1.9 (b), a request made by Process P_1 for Resource R_A is granted. So request edge $<P_1, R_A>$ is converted into assignment edge $<R_A, P_1>$.
  - In Fig. 1.9 (c), Process P_2 requests Resource R_B. So claim edge $<P_2, R_B>$ is converted into request edge $<P_2, R_B>$.
  - In Fig. 1.9 (d), if a request made by Process P_2 for Resource R_B is granted, it forms a cycle in the graph. So allocation is denied, even if R_B is available.

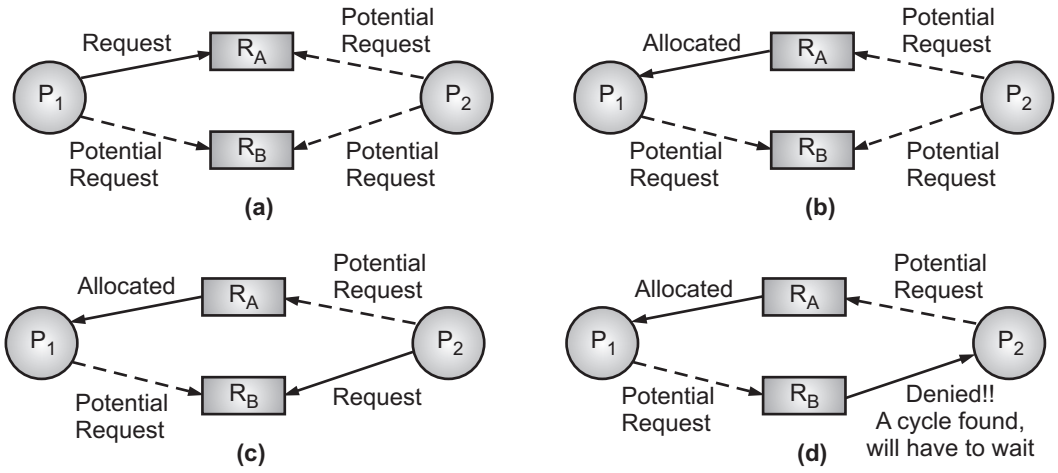


Fig. 1.9: Resource Allocation Graph for Deadlock Avoidance

1.3.4 Banker's Algorithm [April 18, 19]

- The deadlock avoidance algorithm used for several instances of the resources is called Banker's algorithm developed by Edsger Dijkstra (1965).
- It has been named as Banker's algorithm, because this algorithm could be used in a banking system to ensure that the bank never allocates its available cash in such a way that it can no longer satisfy the requirement of all customers (means available cash is always > customer requirement).
- Each process must a priori claim for maximum use. When a process requests a resource, it may have to wait. When a process gets all its resources it must return them in a finite amount of time.
- In Banker's algorithm, any process entering the system must inform the maximum number of resources (less than (>) the total number of available resources) required during its execution.

- If allocating this much number of resources to the process leaves the system in a safe state the resources are allocated.
- If allocation of resources leaves the system in an unsafe state then the resources are not allocated and the process is forced to wait for some other processes to release enough resources.
- To implement the Banker's algorithm certain data structures are required, which help in determining whether the system is in safe state or not.
- Several data structures are used to implement Banker's algorithm. Let n be the number of processes and m be the number of resource types.
  1. **Available:** A vector of length m indicating the number of available resources of each type. If Available [j] = k means there are k instances of resource type R_j available.
  2. **Max:** A n × m matrix defining the maximum demand of each process. If Max [i, j] = k, then process P_i may request at most k instances of resource type R_j.
  3. **Allocation:** A n × m matrix defining number of resources of each type currently allocated to each process. If Allocation [i, j] = k then process P_i is currently allocated k instances of resource type R_j.
  4. **Need:** A n × m matrix indicating the remaining resource need of each process. if Need [i, j] = k, then process P_i may need k more instance resource type R_j, in order to complete its task.
- The Banker's algorithm can be divided into two parts namely, Safety algorithm (if a system is in a safe state or not) and resource request algorithm (make an assumption of allocation and see if the system will be in a safe state).
- If the new state is unsafe, the resources are not allocated and the data structures are restored to their previous state; in this case the processes must wait for the resource.

1.3.4.1 Safety Algorithm

- A safety algorithm is an algorithm used to find whether or not a system is in its safe state.
- Let Work and Finish be vectors of length m and n, respectively.
  1. Initialize Work = Available
     Finish [i] = false for i = 0, 1, ..., n – 1.
  2. Find i such that both:
     (a) Finish [i] = = false
     (b) $Need_i \leq Work$
     If no such i exists, go to step 4.
  3. Work = Work + $Allocation_i$, Finish[i] = true, go to step 2.
  4. If Finish [i] == true for all i, then the system is in a safe state.
- This algorithm requires an order of $m \times n^2$ operation to check whether the system is in safe state.

1.3.4.2 Resource – Request Algorithm

- Once it is confirmed that the system is in safe state, a resource-request algorithm is used for determining whether the request by a process can be satisfied or not.
- The resource-request algorithm determines if requests can be safely granted.
- Let $Request_i$ be a request vector for process P_i. If $Request_i$ [j] = k then process P_i wants k instances of resource type R_j.
  1. If $Request_i \leq Need_i$ go to step 2. Otherwise, raise the error condition, since Process has exceeded its maximum claim.
  2. If $Request_i \leq Available$, go to step 3. Otherwise P_i must wait, since resources are not available.
  3. Pretend to allocate requested resources to P_i by modifying the state as follows:

     $Available = Available – Request_i$;

     $Allocation_i = Allocation_i + Request_i$;

     $Need_i = Need_i – Request_i$

     If safe state $\Rightarrow$ the resources are allocated to P_i.

     If unsafe state $\Rightarrow$ P_i must wait, and the old resource-allocation state is restored.

Example: Consider the following snapshot of system, A, B, C and D are the resource type.

	Allocation			
	A	**B**	**C**	**D**
P_0	0	0	1	2
P_1	1	0	0	0
P_2	1	3	5	4
P_3	0	6	3	2
P_4	0	0	1	4

	MAX			
	A	**B**	**C**	**D**
P_0	0	0	1	2
P_1	1	7	5	0
P_2	2	3	5	6
P_3	0	6	5	2
P_4	0	6	5	6

Available			
A	**B**	**C**	**D**
1	5	2	0

- Answer the following questions using Banker's Algorithm:
  1. What is the contents of the need array?
  2. Is the system in safe state? If yes give the safe sequence.
  3. If a request from process P1 arrives for (0, 4, 2, 0) can it be granted immediately?

  Solution:
  1. The content of need array is as follows:

     Need[i][j] = Max[i][j] – Allocation[i][j]

	Max			
	A	**B**	**C**	**D**
$\mathbf{P_0}$	0	0	1	2
$\mathbf{P_1}$	1	7	5	0
$\mathbf{P_2}$	2	3	5	6
$\mathbf{P_3}$	0	6	5	2
$\mathbf{P_4}$	0	6	5	6

_

	Allocation			
	A	**B**	**C**	**D**
$\mathbf{P_0}$	0	0	1	2
$\mathbf{P_1}$	1	0	0	0
$\mathbf{P_2}$	1	3	5	4
$\mathbf{P_3}$	0	6	3	2
$\mathbf{P_4}$	0	0	1	4

=

	Need			
	A	**B**	**C**	**D**
$\mathbf{P_0}$	0	0	0	0
$\mathbf{P_1}$	0	7	5	0
$\mathbf{P_2}$	1	0	0	2
$\mathbf{P_3}$	0	0	2	0
$\mathbf{P_4}$	0	6	4	2

2. To check whether system is in safe state, Banker's safety algorithm is used.

Initialize Finish = { False, False, False, False, False,}
Work = Available = {1,5,2,0}

Let i=0
Finish[0] == False
Check Need(P_0) <= Work
{0,0,0,0} <= {1,5,2,0}=>true
Therefore, Process P_0 can be granted required resources.
So Work = Work + Allocation[P_0] (After P_0 finishes its task, it will release resources)
Work = {1,5,2,0} + {0,0,1,2}
= {1,5,3,2}
Finish = { True, False, False, False, False}
Safe Sequence = {P_0}

Let i=1
Finish[1] == False
Check Need(P_1) <= Work
{0,7,5,0} <= {1,5,3,2} =>False
Therefore Process P_1 can not be given resources. P_1 must wait.

Let i=2 Finish[2] == False
Check Need(P_2) <= Work
{1,0,0,2} <= {1,5,3,2} =>true
Therefore, Process P_2 can be granted required resources.
So Work = Work + Allocation[P_2] (After P_2 finishes its task, it will release resources)
Work = {1,5,3,2} + {1,3,5,4}
= {2,8,8,6}
Finish = { True, False, True, False, False}
Safe Sequence ={P_0,P_2}

Let i=3 Finish[3] == False
Check Need(P_3) <= Work
{0,0,2,0} <= {2,8,8,6}=>true
Therefore, Process P_3 can be granted required resources.
So Work = Work + Allocation[P3] (After P_3 finishes its task, it will release resources)
Work = {2,8,8,6} + {0,6,3,2}
= {2,14,11,8}
Finish = { True, False, True, True, False}
Safe Sequence ={P_0,P_2,P_3}

Let i=4, Finish[4] == False
Check Need(P_4) <= Work
{0,6,4,2} <= {2,14,11,8}=>true
Therefore, Process P_4 can be granted required resources.
So Work = Work + Allocation[P_4] (After P_4 finishes its task, it will release resources)
Work = {2,14,11,8} + {0,0,1,4}
= {2,14,12,12}
Finish = { True, False, True, True, True}
Safe Sequence ={P_0,P_2,P_3,P_4}

Again Check Finish[1] == False
Check Need(P_1) <= Work
{0,7,5,0} <= {2,14,12,12}=>true
Therefore, Process P_1 can be granted required resources.
So Work = Work + Allocation[P_1] (After P_1 finishes its task, it will release resources)
Work = {2,14,12,12} + {1,0,0,0}
= {3,14,12,12}
Finish = { True, True, True, True, True}
Safe Sequence = {P_0,P_2,P_3,P_4,P_1}
Yes, system is in safe state.

3. If request from process P_1 arrives for {0,4,2,0) can it granted immediately?

Resource – request algorithm is used to check request can be granted immediately.

(i) Request(P_1) < = Need(P_1) => {0,4,2,0} <= {0,7,5,0} => True

(ii) Request(P_1) < = Available(P_1) => {0,4,2,0} <= {1,5,2,0} => True

(iii) Then system pretends to have allocated the requested resources to process P_1 by modifying the state as follows:

Available = Available – Request(P_1);

$Allocation_i$ = $Allocation_i$ + Request(P_1);

Need(P_1) = Need(P_1) – Request(P_1)

	Allocation			
	A	**B**	**C**	**D**
$\mathbf{P_0}$	0	0	1	2
$\mathbf{P_1}$	1	4	2	0
$\mathbf{P_2}$	1	3	5	4
$\mathbf{P_3}$	0	6	3	2
$\mathbf{P_4}$	0	0	1	4

	Need			
	A	**B**	**C**	**D**
$\mathbf{P_0}$	0	0	0	0
$\mathbf{P_1}$	0	3	3	0
$\mathbf{P_2}$	1	0	0	2
$\mathbf{P_3}$	0	0	2	0
$\mathbf{P_4}$	0	6	4	2

Available			
A	**B**	**C**	**D**
1	1	0	0

- Now again check whether the system is in safe state, by using Banker's safety algorithm and find a safe sequence.

Initialize Finish = { False, False, False, False, False,}
Work = Available = {1,1,0,0}
Let i=0
Finish[0] == False
Check Need(P_0) <= Work
{0,0,0,0} <= {1,1,0,0} =>true
Therefore, Process P_0 can be granted required resources.
So Work = Work + Allocation[P_0] (After P_0 finishes its task, it will release resources)
Work = {1,1,0,0} + {0,0,1,2}
= {1,1,1,2}
Finish = { True, False, False, False, False}
Safe Sequence = {P_0}

Let i=1
Finish[1] == False
Check Need(P_1) <= Work
{0,3,3,0} <= {1,1,1,2} => False
Therefore, Process P_1 can not be given resources. P_1 must wait.

Let i=2 Finish[2] == False
Check Need(P_2) <= Work
{1,0,0,2} <= {1,1,1,2} => true
Therefore, Process P_2 can be granted required resources.
So Work = Work + Allocation[P_2] (After P_2 finishes its task, it will release resources)
Work = {1,1,1,2} + {1,3,5,4}
= {2,4,6,6}
Finish = { True, False, True, False, False}
Safe Sequence = {P_0,P_2}

Let i=3 Finish[3] == False
Check Need(P_3) <= Work
{0,0,2,0} <= {2,4,6,6} =>true
Therefore, Process P_3 can be granted required resources.
So Work = Work + Allocation[P_3] (After P_3 finishes its task, it will release resources)
Work = {2,4,6,6} + {0,6,3,2}
= {2,10,9,8}
Finish = { True, False, True, True, False}
Safe Sequence = {P_0,P_2,P_3}

Let i=4, Finish[4] == False
Check Need(P_4) <= Work
{0,6,4,2} <= {2,10,9,8}=>true
Therefore, Process P_4 can be granted required resources.
So Work = Work + Allocation[P_4] (After P_4 finishes its task, it will release resources)
Work = {2,10,9,8} + {0,0,1,4}
= {2,10,10,12}
Finish = { True, False, True, True, True}
Safe Sequence ={P_0,P_2,P_3,P_4}

Again Check Finish[1] == False
Check Need(P_1) <= Work
{0,3,3,0} <= {2,10,10,12} => true
Therefore, Process P1 can be granted required resources.
So Work = Work + Allocation[P_1] (After P_1 finishes its task, it will release resources)
Work = {2,10,10,12} + {1,4,2,0}
= {3,14,12,12}
Finish = { True, True, True, True, True}
Safe Sequence = {P_0,P_2,P_3,P_4,P_1}
Yes, the system is in the safe state. So, requests of Process P_1 can be granted immediately.

1.4 DEADLOCK DETECTION

- The systems that do not implement algorithms for deadlock prevention or avoidance must implement an algorithm for deadlock detection and recovery.
- In other words, there is a possibility of deadlock if neither the deadlock prevention nor deadlock avoidance method is applied in a system.
- In such a situation, an algorithm must be provided for detecting the occurrence of deadlock in a system.
- Once the deadlock is detected, a methodology must be provided for the recovery of the system from the deadlock.

- The goal of deadlock detection is to determine if a deadlock has occurred, and to determine exactly those processes and resources involved in the deadlock. Once this is determined, the deadlock can be removed from the system.
- The system must maintain information about:
  1. Process status (resource allocated and requested by a process).
  2. An algorithm to detect the deadlock.
- The deadlock prevention and avoidance methods are very conservative, as it imposes restrictions on the processes by limiting access to the resources.
- If a system does not employ any protocol that ensures that no deadlock will ever occur, then a detection and recovery scheme must be used by the system.
- It requires an algorithm to determine that a deadlock exists and identify the resources and processes involved in the deadlock. These deadlock. Recovery from a deadlock may be quite difficult and expensive.
- There are two algorithms i.e. Single instance resource type and Several instances of a resource type.

1.4.1 Single Instance Resource Type [April 17]

- If all resources have a single instance in a system, then the deadlock detection algorithm uses a variant of resource allocation graph, known as wait-for-graph.
- The wait-for-graph shows the dependency of a process on another process for the resource allocation. This graph contains only processes, resources are removed.
- An edge from process P_i to process P_j exists in wait-for-graph if and only if the corresponding resource allocation graph contains two edges $P_i \rightarrow R_q$ and $R_q \rightarrow P_j$ for some resource type R_q. If the wait-for-graph contains cycle, then a deadlock exists in the system.
- To detect a deadlock, the system needs to maintain a wait-for-graph and periodically invoke an algorithm that searches for cycles in the graph.
- The algorithm requires an order of n^2 operations where n is the number of vertices.
- In the Fig. 1.10, wait-for-graph is shown corresponding to resource allocation graph. In wait-for-graph:

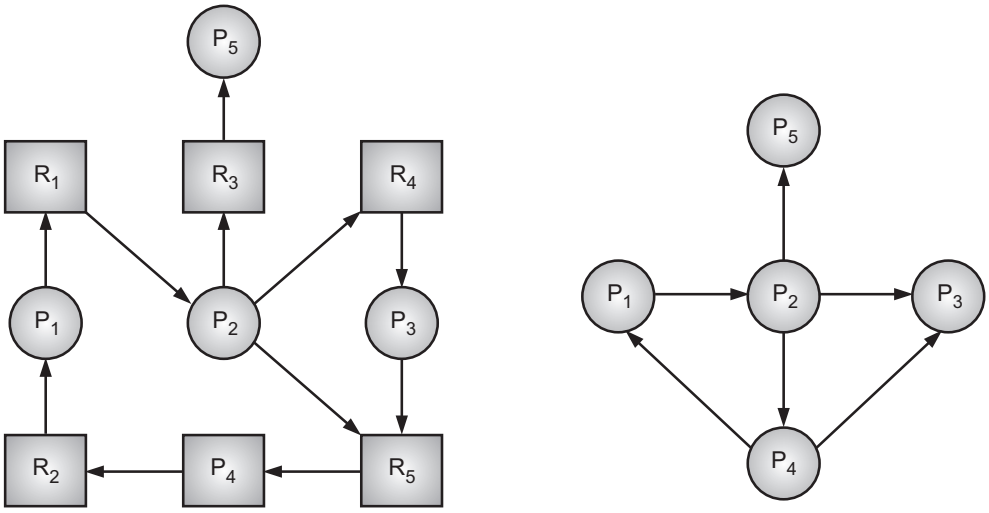
  Edge $P_1 \rightarrow P_2$ exists since there is corresponding edge from $P_1 \rightarrow R_1$ and $R_1 \rightarrow P_2$

  Edge $P_2 \rightarrow P_3$ exists since there is corresponding edge from $P_2 \rightarrow R_4$ and $R_4 \rightarrow P_3$

  Edge $P_3 \rightarrow P_4$ exists since there is corresponding edge from $P_3 \rightarrow R_5$ and $R_5 \rightarrow P_4$

  Edge $P_4 \rightarrow P_1$ exists since there is corresponding edge from $P_4 \rightarrow R_2$ and $R_2 \rightarrow P_1$

  Edge $P_2 \rightarrow P_5$ exists since there is corresponding edge from $P_2 \rightarrow R_3$ and $R_3 \rightarrow P_5$
- As there is cycle $P_1 \rightarrow P_2 \rightarrow P_4 \rightarrow P_1$ in wait-for-graph, Process P_1, P_2 and P_4 are deadlocked.



(a) Resource Allocation Graph **(b) Wait-For-Graph**

Fig. 1.10

1.4.2 Several Instances of the Resource Type

- The wait-for-graph is not very useful if there are multiple instances for a resource. When multiple instances of a resource type exist, the wait-for graph becomes inefficient to detect the deadlock in the system.
- For such a system, another algorithm which uses certain data structures similar to the ones used in banker's algorithm is applied.
- The deadlock detection algorithm uses same data structures as Banker's algorithm:
  - Available = A vector of length m indicating number of available resources of each type.
  - Allocation = An n × m matrix indicating the number of resources of each type currently allocated to each process.
  - Request = A n × m matrix indicating the current request of each process. If Request[i][j] = k then process P_i is requesting k more instances of resource type Rj.

Algorithm:

Step 1: Let Work and Finish be vectors of length m and n, respectively. Initialize Work = Available. For i = 1, 2, ..., n, if $Allocation_i \neq 0$, then Finish[i] = false; otherwise, Finish[i] = true.

Step 2: Find an index i such that both:

(a) Finish[i] == false

(b) $Request_i \leq Work$

If no such i exists, go to step 4.

Step 3: Work = Work + $Allocation_i$, Finish[i] = true go to step 2.

Step 4: If Finish[i] == false, for some i, $1 \le i \le n$, then the system is in a deadlock state. Moreover, if Finish[i] == false, then Pi is deadlocked.

- Algorithm requires an order of $O(m \times n^2)$ operations to detect whether the system is in a deadlocked state.

Example: Consider the system with 3 resources types A, B and C with 7, 2, 6 instances respectively. Consider the following snapshot:

	Allocation		
	A	**B**	**C**
$\mathbf{P_0}$	0	1	0
$\mathbf{P_1}$	2	0	0
$\mathbf{P_2}$	3	0	3
$\mathbf{P_3}$	2	1	1
$\mathbf{P_4}$	0	0	2

	Request		
	A	**B**	**C**
$\mathbf{P_0}$	0	0	0
$\mathbf{P_1}$	2	0	2
$\mathbf{P_2}$	0	0	1
$\mathbf{P_3}$	1	0	0
$\mathbf{P_4}$	0	0	2

Total Resources		
A	**B**	**C**
7	2	6

Answer the following questions:

1. What are the contents of the Available array?
2. Is there any deadlock?

Solution: The content of Available array is as follows:

Available = Total Resources – Total Allocation

Total Allocation = {7,2,6}

	Allocation		
	A	**B**	**C**
$\mathbf{P_0}$	0	1	0
$\mathbf{P_1}$	2	0	0
$\mathbf{P_2}$	3	0	3
$\mathbf{P_3}$	2	1	1
$\mathbf{P_4}$	0	0	2
	7	2	6

Available = {7, 2, 6} – {7, 2, 6}

= {0, 0, 0}

Now execute the deadlock detection algorithm.

Initialize Finish = {False, False, False, False, False} Work = Available = {0,0,0}
Let i=0 Check Finish[0] == False Check Request(P_0) <= Work {0,0,0} <= {0,0,0}=>true So Work = Work + Allocation[P0] Work = {0,0,0} + {0,1,0} = {0,1,0} Finish = { True, False, False, False, False}
Let i=1 Finish[1] == False Check Request(P_1) <= Work {2,0,2} <= {0,1,0} =>False Finish = { True, False, False, False, False} Let i=2; Finish[2] == False Check Request(P_2) <= Work {3,0,3} <= {0,1,0} => False Finish = { True, False, False, False, False}
Let i=3; Finish[3] == False Check Request(P_3) <= Work {2,1,1} <= {0,1,0} =>False Finish = { True, False, False, False, False}
Let i=4; Finish[4] == False Check Request(P_4) <= Work {0,0,2} <= {0,1,0} =>False Finish = { True, False, False, False, False} As Finish[i] == False for process P_1, P_2, P_3 and P_4, the system is in a deadlock state. Process P_1, P_2, P_3 and P_4 are deadlocked.

1.5 RECOVERY FROM DEADLOCK [April 16, Oct. 18]

- When a deadlock detection algorithm detects a deadlock in a system, some recovery schemes/methods must be used to recover the system from deadlock.
- Deadlock recovery methods are used to clear deadlocks from a system so that it may proceed to operate free of the deadlock, and so that the deadlocked processes may complete their execution and free their resources.
- Once the system has detected a deadlock in the system, some method is needed to recover the system from the deadlock and continue with the processing.
- The two different ways in which system can be recovered are:
  1. Terminate one or more process to break the circular wait condition and
  2. Preempt the resources from the processes involved in the deadlock.
- The deadlock recovery methods are shown in Fig. 1.11.

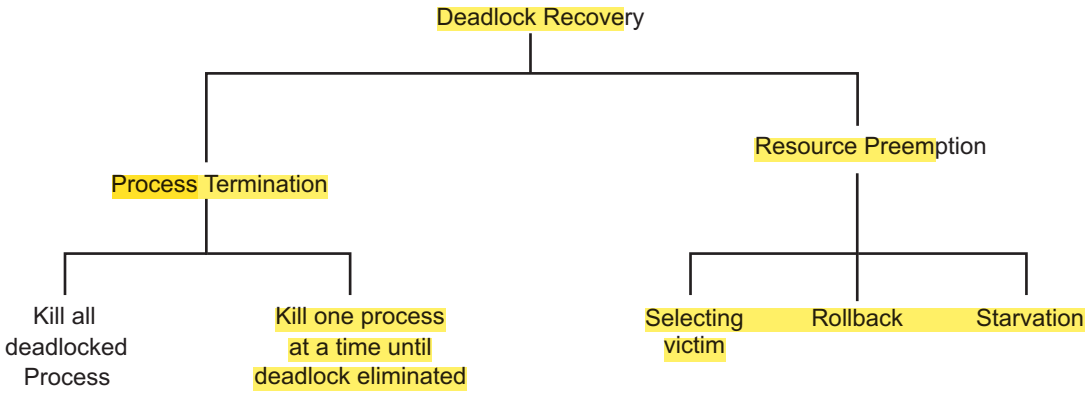


Fig. 1.11: Deadlock Recovery Scheme

- The Fig. 1.11 shows the different ways for breaking a deadlock.
  - The first option in deadlock recovery is to kill one or more processes in order to break the circular wait.
  - The next option is to preempt some resources from one or more of the deadlocked processes.

1.5.1 Process Termination

- When the system detects a deadlock, it needs to recover from it. There are two methods that can be used for terminating the processes to recover from the deadlock.
  1. **Kill all Deadlocked Processes:** This scheme/method is a very expensive way of breaking a deadlock. This method will definitely ensure the recovery of a system from the deadlock.
  2. **Kill One Process at a Time Until the Deadlock is Eliminated:** This scheme/ method involve a lot of overhead, since after killing each process; deadlock detection algorithm must be invoked to determine whether any processes are still deadlocked.

- While killing the process one must take care, because partial completion of the process may leave the system in an inconsistent state.
- Many factors are involved in taking the decision of the process selection:
  1. Priority of the process.
  2. Cost involved in killing the process (how much the process has completed and how much time it will need to complete?).
  3. Which resources and what type of resources are held by the process?
  4. How many total processes will be involved in roll back?

1.5.2 Resource Pre-emption [April 16, 17, Oct. 17]

- Another method to recover the system from the state of deadlock is to preempt the resources from the processes one by one and allocate them to other processes until the circular wait condition is eliminated.
- In this method successively, some resources are preempted and given to other processes unless a deadlock is broken.
- This method requires following three major decisions to be taken:
  1. **Selecting a Victim or Select a Process for Preemption:** The choice/selection of processes and resources may be based on following criteria:
     - **(i) Priority:** Resources of low priority processes will be preempted.
     - **(ii) Own cost:** Process that has completed very less of its execution will preempt its resources.
     - **(iii) Cost affecting other processes**: How many maximum processes can restart their execution if resources of a particular process P are pre-empted?
  2. **Rollback of the Process:** The process, from which resource or resources are preempted, will not be able to continue execution, such process is roll backed partially or completely, to some safe state.
  3. **Prevent Starvation:** A process may be held for a long time waiting for its resources; then the process is said to be starved. So care must be taken so that some process is not selected again and again or repeatedly (leading to the situation of starvation) as a victim. To achieve this, the cost factor may contain the number of rollbacks of the process. This count will also be considered while selecting the victim.

- Let us summarize three different techniques for handling a deadlock:

Factors	Deadlock Prevention	Deadlock Avoidance	Deadlock Detection and Recovery
Method	This method ensures that at least one of four conditions should not be held.	This method does not grant a resource request if it leads to deadlock.	This method allows entering a deadlock but implements an algorithm which is invoked periodically to detect deadlock and recover from it.
Handling of process's requests	It prevents a deadlock by restraining how requests can be made. These restraints ensure that one of these conditions cannot occur.	The system requires prior information regarding overall potential use of each resource for each process. Then the system dynamically considers every request and decides whether it is safe to grant requests to a process.	Process's request can be satisfied by: Process preemption or Resource Preemption of other processes.
Advantages	• No preemption necessary. • Don't require runtime information. • Useful for processes that performs single burst of activity.	• No preemption necessary. • Proper resource utilization. • Good system throughput.	• Every request of process granted. • Facilitates online handling.
Dis-advantages	• Low Resource utilization. • Starvation. • low system throughput. • Disallow incremental resource requests.	• Future resource requirements must be known. • Sometimes processes requests are not granted, though resources are available. Since it may lead to a deadlock in future.	• Overhead includes run-time cost of maintaining necessary information. • Execution deadlock detection algorithms periodically incur overhead in computation time. • Losses inherent in recovery from deadlock.

UNIVERSITY SOLVED PROBLEMS [April 17, Oct. 17]

Problem 1: Consider a system with 5 processes {P_0, P_1, P_2, P_3, P_4} and four resource types {A, B, C, D}. There are 3 instances of type A, 14 instances of type B, 12 instances of type C and 12 instances of type D. The allocation and maximum demand matrices are as follows:

	Allocation			
	A	**B**	**C**	**D**
P_0	0	6	3	2
P_1	0	0	1	2
P_2	1	0	0	0
P_3	1	3	5	4
P_4	0	0	1	4

	Max			
	A	**B**	**C**	**D**
P_0	0	6	5	2
P_1	0	0	1	2
P_2	0	7	5	0
P_3	2	3	5	6
P_4	0	6	5	6

Answer the following questions using Banker's algorithm.

1. What are the contents of the need array?
2. Is a system in a safe state?
3. If the request from process P_4 arrives for (0, 0, 4, 1) can the request be immediately granted? **(April 05)**

Solution:

Total Resources	–	Total Allocation	=	Available
A B C D		A B C D	=	A B C D
3 14 12 12	–	2 9 10 12	=	1 5 2 0

1. The contents of Need Matrix,

   Need = Max – Allocation

	Need					Max					Allocation			
	A	**B**	**C**	**D**		**A**	**B**	**C**	**D**		**A**	**B**	**C**	**D**
P_0	0	0	2	0		0	6	5	2		0	6	3	2
P_1	0	0	0	0	=	0	0	1	2	–	0	0	1	2
P_2	0	7	5	0		1	7	5	0		1	0	0	0
P_3	1	0	0	2		2	3	5	6		1	3	5	4
P_4	0	6	4	2		0	6	5	6		0	0	1	4

2. Now using Banker's Safety Algorithm.

   Work = Available = {1, 5, 2, 0}

   Finish = {F, F, F, F, F}

(i) Let i = 0, Finish[i] = F,

Need (P_0) < = Work $\Rightarrow$ {0, 0, 2, 0} < = {1, 5, 2, 0}

So work = Work + Allocation

= {1, 5, 2, 0} + {0, 6, 3, 2}

Work = {1, 11, 5, 2}

(After P_0 finishes, it will release resources)

Finish = {T, F, F, F, F}

Safe sequence = {P_0}

(ii) Let i = 1, Finish[1] = F

Need (P_1) < = Work $\Rightarrow$ {0, 0, 0, 0} < = {1, 11, 5, 2}

So work = Work + Allocation

= {1, 11, 5, 2} + {0, 0, 1, 2}

Work = {1, 11, 6, 4}

(After P_1 finishes, it will release resources)

Finish = {T, T, F, F, F}

Safe sequence = {P_0, P_1}

(iii) Let i = 2, Finish[2] = F

Need (P_2) < = Work $\Rightarrow$ {0, 7, 5, 0} < = {1, 11, 6, 4}

So work = Work + Allocation

= {1, 11, 6, 4} + {1, 0, 0, 0}

Work = {2, 11, 6, 4}

(After P_2 finishes, it will release resources)

Finish = {T, T, T, F, F}

Safe sequence = {P_0, P_1, P_2}

(iv) Let i = 3, Finish[3] = F

Need (P_3) < = Work $\Rightarrow$ {1, 0, 0, 2} < = {2, 11, 6, 4}

So work = Work + Allocation

= {2, 11, 6, 4} + {1, 3, 5, 4}

Work = {3, 14, 11, 8}

Finish = {T, T, T, T, F}

Safe sequence = {P_0, P_1, P_2, P_3}

(v) Let i = 4, Finish[4] = F

Need (P_4) < = Work $\Rightarrow$ {0, 6, 4, 2} < = {3, 14, 11, 8}

So work = Work + Allocation

$\Rightarrow$ {3, 14, 11, 8} + {0, 0, 1, 4}

= {3, 14, 12, 12}

Finish = {T, T, T, T, T}

Safe sequence = {P_0, P_1, P_2, P_3, P_4}

$\therefore$ System is in safe state.

3. If a request (0, 0, 4, 1) from P_4 arrives. According to Resource Request Algorithm,
   (i) Request (P_4) = Need 4 $\Rightarrow$
   {0, 0, 4, 1} < = {0, 0, 6, 4} $\Rightarrow$ True go to Step 2
   (ii) Check if Request (P_4) < = Available
   {0, 0, 4, 1} < = {1, 5, 2, 0} $\Rightarrow$ No
   So P_4 has to wait since resources are not available.
   $\therefore$ Requests cannot be granted immediately.

Problem 2: Consider a system with 7 processes A through G and six types of resources R through W with one resource for each type. Resource ownership is as follows:

A holds R and wants S
B holds nothing but wants T
C holds nothing but wants S
D holds U and wants S and T
E holds T and wants V
F holds W and wants S
G holds V and wants U

Is the system deadlocked, and if so, which processes are involved?

Solution: Resource – Allocation graph is as follows:

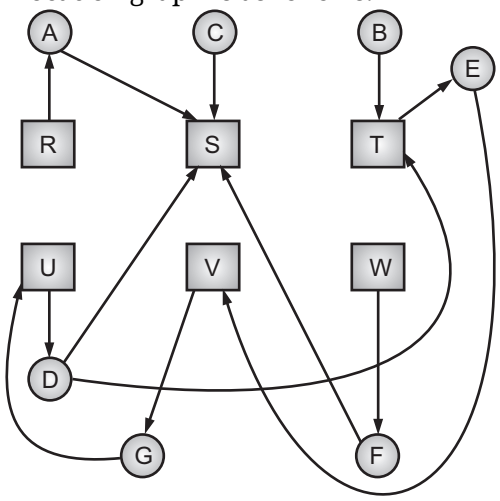


Fig.1.12 (a): Allocation Graph

Wait – for – Graph for above resource allocation graph is:

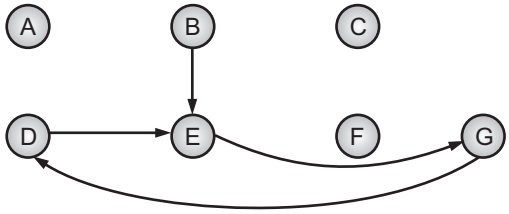


Fig. 1.12 (b): Resource Allocation Graph

As cycle exists in graph O $\rightarrow$ E – G – D; processes D, E, G are involved in a deadlock.

Problem 3: Consider a given snapshot of the system. A system has 5 processes and 3 types of resources A, B, C.

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

Max		
A	**B**	**C**
7	5	3
3	2	2
9	0	2
2	2	2
4	3	3

Available		
A	**B**	**C**
3	3	2

Answer the following questions using Banker's Algorithm:

1. What are the contents of the matrix need?
2. Is the system in safe state?
3. If a request from process P_1 arrives as (1, 0, 2) can the request be granted immediately?

Solution:

1. The contents of Need Matrix,
   Need = Max. – Allocation

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

2. Now using Banker's Safety Algorithm,

   Work = Available = {3, 3, 2}
   Finish = {F, F, F, F, F}

   (i) Let i = 0, Finish[0] = F
   Need (P_0) < = Work $\Rightarrow$ {7, 4, 3} < = {3, 3, 2}
   No, P_0 cannot be given resources. P_0 has to wait.

   (ii) Let i = 1, Finish[1] = F
   Need (P_1) < = Work $\Rightarrow$ {1, 2, 2} < = {3, 3, 2}
   $\therefore$ Work = Work + Allocation
   = (3, 3, 2) + (2, 0, 0)
   = (5, 3, 2)

[P_1 releases resources after its condition]

Finish = {F, T, F, F, F}

Safe sequence = {P_1}

(iii) Let i = 2, Finish[2] = F

Need (P_2) < = Work ⇒ {6, 0, 0} < = {5, 3, 2}

No, P_2 has a wait.

(iv) Let i = 3 Finish[3] = F

Need (P_3) < = Work ⇒ {0, 1, 1} < = {5, 3, 2}

∴ Work = Work + Allocation

= (5, 3, 2) + (2, 1, 1)

= (7, 4, 3)

After P_3 finishes, it releases resources

Finish = {F, T, F, T, F}

Safe sequence = {P_1, P_3}

(v) Let i = 4 Finish[4] = F

Need (P_4) < = Work ⇒ {4, 3, 1} < = {7, 4, 3} ⇒ Yes

∴ Work = Work + Allocation

= (7, 4, 3) + (0, 0, 2)

= (7, 4, 5)

After P_4 finishes, it releases resources

Finish = {F, T, F, T, T}

Safe sequence = {P_1, P_3, P_4}

(vi) Let, i = 0

Need (P_0) < = Work ⇒ {7, 4, 3} < = {7, 4, 5}

∴ Work = Work + Allocation

= (7, 4, 5) + (0, 1, 0)

= (7, 5, 5)

After P_0 finishes, it releases resources

Finish = {T, T, F, T, T}

Safe sequence = {P_1, P_3, P_4, P_0}

(vii) Let i = 2

Need (P_2) < = Work ⇒ {6, 0, 0} < = {7, 5, 5}

∴ So work = Work + Allocation

= (7, 5, 5) + (3, 0, 2)

= (10, 5, 7)

Finish = {T, T, T, T, T}

Safe sequence = {P_1, P_3, P_4, P_0, P_2}

∴ Yes, system is in safe state.

3. If a request from process P_1 arrives for (1, 0, 2), can it be granted immediately?
   According to Resource request algorithm:

   (i) Request (P_1) < = Need (P_1)
   (1, 0, 2) < = (1, 2, 2)

   (ii) Request (P_1) < = Available
   (1, 0, 2) < = (3, 3, 2) which is true.

   Then system pretends to fulfill request, then modify resource allocation state as follows:

   (a) Available = Available – Request (P_1)

   (b) Allocation (P_1) = Allocation (P_1) + Request (P_1)

   (c) Need (P_1) = Need (P_1) – Request (P_1)

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

Need		
A	**B**	**C**
7	4	3
0	2	0
6	0	0
0	1	1
4	3	1

–

Available		
A	**B**	**C**
2	3	0

   Now, check system is in safe state

   Work = Available = {2, 3, 0}
   Finish = {F, F, F, F, F}

   (i) Let i = 0, Finish[0] = F
   Need (P_0) < = Work $\Rightarrow$ {7, 4, 3} < = {2, 3, 0}

   No, P_0 has to wait for resources.

   (ii) Let i = 1, Finish[1] = F
   Need (P_1) < = Work $\Rightarrow$ {0, 2, 0} < = {2, 3, 0}
   $\therefore$ Work = Work + Allocation
   = (2, 3, 0) + (3, 0, 2)
   = (5, 3, 2)

   After P_1 finishes, it releases resources

   Finish = {F, T, F, F, F}
   Safe sequence = {P_1}

   (iii) Let i = 2, Finish[2] = F
   Need (P_2) < = Work $\Rightarrow$ {6, 0, 0} < = {5, 3, 2}

   No, P_2 has to wait for resources.

(iv) Let i = 3, Finish[3] = F

Need (P_3) < = Work $\Rightarrow$ {0, 1, 1} < = {5, 3, 2}

$\therefore$ Work = Work + Allocation

= (5, 3, 2) + (2, 1, 1)

= (7, 4, 3)

After P_3 finishes, it releases resources

Finish = {F, T, F, T, F}

Safe sequence = {P_1, P_3}

(v) Let i = 4, Finish[4] = F

Need (P_4) < = Work $\Rightarrow$ {4, 3, 1} < = {7, 4, 3}

$\therefore$ Work = Work + Allocation

= (7, 4, 3) + (0, 0, 2)

= (7, 4, 5)

After P_4 finishes, it releases resources

Finish = {F, T, F, T, T}

Safe sequence = {P_1, P_3, P_4}

(vi) Let i = 0, Finish[0] = F

Need (P_0) < = Work $\Rightarrow$ {7, 4, 3} < = {7, 4, 5}

$\therefore$ Work = Work + Allocation

= (7, 4, 5) + (0, 1, 0)

= (7, 5, 5)

After P_0 finishes, it releases resources

Finish = {T, T, F, T, T}

Safe sequence = {P_1, P_3, P_4, P_0}

(vii) Let i = 2, Finish[2] = F

Need (P_2) < = Work $\Rightarrow$ {6, 0, 0} < = {7, 5, 5}

$\therefore$ Work = Work + Allocation

= (7, 5, 5) + (3, 0, 2)

= (10, 5, 7)

Safe sequence = {P_1, P_3, P_4, P_0, P_2}

Yes, the system is in safe state. So requests can be granted immediately.

Problem 4: Consider the following snapshot of a system:

Process	Allocation	Max.	Available
	A B C	A B C	A B C
P0	2 3 2	9 7 5	3 3 2
P1	4 0 0	5 2 2	
P2	5 0 4	1 1 0 4	
P3	4 3 3	4 4 4	
P4	2 2 4	6 5 5	

Answer the following questions using Banker's algorithm:

(i) What is the content of the need Matrix?

(ii) Is the system in a safe state? If yes, give the safe sequence.

Solution:

1. The contents of Need Matrix,

   Need = Max – Allocation

	Need			Max			Allocation		
	A	B	C	A	B	C	A	B	C
P_0	7	4	3	9	7	5	2	3	2
P_1	1	2	2	5	2	2	4	0	0
P_2	6	0	0	11	0	4	5	0	4
P_3	0	1	1	4	4	4	4	3	3
P_4	4	3	1	6	5	5	2	2	4

2. Now using Banker's Safety Algorithm.

   Work = Available = {3, 3, 2}

   Finish = {F, F, F, F, F}

(i) Let i = 0, Finish[0] = F

   Need (P_0) < = Work $\Rightarrow$ {7, 4, 3} < = {3, 3, 2}

   No, P_0 cannot be given resources. P_0 has to wait.

(ii) Let i = 1, Finish[1] = F

   Need (P_1) < = Work $\Rightarrow$ {1, 2, 2} < = {3, 3, 2}

   $\therefore$ Work = Work + Allocation

   = (3, 3, 2) + (2, 0, 0)

   = (5, 3, 2)

   [P_1 releases resources after its condition]

   Finish = {F, T, F, F, F}

   Safe sequence = {P_1}

(iii) Let i = 2, Finish[2] = F

Need (P_2) < = Work $\Rightarrow$ {6, 0, 0} < = {5, 3, 2}

No, P_2 has a wait.

(iv) Let i = 3 Finish[3] = F

Need (P_3) < = Work $\Rightarrow$ {0, 1, 1} < = {5, 3, 2}

$\therefore$ Work = Work + Allocation

= (5, 3, 2) + (2, 1, 1)

= (7, 4, 3)

After P_3 finishes, it releases resources

Finish = {F, T, F, T, F}

Safe sequence = {P_1, P_3}

(v) Let i = 4 Finish[4] = F

Need (P_4) < = Work $\Rightarrow$ {4, 3, 1} < = {7, 4, 3} $\Rightarrow$ Yes

$\therefore$ Work = Work + Allocation

= (7, 4, 3) + (0, 0, 2)

= (7, 4, 5)

After P_4 finishes, it releases resources

Finish = {F, T, F, T, T}

Safe sequence = {P_1, P_3, P_4}

(vi) Let i = 0

Need (P_0) < = Work $\Rightarrow$ {7, 4, 3} < = {7, 4, 5}

$\therefore$ Work = Work + Allocation

= (7, 4, 5) + (0, 1, 0)

= (7, 5, 5)

After P_0 finishes, it releases resources

Finish = {T, T, F, T, T}

Safe sequence = {P_1, P_3, P_4, P_0}

(vii) Let i = 2

Need (P_2) < = Work $\Rightarrow$ {6, 0, 0} < = {7, 5, 5}

$\therefore$ So work = Work + Allocation

= (7, 5, 5) + (3, 0, 2)

= (10, 5, 7)

Finish = {T, T, T, T, T}

Safe sequence = {P_1, P_3, P_4, P_0, P_2}

$\therefore$ Yes, system is in safe state.

Problem 5: Consider a system with 5 processes {P_0, P_1, P_2, P_3, P_4} and four resource types {A, B, C, D}. There are 3 instances of type A, 14 instances of type B, 12 instances of type C and 12 instances of type D. The allocation and maximum demand matrices are as follows:

	Allocation			
	A	**B**	**C**	**D**
P_0	0	6	3	2
P_1	0	0	1	2
P_2	1	0	0	0
P_3	1	3	5	4
P_4	0	0	1	4

	Max			
	A	**B**	**C**	**D**
P_0	0	6	5	2
P_1	0	0	1	2
P_2	0	7	5	0
P_3	2	3	5	6
P_4	0	6	5	6

Answer the following questions using Banker's algorithm.

1. What are the contents of the need array?
2. Is a system in a safe state?
3. If the request from process P_4 arrives for (0, 0, 4, 1) can the request be immediately granted?

Solution:

Total Resources				–	Total Allocation				=	Available			
A	B	C	D		A	B	C	D	=	A	B	C	D
3	14	12	12	–	2	9	10	12	=	1	5	2	0

1. The contents of Need Matrix,

   Need = Max – Allocation

	Need					Max					Allocation			
	A	**B**	**C**	**D**		**A**	**B**	**C**	**D**		**A**	**B**	**C**	**D**
P_0	0	0	2	0		0	6	5	2		0	6	3	2
P_1	0	0	0	0	=	0	0	1	2	–	0	0	1	2
P_2	0	7	5	0		1	7	5	0		1	0	0	0
P_3	1	0	0	2		2	3	5	6		1	3	5	4
P_4	0	6	4	2		0	6	5	6		0	0	1	4

2. Now using Banker's Safety Algorithm.

   Work = Available = {1, 5, 2, 0}

   Finish = {F, F, F, F, F}

(i) Let i = 0, Finish[i] = F,

Need (P_0) < = Work $\Rightarrow$ {0, 0, 2, 0} < = {1, 5, 2, 0}

So work = Work + Allocation

= {1, 5, 2, 0} + {0, 6, 3, 2}

Work = {1, 11, 5, 2}

(After P_0 finishes, it will release resources)

Finish = {T, F, F, F, F}

Safe sequence = {P_0}

(ii) Let i = 1, Finish[1] = F

Need (P_1) < = Work $\Rightarrow$ {0, 0, 0, 0} < = {1, 11, 5, 2}

So work = Work + Allocation

= {1, 11, 5, 2} + {0, 0, 1, 2}

Work = {1, 11, 6, 4}

(After P_1 finishes, it will release resources)

Finish = {T, T, F, F, F}

Safe sequence = {P_0, P_1}

(iii) Let i = 2, Finish[2] = F

Need (P_2) < = Work $\Rightarrow$ {0, 7, 5, 0} < = {1, 11, 6, 4}

So work = Work + Allocation

= {1, 11, 6, 4} + {1, 0, 0, 0}

Work = {2, 11, 6, 4}

(After P_2 finishes, it will release resources)

Finish = {T, T, T, F, F}

Safe sequence = {P_0, P_1, P_2}

(iv) Let i = 3, Finish[3] = F

Need (P_3) < = Work $\Rightarrow$ {1, 0, 0, 2} < = {2, 11, 6, 4}

So work = Work + Allocation

= {2, 11, 6, 4} + {1, 3, 5, 4}

Work = {3, 14, 11, 8}

Finish = {T, T, T, T, F}

Safe sequence = {P_0, P_1, P_2, P_3}

(v) Let i = 4, Finish[4] = F

Need (P_4) < = Work $\Rightarrow$ {0, 6, 4, 2} < = {3, 14, 11, 8}

So work = Work + Allocation

$\Rightarrow$ {3, 14, 11, 8} + {0, 0, 1, 4}

= {3, 14, 12, 12}

Finish = {T, T, T, T, T}

Safe sequence = {P_0, P_1, P_2, P_3, P_4}

$\therefore$ System is in safe state.

3. If a request (0, 0, 4, 1) from P_4 arrives. According to Resource Request Algorithm,

(i) Request (P_4) = Need 4 $\Rightarrow$

{0, 0, 4, 1} < = {0, 0, 6, 4} $\Rightarrow$ True go to Step 2

(ii) Check if Request (P_4) < = Available

{0, 0, 4, 1} < = {1, 5, 2, 0} $\Rightarrow$ No

So P_4 has to wait since resources are not available.

$\therefore$ Requests cannot be granted immediately.

Problem 6: Consider the following sets P, R and E:

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

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

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

Also consider the following number of instances per resource type:

(i) One instance of resource type R_1 and R_3.

(ii) Two instances of resource type R_2.

(iii) Three instances of resource type R_4.

Construct the resource - allocation graph for the above problem. Check whether the system is in the deadlock. **[Oct. 16]**

Solution:

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

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

Resource instances:

- One instance of resource typeR_1
- Two instances of resource typeR_2
- One instance of resource typeR_3
- Three instances of resource typeR_4

Process States:

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

The resource allocation graph for above instance is as follows:

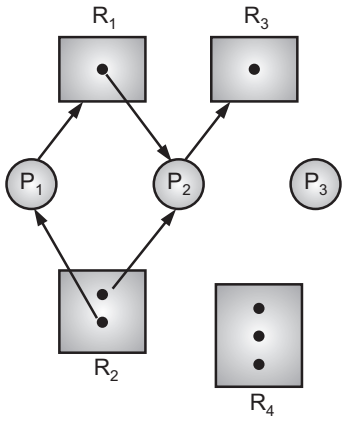


Fig. 1.13

Since, the above resource allocation graph does not contain cycles, so no process in the system is deadlocked. So there is no deadlock.

Problem 7: Consider a given snapshot of the system. A system has 5 processes and 3 types of resources A, B, C.

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

Max		
A	**B**	**C**
7	5	3
3	2	2
9	0	2
2	2	2
4	3	3

Available		
A	**B**	**C**
3	3	2

Answer the following questions using Banker's Algorithm:

1. What are the contents of the matrix needed?
2. Is the system in safe state?
3. If a request from process P1 arrives as (1, 0, 2) can the request be granted immediately?

Solution:

1. The contents of Need Matrix,

   Need = Max. – Allocation,

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

2. Now using Banker's Safety Algorithm,

Work = Available = {3, 3, 2}

Finish = {F, F, F, F, F}

(i) Let i = 0, Finish[0] = F

Need (P_0) < = Work $\Rightarrow$ {7, 4, 3} < = {3, 3, 2}

No, P_0 cannot be given resources. P_0 has to wait.

(ii) Let i = 1, Finish[1] = F

Need (P_1) < = Work $\Rightarrow$ {1, 2, 2} < = {3, 3, 2}

$\therefore$ Work = Work + Allocation

= (3, 3, 2) + (2, 0, 0)

= (5, 3, 2)

[P_1 releases resources after its condition]

Finish = {F, T, F, F, F}

Safe sequence = {P_1}

(iii) Let i = 2, Finish[2] = F

Need (P_2) < = Work $\Rightarrow$ {6, 0, 0} < = {5, 3, 2}

No, P_2 has a wait.

(iv) Let i = 3 Finish[3] = F

Need (P_3) < = Work $\Rightarrow$ {0, 1, 1} < = {5, 3, 2}

$\therefore$ Work = Work + Allocation

= (5, 3, 2) + (2, 1, 1)

= (7, 4, 3)

After P_3 finishes, it releases resources

Finish = {F, T, F, T, F}

Safe sequence = {P_1, P_3}

(v) Let i = 4 Finish[4] = F

Need (P_4) < = Work $\Rightarrow$ {4, 3, 1} < = {7, 4, 3} $\Rightarrow$ Yes

$\therefore$ Work = Work + Allocation

= (7, 4, 3) + (0, 0, 2)

= (7, 4, 5)

After P_4 finishes, it releases resources

Finish = {F, T, F, T, T}

Safe sequence = {P_1, P_3, P_4}

(vi) Let i = 0

Need (P_0) < = Work $\Rightarrow$ {7, 4, 3} < = {7, 4, 5}

$\therefore$ Work = Work + Allocation

= (7, 4, 5) + (0, 1, 0)

= (7, 5, 5)

After P_0 finishes, it releases resources

Finish = {T, T, F, T, T}

Safe sequence = $\{P_1, P_3, P_4, P_0\}$

(vii) Let i = 2

Need (P_2) < = Work $\Rightarrow$ {6, 0, 0} < = {7, 5, 5}

$\therefore$ So work = Work + Allocation

= (7, 5, 5) + (3, 0, 2)

= (10, 5, 7)

Finish = {T, T, T, T, T}

Safe sequence = $\{P_1, P_3, P_4, P_0, P_2\}$

$\therefore$ Yes, system is in safe state.

3. If a request from process P_1 arrives for (1, 0, 2), can it be granted immediately?

According to Resource request algorithm:

(i) Request (P_1) < = Need (P_1)

(1, 0, 2) < = (1, 2, 2)

(ii) Request (P_1) < = Available

(1, 0, 2) < = (3, 3, 2) which is true.

Then system pretends to fulfill request, then modify resource allocation state as follows:

(a) Available = Available – Request (P_1)

(b) Allocation (P_1) = Allocation (P_1) + Request (P_1)

(c) Need (P_1) = Need (P_1) – Request (P_1)

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

Need		
A	**B**	**C**
7	4	3
0	2	0
6	0	0
0	1	1
4	3	1

Available		
A	**B**	**C**
2	3	0

–

Now, check system is in safe state

Work = Available = {2, 3, 0}

Finish = {F, F, F, F, F}

(i) Let i = 0, Finish[0] = F

Need (P_0) < = Work $\Rightarrow$ {7, 4, 3} < = {2, 3, 0}

No, P_0 has to wait for resources.

(ii) Let i = 1, Finish[1] = F

Need (P_1) < = Work $\Rightarrow$ {0, 2, 0} < = {2, 3, 0}

$\therefore$ Work = Work + Allocation

= (2, 3, 0) + (3, 0, 2)

= (5, 3, 2)

After P_1 finishes, it releases resources

Finish = {F, T, F, F, F}

Safe sequence = {P_1}

(iii) Let i = 2, Finish[2] = F

Need (P_2) < = Work $\Rightarrow$ {6, 0, 0} < = {5, 3, 2}

No, P_2 has to wait for resources.

(iv) Let i = 3, Finish[3] = F

Need (P_3) < = Work $\Rightarrow$ {0, 1, 1} < = {5, 3, 2}

$\therefore$ Work = Work + Allocation

= (5, 3, 2) + (2, 1, 1)

= (7, 4, 3)

After P_3 finishes, it releases resources

Finish = {F, T, F, T, F}

Safe sequence = {P_1, P_3}

(v) Let i = 4, Finish[4] = F

Need (P_4) < = Work $\Rightarrow$ {4, 3, 1} < = {7, 4, 3}

$\therefore$ Work = Work + Allocation

= (7, 4, 3) + (0, 0, 2)

= (7, 4, 5)

After P_4 finishes, it releases resources

Finish = {F, T, F, T, T}

Safe sequence = {P_1, P_3, P_4}

(vi) Let i = 0, Finish[0] = F

Need (P_0) < = Work $\Rightarrow$ {7, 4, 3} < = {7, 4, 5}

$\therefore$ Work = Work + Allocation

= (7, 4, 5) + (0, 1, 0)

= (7, 5, 5)

After P_0 finishes, it releases resources

Finish = {T, T, F, T, T}

Safe sequence = {P_1, P_3, P_4, P_0}

(vii) Let i = 2, Finish[2] = F

Need (P_2) < = Work $\Rightarrow$ {6, 0, 0} < = {7, 5, 5}

$\therefore$ Work = Work + Allocation

= (7, 5, 5) + (3, 0, 2) = (10, 5, 7)

Safe sequence = {P_1, P_3, P_4, P_0, P_2}

Yes, the system is in a safe state. So request can be granted immediately.

Problem 8: Consider a system with four processes P_1, P_2, P_3, P_4 and four resource types A, B, C, D with one instance of each type. Resource ownership is as follows:

P1 holds A and wants C

P2 holds B

P3 holds D wants B

P4 holds C wants D

Is the system deadlocked?

(Draw resource-allocation graph and wait-for graph).

Solution:

R = {A, B, C, D}

E = {$P_1 \rightarrow C_1$, $P_4 \rightarrow D$, $P_3 \rightarrow B$, $A \rightarrow P_1$, $C \rightarrow P_4$, $D \rightarrow P_3$, $B \rightarrow P_2$}

Resource instances:

- One instance of resource type A,B,C,D

Process states:

- Process P_1 is holding an instance of resource type A and is waiting for an instance of resource type C.
- Process P_2 is holding an instance of B.
- Process P_3 is holding an instance of resource type D and is waiting for an instance of resource type B
- Process P_4 is holding an instance of resource type C and is waiting for an instance of resource type D

The resource allocation graph for above instance is as shown in Fig. 1.14.

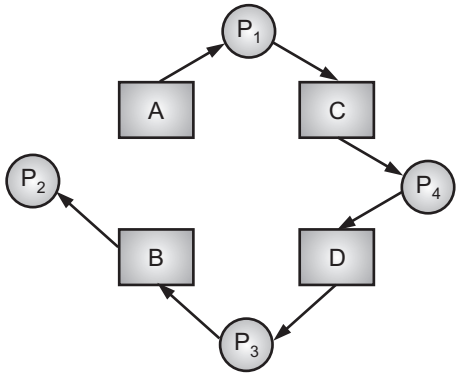


Fig. 1.14

Wait-for-Graph for above instance is as follows:

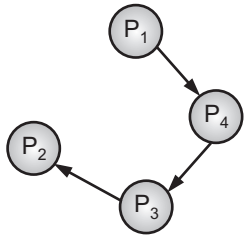


Fig. 1.15

As wait-for-Graph does not contain cycle, System has no deadlock.

Problem 9: Consider the following snapshot of the system:

Process	Allocation				Max			
	A	B	C	D	A	B	C	D
P_0	0	3	2	4	6	5	4	4
P_1	1	2	0	1	4	4	4	4
P_2	0	0	0	0	0	0	1	2
P_3	3	3	2	2	3	9	3	4
P_4	1	4	3	2	2	5	3	3
P_5	2	4	1	4	4	6	3	4

A system has total 10, 20, 13, 15 instances of resource types A, B, C, D respectively.
Answer the following using Banker's algorithm:

(i) What is the content of need and available matrix?

(ii) Is the system in safe state>

(iii) If a request from process P1 arrives for (2, 2, 3, 3) can it be granted immediately?

Solution:

Total Resources				–	Total Allocation				=	Available			
A	B	C	D		A	B	C	D	=	A	B	C	D
10	20	12	15	–	7	16	8	13	=	3	4	4	2

1. The contents of Need Matrix,
   Need = Max – Allocation,

	Need					Max					Allocation			
	A	B	C	D		A	B	C	D		A	B	C	D
P_0	6	2	2	0		6	5	4	4		0	3	2	4
P_1	3	2	4	3	=	4	4	4	4		1	2	0	1
P_2	0	0	1	0		0	0	1	2		0	0	0	0
P_3	0	6	1	2		3	9	3	4		3	3	2	2
P_4	1	1	0	1		2	5	3	3		1	4	3	2
P_5	2	2	2	1		4	6	3	5		2	4	1	4

2. Now using Banker's Safety Algorithm.

Work = Available = {3, 4, 4, 2}

Finish = {F, F, F, F, F, F}

(i) Let i = 0, Finish[i] = F,

Need (P_0) < = Work $\Rightarrow$ {6, 2, 2, 0} < = {3, 4, 4, 2}

No, P_0 has to wait for resources.

(ii) Let i = 1, Finish[1] = F

Need (P_1) < = Work $\Rightarrow$ {3, 2, 4, 3} < = {3, 4, 4, 2}

No, P_1 has to wait for resources.

(iii) Let i = 2, Finish[2] = F

Need (P_2) < = Work $\Rightarrow$ {0, 0, 1, 0} < = {3, 4, 4, 2}

So work = Work + Allocation

= {3, 4, 4, 2} + {0, 0, 0, 0}

Work = {3, 4, 4, 2}

(After P_2 finishes, it will release resources)

Finish = {F, F, T, F, F, F}

Safe sequence = { P_2}

(iv) Let i = 3, Finish[3] = F

Need (P_3) < = Work $\Rightarrow$ {0,6,1, 2} < = {3, 4, 4, 2}

No, P_3 has to wait for resources.

(v) Let i = 4, Finish[4] = F

Need (P_4) < = Work $\Rightarrow$ {1, 1, 0, 1} < = {3, 4, 4, 2}

So work = Work + Allocation

$\Rightarrow$ {3, 4, 4, 2} + { 1, 4, 3, 2}

= {4, 8, 7, 4}

(After P_4 finishes, it will release resources)

Finish = {F, F, T, F, T, F}

Safe sequence = { P_2,P_4}

(vi) Let i = 5, Finish[5] = F

Need (P_5) < = Work $\Rightarrow$ {2, 2, 2,1} < = {4, 8, 7, 4}

So work = Work + Allocation

$\Rightarrow$ {4, 8, 7, 4} + { 2, 4, 1, 4}

= {6, 12, 8, 8}

(After P_5 finishes, it will release resources)

Finish = {F, F, T, F, T, T}

Safe sequence = { P_2,P_4, P_5}

(vii) Let i = 0, Finish [0] = F

Need (P_0) < = Work $\Rightarrow$ {6, 2, 2, 0} } < = {6, 12, 8, 8}

So work = Work + Allocation

$\Rightarrow$ {6, 12, 8, 8} + { 0, 3, 2, 4}

= {6, 15, 10, 12}

(After P_0 finishes, it will release resources)

Finish = {T, F, T, F, T,T}

Safe sequence = { P_2,P_4, P_5, P_0}

(viii) Let i = 1, Finish[1] = F

Need (P_1) < = Work $\Rightarrow$ {3, 2, 4, 3} < = {6, 15, 10, 12}

So work = Work + Allocation

$\Rightarrow$ {6, 15, 10, 12} + { 1, 2, 0, 1}

= {7, 17, 10, 13}

(After P_1 finishes, it will release resources)

Finish = {T, T, T, F, T, T}

Safe sequence = { P_2, P_4, P_5, P_0,P_1}

(ix) Let i = 3, Finish[3] = F

Need (P_3) < = Work $\Rightarrow$ {0, 6, 1, 2} < = {7, 17, 10, 13}

So work = Work + Allocation

$\Rightarrow$ {7, 17, 10, 13} + { 3, 3, 2, 2}

= {10, 20, 12, 15}

(After P_3 finishes, it will release resources)

Finish = {T, T, T, T, T, T}

Safe sequence = { P_2, P_4, P_5, P_0, P_1, P_3}

$\therefore$ Yes, the system is in a safe state.

If request (2, 2, 3, 3) from P_1 arrives, then according to Resource Request Algorithm,

(i) Request (P_1) <= Need 4 $\Rightarrow$

(2, 2, 3, 3) < = {3, 2, 4, 3} $\Rightarrow$ True go to Step 2

(ii) Check if Request (P_1) < = Available

(2, 2, 3, 3) < = {3, 4, 4, 2} $\Rightarrow$ No

So, P_1 has to wait since resources are not available.

$\therefore$ Requests cannot be granted immediately.

PRACTICE QUESTIONS

Q. I Multiple Choice Questions:

1. Which is a situation that occurs/happens in operating system when any process enters a waiting state because another waiting process is holding the demanded resource?
   (a) Safe state (b) Deadlock
   (c) Both (a) and (b) (d) None of the mentioned
2. Under the standard mode of operation, any process in operating system may use a resource in only the below-mentioned sequence,
   (a) Request-Use-Release (b) Use-Release-Request
   (c) Release-Request-Use (d) None of the mentioned
3. The deadlock situation can only arise if all the following four conditions hold simultaneously,
   (a) Mutual Exclusion (at least one resource should be non-shareable (only one process at a time)).
   (b) Hold and wait (a process is holding at least one resource and is waiting for additional resources).
   (c) No preemption (a resource cannot be taken from a process unless the process releases the resource).
   (d) Circular wait (the set of processes are waiting for each other in the circular form).
   (e) All of the mentioned
4. In which deadlock method, the system checks each transaction before it is executed to make sure it does not lead to deadlock. If there is even a slight chance that a transaction may lead to deadlock in the future, it is never allowed to execute.
   (a) Deadlock Detection (b) Deadlock Avoidance
   (c) Deadlock Prevention (d) None of the mentioned
5. A useful tool in characterizing the allocation of resources to processes is called as,
   (a) Resource allocation graph (b) Resource detection graph
   (c) Resource prevention graph (d) None of the mentioned
6. In which deadlock method, as once the deadlock occurs additional information is required, like how the OS should use resources.
   (a) Deadlock Detection (b) Deadlock Avoidance
   (c) Deadlock Prevention (d) None of the mentioned

7. A deadlock detection algorithm wait-for graph is applicable when,
   (a) all resources have a single instance (b) all resources have multiple instances
   (c) Both (a) and (b) (d) None of the mentioned
8. If we preempt a resource from a process, the process cannot continue with its normal execution and it must be,
   (a) aborted (b) queued
   (c) roll backed (d) terminated
9. Which is a state if the system can allocate resources to every process in some order and still avoid a deadlock.
   (a) safe (b) unsafe
   (c) Both (a) and (b) (d) None of the mentioned
10. Which is an algorithm is a resource allocation and deadlock avoidance algorithm that tests for safety by simulating the allocation for predetermined maximum possible amounts of all resource?
    (a) resource allocation (b) Banker
    (c) Both (a) and (b) (d) None of the mentioned
11. A system has 3 processes sharing 4 resources. If each process needs a maximum of 2 units, then
    (a) Deadlock can never occur (b) Deadlock may occur
    (c) Dead has to occur (d) None of the mentioned
12. Every time a request for allocation cannot be granted immediately, the detection algorithm is invoked which help identify,
    (a) the set of processes in the deadlock queue
    (b) the specific process that caused the deadlock
    (c) the set of processes that have been deadlocked
    (d) None of the mentioned
13. A computer system has 6 tape drives, with 'n' processes competing for them. Each process may need 3 tape drives. The maximum value of 'n' for which the system is guaranteed to be deadlock free is?
    (a) 1 (b) 3
    (c) 4 (d) 2
14. An edge from process P_i to P_j in a wait for graph indicates that,
    (a) P_j is waiting for P_i to release a resource that P_j needs
    (b) P_i is waiting for P_j to leave the system
    (c) P_i is waiting for P_j to release a resource that P_i needs
    (d) P_j is waiting for P_i to leave the system

15. Deadlock recovery contains following which methods,
    (a) Process Termination (eliminates the deadlock, we can simply kill one or more processes. For this, we use two methods Abort all the deadlocked Processes and Abort one process at a time until deadlock is eliminated)
    (b) Resource Preemption (eliminates deadlocks using resource preemption, we preempt some resources from processes and give those resources to other processes. This method will raise three issues Selecting a victim, Rollback and Starvation)
    (c) Both (a) and (b)
    (d) None of the mentioned

Answers

1. (b)	2. (a)	3. (e)	4. (c)	5. (a)	6. (b)	7. (a)	8. (c)	9. (a)	10. (b)
11. (a)	12. (c)	13. (d)	14. (c)	15. (c)					

Q. II Fill in the Blanks:

1. A _____ happens in operating system when two or more processes need some resource to complete their execution that is held by the other process.
2. A state is _safe_ if the system can allocate resources to each process(up to its maximum requirement) in some order and still avoid a deadlock.
3. In _____ there should be a resource that can only be held by one process at a time. Mutual Exclusion
4. A deadlock can be detected by a resource _____ as it keeps track of all the resources that are allocated to different processes. Scheduler
5. To avoid deadlock there must be a _____ number of resources to allocate. fixed
6. The request and release of resources are done in operating system by system _____. calls
7. In a _____ a process is waiting for the resource held by the second process, which is waiting for the resource held by the third process and so on, till the last process is waiting for a resource held by the first process. circuler wait
8. A process can hold _____ resources and still request more resources from other processes which are holding them. multiple
9. Deadlock _____ is a set of methods to ensure that at least one of the necessary conditions cannot hold prevention
10. _____ state not always cause deadlock. unsafe
11. Data structures are used to implement the _____ algorithm includes Available, Max, Allocation and Need. Bankers
12. When all the low priority processes got blocked, while the high priority processes execute then this situation is termed as _____ Starvation

13. safty algorithm for finding out whether or not a system is in a safe state.
14. Deadlock can arise if following four conditions hold simultaneously Mutual Exclusion, Hold and Wait, No Preemption and Circular Wait.
15. A resource request algorithm and it is mainly used to determine whether requests can be safely granted or not.
16. Resources are preempted from the processes involved in the deadlock, preempted resources are allocated to other processes so that there is a possibility of recovering the system from deadlock.

Answers

1. deadlock	2. safe	3. Mutual exclusion	4. scheduler
5. fixed	6. calls	7. circular wait	8. multiple
9. prevention	10. Unsafe	11. Banker's	12. starvation
13. Safety	14. simultaneously	15. resource-request	16. preempted

Q. III State True or False:

1. Deadlock is a situation where a set of processes are blocked because each process is holding a resource and waiting for another resource acquired by some other process.
2. Banker's algorithm is a deadlock avoidance algorithm. It is named so because this algorithm is used in banking systems to determine whether a loan can be granted or not.
3. In a safe state, the operating system cannot prevent processes from requesting resources in such a way that any deadlock occurs. F
4. In deadlock detection method, deadlocks are allowed to occur. Then the state of the system is examined to detect that a deadlock has occurred and subsequently it is corrected.
5. The main objective of the deadlock prevention method is to violate any one condition among the four; because if any of one condition is violated then the problem of deadlock will never occur.
6. Deadlock can be avoided if some of the information about the processes is well known by the Operating System (OS) before the allocation of resources starts.
7. In hold and wait a process is holding at least one resource and waiting for resources.
8. An unsafe state is not a deadlocked state. F
9. In no-preemption a resource cannot be taken from a process unless the process releases the resource.
10. For deadlock recovery the systems uses killing the process method (killing all the processes one by one and Resource Preemption (resources are preempted from the processes involved in the deadlock, preempted resources are allocated to other processes so that there is a possibility of recovering the system from deadlock)

11. Resource request algorithm to avoid deadlock which will try to lend resources to the process and analyze whether the state after lending resources is safe state or unsafe state.
12. A deadlock is the occurs when a process enters into a available state because a resource request is being made by the other waiting process, which in turn become a waiting status for the other resource. F
13. Banker's safety Algorithm is used for deadlock recovery. F
14. If wait-for-graph contains cycle, then there exists a deadlock in the system.

Answers

1. (T)	2. (T)	3. (F)	4. (T)	5. (T)	6. (T)	7. (T)	8. (F)	9. (T)	10. (T)
11. (T)	12. (F)	13. (F)	14. (T)						

Q. IV Answer the following Questions:

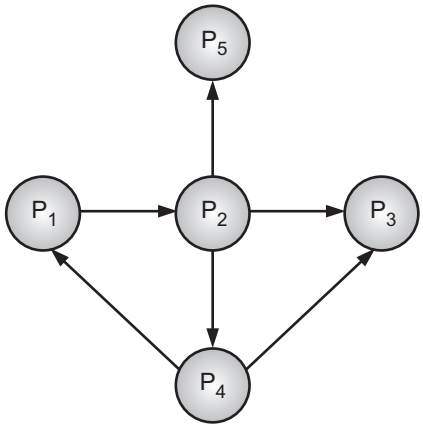
(A) Short Answer Questions:

1. Define deadlock.
2. When the deadlock occurs?
3. List necessary condition for deadlocks.
4. What is safety algorithm?
5. What is the purpose of safety resource algorithm?
6. Define system model.
7. What is deadlock detection?
8. Define deadlock avoidance.
9. What is killing process in deadlocks?
10. List methods for deadlock recovery.
11. Wait for graph is used for deadlock avoidance in the system True/False? Justify.
12. How to Bankers' algorithm used deadlocks?
13. Give the sequence of operations in which process can utilize a resource under normal mode of operations.
14. What is wait-for-graph?
15. Justify: "system must avoid deadlock".
16. Define safe and unsafe states.
17. What is safe sequence?
18. Define starvation in deadlocks.

(B) Long Answer Questions:

1. What is deadlock? How it can happen? Explain with example.
2. What necessary conditions for a deadlock to occur? Explain them in detail.
3. What is wait-for-graph? Explain with example.
4. Explain deadlock avoidance technique for a single instance of each resource type.

5. Describe method for deadlock recovery in detail.
6. Discuss the different data structures used in Bankers algorithm.
7. With the help of example explain safe state.
8. Describe Bankers' algorithm with wts its data structures with algorithm and example.
9. Draw wait-for-graph for the following:



10. Consider a system consisting of four resources of the same type that are shared by three processes, each of which needs at most two resources. Show that the system does not contain any deadlock.
11. With the help of example describe deadlock system model.
12. Describe resource request algorithm with example.

UNIVERSITY QUESTIONS AND ANSWERS

April 2016

1. Define starvation? **[1 M]**

Ans. Refer to Section 1.5.2, Point (3).

2. Consider a system with 7 processes A through G and six types of resources R through W with one resource for each type.

Resource ownership is as follows:

A holds R and wants S

B holds nothing but wants T

C holds nothing but wants S

D holds U and wants S and T

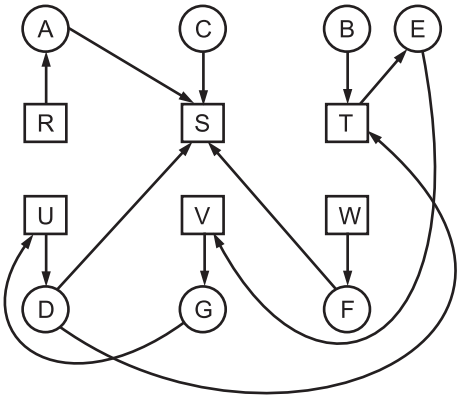
E holds T and wants V

F holds W and wants S

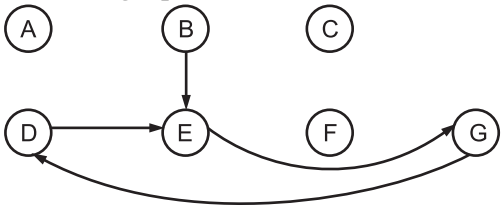
G holds V and wants U

Is the system deadlocked, and if so, which processes are involved? **[5 M]**

Ans. Resource allocation graph:



Wait-for-graph:



As cyclic exists in graph $0 \rightarrow \varepsilon \rightarrow G \rightarrow D$;

Processes D, E, G are involved in a deadlock.

3. Explain the term 'select a victim and rollback' in the context of deadlock recovery. **[4 M]**

Ans. Refer to Section 1.5.2, Point (1).

October 2016

1. "Wait for a graph is used for deadlock avoidance in the system" True/False? Justify. **[1 M]**

Ans. Refer to Section 1.4.1.

2. Consider the following sets P, R and E:

$P = [P_1, P_2, P_3]$

$R = [R_1, R_2, R_3, R_4]$

$E = [P_1 \rightarrow R_1, P_2 \rightarrow R_3, R_1 \rightarrow P_2, R_2 \rightarrow P_2, R_2 \rightarrow P_1]$

Also consider the following number of instances per resource type:

(i) One instance of resource type R_1 and R_2

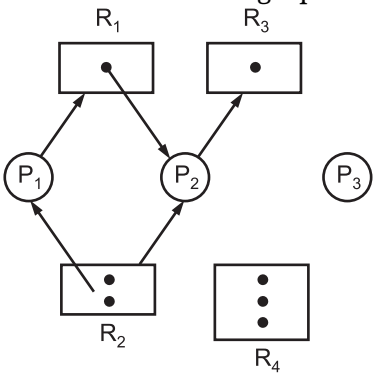
(ii) Two instances of resource type R_2.

(iii) Three instances of resource type R_4.

Construct the Resource allocation graph for the above problem.

Check whether the system is in the deadlock. **[5 M]**

Ans. Resource allocation graph:



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

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

Since, above resource allocation graph does not contain cycle - so no process in system is dead locked. So there is no deadlock.

3. What is deadlock? State different methods to handle a deadlock. **[1 M]**

Ans. Refer to Sections 1.1 and 1.3.

April 2017

1. Define starvation. **[1 M]**

Ans. Refer to Section 1.5.2, Point (3).

2. What is deadlock? Explain deadlock prevention strategies. **[5 M]**

Ans. Refer to Sections 1.1.1 and 1.3.1.

3. What is a wait-for-graph? **[2 M]**

Ans. Refer to Section 1.4.1.

4. Consider the following snapshot of the system:

	Allocation			Max			Available		
	A	B	C	A	B	C	A	B	C
P_0	0	1	0	7	5	3	3	3	2
P_1	2	0	0	3	2	2			
P_2	3	0	2	9	0	2			
P_3	2	1	1	2	2	2			
P_4	0	0	2	4	3	3			

Answer the following using Banker's algorithm:

(i) What is content of Need Matrix?

(ii) Is the system in a safe state? **[5 M]**

Ans. Refer to Solved Problems.

October 2017

1. What is rollback? **[1 M]**

Ans. Refer to Section 1.5.2, Point (2).

2. Consider a system with four processes P_1, P_2, P_3, P_4 and four resource type A, B, C, D with one instance of each type. Resource ownership is as follows:
P_1 holds A and wants C.
P_2 holds B
P_3 holds D and wants B
P_4 holds C and wants D
Is system deadlock? (Draw resource allocation graph and wait-for graph). **[5 M]**

Ans. Refer to Sections 1.2.2 and 1.4.1.

3. How the system is prevented from deadlock? **[2 M]**

Ans. Refer to Section 1.3.1.

April 2018

1. Define Request edge and Claim edge. **[1 M]**

Ans. Refer to Section 1.4.1.

2. Write a short note on deadlock prevention strategies. **[2 M]**

Ans. Refer to Section 1.3.1.

3. Consider the following snapshot of a system with 5 processes P0, P1, P2, P3 and P4 and three resource types A, B, C:

Process	Allocation			Max		
	A	B	C	A	B	C
P_0	2	3	2	9	7	5
P_1	4	0	0	5	2	2
P_2	5	0	4	11	0	4
P_3	4	3	3	4	4	4
P_4	2	2	4	6	5	5

Available		
A	B	C
3	3	2

Answer the following question using Banker's algorithm.
(i) What are the contents of need matrix?
(ii) Is the system in a safe state? If yes, find safe sequence. **[5 M]**

Ans. Refer to Solved Problems.

October 2018

1. State the necessary conditions for a deadlock to occur. **[1 M]**

Ans. Refer to Section 1.2.1.

2. Explain the different ways for deadlock recovery. **[5 M]**

Ans. Refer to Section 1.5.

3. Assume a system has 3 processes and 4 resources with 1, 2, 1, 4 instances of each resource respectively: **[4 M]**
Process P0 holds 1 instance of R1
Process P1 holds 1 instance of R0 and R1
Process P1 requests for 1 instance of R2
Process P2 holds 1 instance of R1
Check if deadlock is present. **[4 M]**

Ans. Refer to Section 1.1.

April 2019

1. List the deadlock handling techniques. **[1 M]**

Ans. Refer to Section 1.3.

2. Consider given snapshot of system. A system has 5 processes and 3 types of resources A, B, C. **[5 M]**

	Allocation		
	A	**B**	**C**
P_0	2	8	5
P_1	2	2	3
P_2	3	2	2
P_3	1	1	3
P_4	3	3	4

	Max		
	A	**B**	**C**
P_0	3	10	6
P_1	3	4	3
P_2	3	7	8
P_3	1	2	3
P_4	3	8	7

Available		
A	**B**	**C**
0	2	1

Answer the following questions using Banker's Algorithm:

(i) What is the content of need Matrix?

(ii) Is the system in safe state? If yes, give the safe sequence.

Ans. Refer to Solved problems.

■■■

CHAPTER 2

File System Management

Objectives...

- To understand Concept of Files (Attributes, Types and Operations)
- To learn different File Access Methods
- To study Basic Concepts of Directory (Structure and Types)
- To know File Allocation Methods
- To learn Methods for Free Space Management

2.0 INTRODUCTION

- The non-volatility of the memory enables the disks to store information indefinitely. This information can also be made available online all the time.
- Users think of all such information as files. It provides the mechanism for on-line storage of an access to both data and programs of the operating system and all users of the computer system. OS provides support for such management through a file system.
- File management is one of the basic and important features of an operating system. The operating system is used to manage files of computer system.
- Every operating system imposes a file system that helps to organize, manage and retrieve data on the disk. The file system resides permanently on the disk.
- The design of the file system involves following two key issues:
  1. The first issue includes defining a file and its attributes, operations that can be performed on a file, and the directory structure.
  2. The second issue includes creating data structures and algorithms for mapping the logical file system onto the secondary storage devices.
- A file is a collection of logically related information. A file is collection of specific information stored in the memory of computer system.
- File management is defined as, the process of manipulating files in a computer system, its management includes the process of creating, modifying and deleting the files.
- File system is the part of the operating system which is responsible for file management.

- File system provides a mechanism to store the data and access to the file contents including data and programs.
- File system is a method and data structure that the operating system uses to control how data is stored and retrieved.
- A computer file is a computer resource for recording data in a computer storage device.
- A directory organizes files and folders in a hierarchical (tree-like) manner. The directory contains information about the files, including attributes, location and ownership.

Need for File System:

- The most important user requirement of a computer system is to be able to store and retrieve data.
- In modern computers, the unit which is used in order to save data to hard disk drives, floppy disks or other storage devices is the file.
- The OS is responsible to retrieve, create, delete and save data to the files of the storage devices that are connected to the computer system on which it is installed.
- Moreover, it is responsible for managing a mechanism which allows the organization of the files in directories (folders).
- The way an OS can have access to the files and directories of a storage device is called file system. Well-known file systems for hard disk drives are the New Technology File System (NTFS), File Allocation Table (FAT) supported mainly by Microsoft Windows OSs, Unix File System (UFS) and Extended File System (EXT) for UNIX - like OSs and Hierarchical File System (HFS), and HFS Plus for Mac OS.
- File system is the software which empowers users and applications to organize and manage their files. So for most users, the file system is the most visible portion of an operating system.
- The file system consists of two distinct parts: a collection of files (each storing related data) and a directory structure, which organizes and provides information about all the files in the system.

File System Structure:

- File systems provide efficient and convenient access to the disk by allowing data to be stored, located, and retrieved easily.
- The file system design problem is divided into two groups.
  1. The first group is dealing with how the file system should look to the user. It includes the identification of a file and its attributes, operations allowed on a file and the directory structure.
  2. Second group is defining the algorithm and data structure used to map the logical file system onto the physical secondary storage devices.

3. The file system is divided into many different levels as shown in Fig. 2.1. These are:
   (i) Application Programs (AP)
   (ii) Logical File System (LFS)
   (iii) File Organization Module (FOM)
   (iv) Basic File System (BFS)
   (v) Input/Output Control (IOC)
   (vi) Devices.

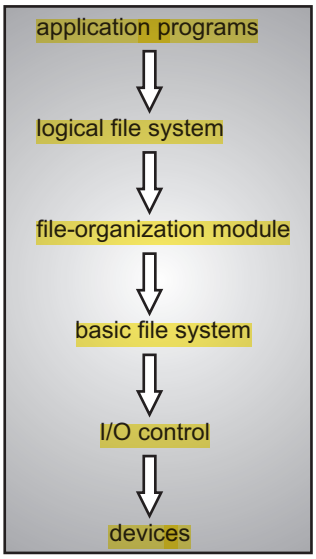


Fig. 2.1: Layered File System

- To create a new file application program, call a logical file system. Logical file systems know about directory structure.
- The information about symbolic file names is given to the file organization by a logical file system.
- Logical file systems get this information from directory structure. Logical file system is the highest level in the OS; it does protection, and security.
- File organization module knows about blocks and files. It is called by a logical file system to map directory, input/output to disk blocks. File organizing module generates address of blocks for basic file system to read.
- Basic file system does actual reading/writing of blocks to and from disk. Each block has a unique numeric disk address.
- Input/output controls are made up of device drivers and interrupt handlers and are responsible for actual transfer of data to memory and disk. At the end is the actual device involved in input /output.

2.1 FILE CONCEPT

- Computers can store information on various storage media, such as magnetic disk, magnetic tapes and optical disk.
- An operating system provides a uniform logical view of information storage. It abstracts physical properties of its storage devices to define a logical storage unit i.e., file.
- File is the storage unit of data in all the computer systems. A file is a named collection of related information that is recorded on secondary storage.
- Commonly, files represent programs (both source and object forms) and data. Data files may be numeric, alphabetic, alphanumeric, or binary.
- Files may be free form, such as text files, or may be formatted rigidly.
- In general, a file is a sequence of bits, bytes, lines or records, the meaning of which is defined by the file's creator and user. Files are mapped by the operating system onto physical devices.
- To be able to use some commands on a file, the operating system needs to know the structure of each file.
- For example, while printing a file; operating system prints garbage for a binary object file. It can be prevented if the operating system has been told that the file is a binary object file.
- There are two major disadvantages in letting the operating system know the structure of the files:
  1. There are various types of the files and for each type, the operating system requires a separate program to understand that file, so the size of operating system increases. It must have code to support all different types of files.
  2. Operating system will not be able to manage any new type of the file, until and unless code for the particular type is not added to the operating system.

2.1.1 Tape Based System

- Early file systems were tape based. Each file was implemented by mapping it into its own reel of tape. But it had number of following problems:
  - As only one file was stored on one tape, even the biggest file was using only two percent of the tape. Remedy for this solution was to find a mechanism by which more than one file can be stored on a tape.
  - Another problem was storing a very big file, partly stored on a number of tapes. To be able to do this, many systems provided multi-volume tape files.
  - There was a problem in determining which files are on which tapes. To get this information, a directory is added to tape. Some systems also called it VTOC (Volume Table Of Contents). Apart from names of all files in the tape, the directory could also have the same extra information about each file.

- For any operation on the file, the means is added by which it can be searched from the directory and then accessed.
- There was a need to separate the operations like rewind a file and rewind the tape.
- After a file is added to the end of the tape, the directory which is generally at the front of the tape is required to be updated.

2.1.2 Disk Based Systems

- When files are stored on hard disk, they are called disk based files. It allows direct access to the data. Storing multiple files on disk and drums is also comparatively easy.
- A disk has a number of tracks on it. All the tracks are further divided into a number of sectors.
- Number of tracks may vary from one disk to another. Large disks may have a number of platters. Each of it has two surfaces. Cylinder is a set of similar tracks on all platters.
- As a sector is made up of a fixed number of bytes, reading and writing is done in terms of sectors. Number of sectors per track and number of bytes per sector also vary from one disk to another.
- To address a particular sector, surface number, track number and sector number must be known.

  t is number of tracks per cylinder
  s is number of sectors per track
  i is cylinder
  j is surface
  k is sector

  The block number $b = k + s \times (j + I \times t)$
- Logical structure of drum and disk is almost the same. Drums give good performance but it is costly, they can be called as a one cylinder disk.
- Apart from providing direct access, disk has one benefit over tape. It is possible to read a block from the disk, modify it and rewrite it at the same place.
- Disk can have a list of files present on it.

Blocking:

- As a disk is divided into a number of sectors and sectors are fixed size, sectors can be called as a physical block on the disk. Such physical blocks are not on tape.
- The software program can decide the size of the physical blocks. Generally, the user data is in terms of records called logical records.
- Numbers of logical records are mapped into a physical block.
- Total number of logical blocks that can fit into one physical block is called packing density.

- The way logical records are fitted into physical blocks is called a packing technique.
- Size of logical records, physical blocks and packing technique determine packing density.

2.1.3 File Attributes [Oct. 16, April 17, 18]

- A file is named for the convenience of its human users, and is referred to by its name.
- A file's attributes vary from one operating system to another but typically consist of these:
  1. **Name:** The symbolic file name, usually a string of characters.
  2. **Identifier**: It is a unique tag, usually a number that identifies the file within the file system.
  3. **Type:** This information is needed for systems that support different types of files.
  4. **Location:** It is a pointer to a device and location of the file on that device.
  5. **Size:** The current size of the files (in bytes, word or blocks).
  6. **Protection:** Access-control information determines who can do reading, writing, executing and so on.
  7. **Time, date and user identification:** It is information of time and date of creation, last modification and last use. These data can be useful for protection, security, and usage monitoring.
- The information about all files is kept in the directory structure, which also resides on secondary storage. Typically, directory entry consists of the file's name and its unique identifier. The identifier in turn locates the other file attributes.

2.1.4 File Operations [Oct. 16]

- A file is an abstract data type. To define the file properly, we need to consider the operation that can be performed on files.
- Each OS uses its own operation to retrieve the data stored in a file. A file is a collection of logically related data that is recorded on the secondary storage in the form of sequence of operations.
- The content of the files are defined by its creator who is creating the file. The various operations which can be implemented on a file such as read, write, open and close etc. are called file operations.
- These operations are performed by the user by using the commands provided by the operating system.
- The operating system can provide system calls to create, write, read, reposition, delete, and truncate files.

- Some common file operations are explained below:
- **Creating a File (Create Operation):** Create operation is used to create a new file in the file system. Following is the two steps are necessary to create a file:
  1. Find space for a new file in the file system.
  2. Make an entry for the new file in the directory.
- **Writing a File (Write Operation):** Write operation is used to write the information into a file. To write a file follow following steps:
  1. To write a file, the name of the file and the information to be written to the file should be provided to the system call.
  2. Search the file name in the directory to find the file's location.
  3. Place the write pointer to the location in the file where the next write is to take place.
  4. Update the write pointer whenever a write occurs.
- **Reading a File (Read Operation):** Read operation reads the contents from a file. To read from a file,
  1. Specify the name of the file and where (in memory) the next block of the file should be put is given to system call.
  2. Search the file name in the directory.
  3. Place the read pointer to the location in the file where the next read is to take place.
  4. Update the read pointer whenever the read occurs.
- Because a process is usually either reading from or writing to a file, the current operation location can be kept as a per-process current-file-position pointer.
- Both the read and write operations use this same pointer, saving space and reducing system complexity.
- **Repositioning or Seeking within a File (Reposition or Seek Operation):** The seek system call re-positions the file pointers from the current position to a specific place in the file i.e. forward or backward depending upon the user's requirement. This operation is generally performed with those file management systems that support direct access files. To repositioning a file:
  1. Search the file name in the directory.
  2. Current-file-position pointer is repositioned to a given value.

  Repositioning within a file need not involve any actual I/O. This file operation is also known as a file seeks.
- **Deleting a File (Delete Operation):** To delete a file,
  1. Search the directory for the named file.
  2. Release all the file space, so that it can be refused by the other files.
  3. Erase the directory entry.

- **Truncating a File (Truncate Operation):** Truncating is simply deleting the file except deleting attributes. The user may want to erase the contents of a file and then recreate it, this function allows all attributes to remain unchanged – except for file length. Length of file is set to zero and its file space is released.
- **Opening a File (Open Operation):** Open operation is the common operation performed on the file. Once the file is created, it must be opened before performing the file processing operations. When the user wants to open a file, it provides a file name to open the particular file in the file system. It tells the operating system to invoke the open system call and passes the file name to the file system.
- **Closing a File (Close Operation):** When the processing of the file is complete, it should be closed so that all the changes made permanent and all the resources occupied should be released. On closing it de-allocates all the internal descriptors that were created when the file was opened.
- **Append operation** adds data to the end of the file. **Rename operation** is used to rename the existing file.
- Most of the file operations mentioned involve searching the directory for the entry associated with the named file.
- To avoid this constant searching, many systems require that an open() system call be made before a file is first used actively.
- The operating system keeps a small table, called the open-file table, containing information about files. When a file operation is requested, the file is specified via an index into this table, so no searching is required.
- When a file is no longer being actively used, it is closed by the process, and the operating system removes its entry from the open-file table. Create and delete are system calls that work with closed rather than open files.
- The implementation of the open() and close() operations is more complicated in an environment where several processes may open the file at the same time.
- This may occur in a system where several different applications open the same file at the same time. The operating system uses two levels of internal table namely a per-process table and a system –wide table.
- The **per-process table** tracks all files that a process has open as shown in Fig. 2.2.
- The per-process table stores the information regarding the use of the file by the process. For example, the current file pointer for each file, access rights and accounting information etc.

- Each entry in the per-process table in turn points to a **system-wide open–file table.** It contains process- independent information, such as the location of the file on disk, access dates and file size etc.
- Once, file has been opened by process, this table includes an entry for the file.

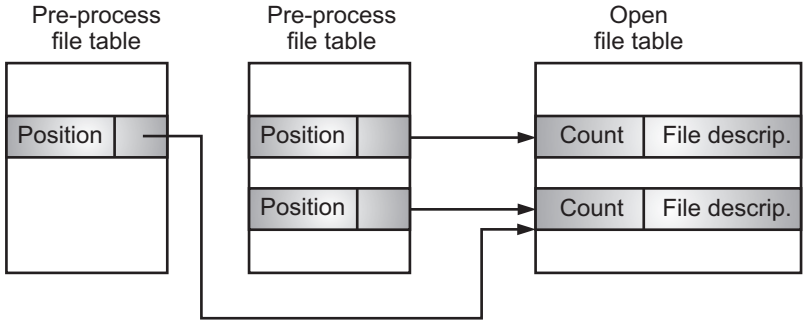


Fig. 2.2: OS Data Structure for Files

- The following information associated with open file kept in open-file table is as follows:
  1. **File Pointer:** On systems that do not include a file offset as part of the read() and write() system calls, the system must track the last read – write location as a current file position pointer.
  2. **File-open Count:** This counter tracks the number of opened and closed files by each process. It reaches zero on the last close, then the system can remove the entry from the open-file table.
  3. **Disk Location of the File:** Most file operations require the system to modify data within the file. The information needed to locate the file on disk is kept in memory so that the system does not have to read it from disk for each operation.
  4. **Access Rights:** Each process opens a file in an access mode. This information is stored on the per-process table so the operating system can allow or deny subsequent I/O requests.

2.1.5 File Types

- File type refers to the ability of the operating system to distinguish different types of file such as text files, source files and binary files etc.
- If an operating system recognizes the types of file, it can then operate on the file in reasonable ways.
- A common technique for implementing file types is to include the type as part of the file name.
- The file name is split into two parts – a name and an extension, usually separated by a period character. In this way, we can recognize the file type from the name of the file itself.

File Type	Usual Extension	Function
executable	exe, com, bin or none	Ready-to-run machine-language program
object	obj, o	Compiled, machine language, not linked
source code	c, cc, java, pas, asm, a	Source code in various language
batch	bat, sh	Commands to the command interpreter
text	txt, doc	Textual data, documents
word processor	wp, tex, rtf, doc	Various word-processor formats
library	lib, a, so, dil	Libraries of routines for programmers
print or view	ps, pdf, jpg	ASCII or binary file in a format for printing or viewing
archive	arc, zip, tar	Related files grouped into one file, sometimes compressed, for archiving or storage
multimedia	mpeg, mov, rm, mp3, avi	binary file containing audio or A/V information

- Operating system like MS-DOS and UNIX have the following types of files:
  1. **Ordinary Files:** These are the files that contain user information. These may have text, databases or executable programs. The user can apply various operations on such files like add, modify, delete or even remove the entire file.
  2. **Directory Files:** These files contain a list of file names and other information related to these files.
  3. **Special Files:** These files are also known as device files. These files represent physical devices like disks, terminals, printers, networks, tape drives etc. These files are of two types:
     (i) Character Special Files: Data is handled character by character as in case of terminals or printers.
     (ii) Block Special Files: Data is handled in blocks as in the case of disks and tapes.

2.2 ACCESS METHODS [Oct. 17, April 19]

- Stored information in the file must be accessed and read into the computer's memory. It can be accessed in several ways like sequentially, direct or by other methods.
- File access mechanism refers to the manner in which the records of a file may be accessed.

2.2.1 Sequential Access

- It is the simplest access method. In sequential access the information in the file is processed in order, one record after the other.
- In sequential access a fixed format is used for records and the key field uniquely identifies the record.
- Read and write in sequential access make up the bulk of the operation on the file which always starts from beginning of the file.
  - **read next** operation reads the next portion of the file and automatically advances the file pointer, which tracks the I/O location.
  - **write next** operation appends to the end of the file and advances to the end of the newly written block(i.e. the new end of the file).
  - **Rewinding file** reset file pointer to the beginning.
- Sequential access, which is depicted in Fig. 2.3, is based on a tape model of a file and works as well on sequential – access devices.

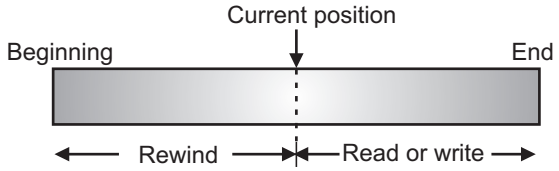


Fig. 2.3: Sequential-Access File

Advantages:

1. It is the simplest method of searching suitable for tape based systems.
2. If the application needs scanning of all records from a file, then sequential access is best.

Disadvantages:

1. It is more time consuming since reading, writing and searching always start from the beginning of the file.
2. Insertion and deletion operation take more time.

2.2.2 Direct Access

- Another method is direct access (or relative access). Random access file organization provides, accessing the records directly.
- A file is made up of fixed length logical records that allow the program to read and write records rapidly in any order.
- This method is based on a disk model of a file, since disks allow random access to any file block.

- For direct access, the file is viewed as a numbered sequence of block records. So the file operation must include the block number as parameter.
  - **read n**, where n is the block number, file pointer is placed at n^{th} block and reads data of that block.
  - **write n** operation writes data into n^{th} block.
- The block number provided by the user to the operating system is normally a relative block number which acts as an index relative to the beginning of the file.
- Thus the first relative block of the file is 0; the next is 1, and so on, even though the actual absolute disk address of the block may be different.
- The use of relative block numbers allows the operating system to decide where the file should be placed and helps to prevent the user from accessing portions of the file system that may not be part of his/her file.
- Fig. 2.4 shows direct access, the file pointer can be positioned randomly as required for read and write operations. This can be done without any particular order in positioning.

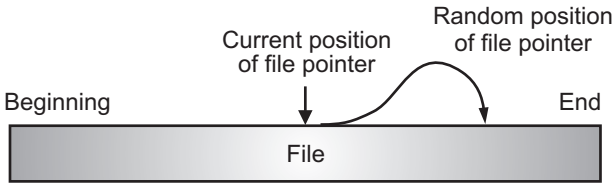


Fig. 2.4: Direct/Random Access File

Advantages:

1. We can access any record randomly.
2. This method is more suitable for disk based systems.

2.2.3 Indexed Access

- The basic form of index includes a record key and storage addresses for a record. To find a record when the storage address is unknown, it is necessary to scan the records.
- An index is a separate file from the master file. To find a specific record, the index is first searched to find the key of the record required.
- When it is found, the corresponding storage address is noted and then the program accesses the record directly.
- This method uses sequential scan of index and direct access to appropriate records. This is a fast way to search or access a file.
- There are two types of indexed access files:
  1. **Indexed Non-Sequential Access:**
     - In this method, the master file is not in any specific order.

- There is one entry in the index for every record in the master file.
- When the user wants to search any record, the operating system finds the address using its key value from the index table and moves directly to that position.

2. Indexed Sequential Access:

- In this access method, the master file is sorted according to key values and the index table contains only some keys of the records.
- When the user wants to search any record, the operating system finds address using its key value from the index table and apply sequential search to find out the record from master table.

- The ISAM (Indexed Sequential Access Method) uses a small master index which points to parts of the cylinder index.
- The cylinder index blocks point to track index and tracks index blocks point to the actual file as shown in Fig. 2.5.
- The file is kept sorted on a defined key. To find a particular record, we first make a binary search of the master index, which provides the block number of the cylinder index.
- Then make a binary search of the cylinder index, which provides a block number of track index. Again within a track, binary search is used to obtain a desired record.

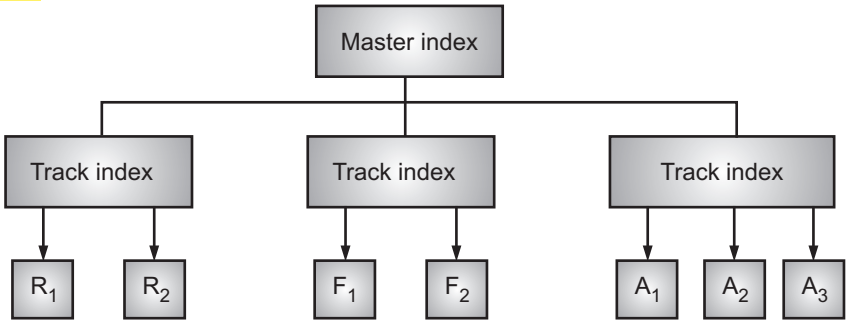


Fig. 2.5: Indexed Sequential Access Method

2.3 DIRECTORY STRUCTURE [April 18]

- Computer system stores millions of files on disk. To manage all this data, we need to organize them. This organization involves use of directories.
- So a directory can be viewed as a collection of nodes containing information about files as shown in Fig. 2.6.
- File system directories usually include both files and other directories leading to tree -like structures of the file system.

- A file can be defined as a data structure which stores the sequence of records. Files are stored in a file system, which may exist on a disk or in the main memory. The collection of files is known as Directory.
- The hierarchical directory systems form the most common structure used in modern OSs.

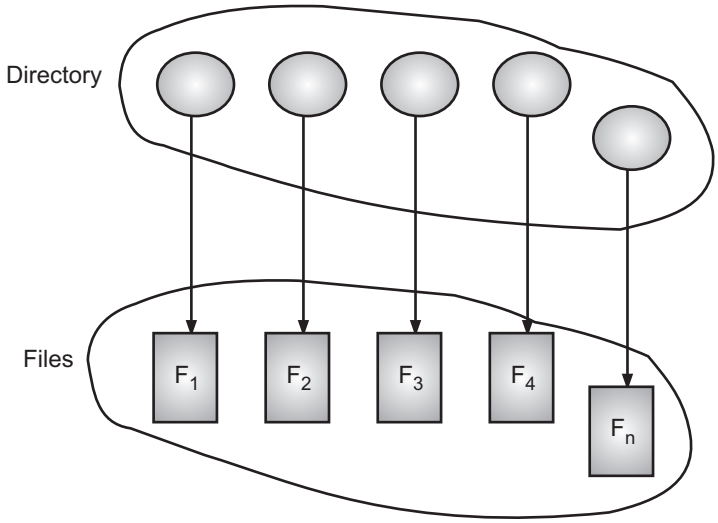


Fig. 2.6: Directory Structure

- A directory is a file system cataloging structure which contains references to other computer files, and possibly other directories.
- A directory structure is the way an operating system arranges files that are accessible to the user. Files are typically displayed in a hierarchical tree structure.

2.3.1 Storage Structure

- A disk can be used in its entirety for a file system.
- Sometimes, it is desirable to place multiple file systems on a disk or to use parts of a disk for a file system. These parts are known as partitions, slices etc.
- These partitions can be combined to form a larger structure known as volume. Thus volume is a chunk of storage that holds a file system.
- Each volume that contains a file system must also contain information about files in the system. This information is kept in entries in a device directory or volume table of contents.
- The device directory records information such as name, location, size and type for all files in the file system.
- Volumes can also store multiple operating systems, allowing a system to boot and run more than one.

- The Fig. 2.7 shows typical file organization.

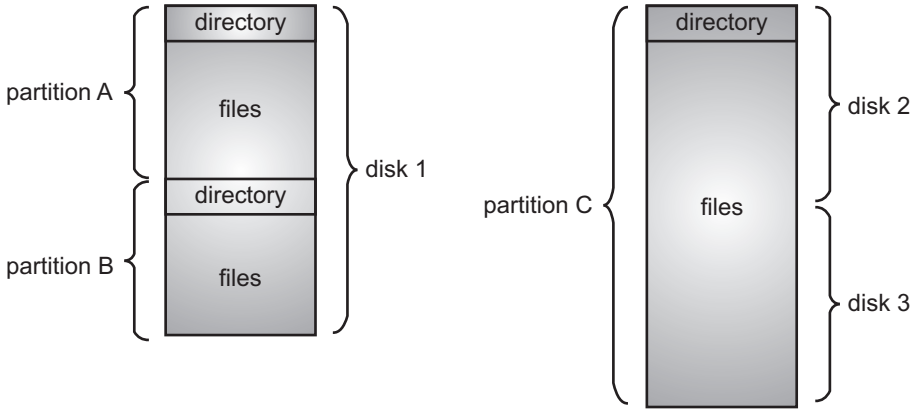


Fig. 2.7: A Typical File System Organization

2.3.2 Directory Overview

- The directory can be viewed as a symbol table that translates file names into their directory entries.
- Directory can be defined as, the listing of the related files on the disk.
- When considering a particular directory structure, we need to keep in mind the operations that are to be performed on a directory:
  1. **Search for a file:** To find the entry for a particular file or find all files whose names match a particular pattern.
  2. **Create a file:** New files need to be created and added to the directory.
  3. **Delete a file:** To remove a file entry from a directory when it is no longer needed.
  4. **List a directory:** To list the files in a directory along with its information.
  5. **Rename a file:** Renaming a file may also allow its position within the directory structure to be changed.
  6. **Backup:** We may wish to access every directory and every file within a directory structure. For reliability, it is a good idea to save the contents and structure of the entire file system at regular intervals.
- Several users create thousands and millions of files in a file system. To organize these files properly file system uses directory structure.
- In the following sections, the most common schemes for defining the logical structure of a directory are described.

2.3.2.1 Single-Level Directory

- The simplest directory structure is the single – level directory. All files are contained in the same directory, which is easy to support and understand.

- The entire system will contain only one directory in a single–level directory which is supposed to mention all the files present in the file system.
- The directory contains one entry per each file present on the file system.

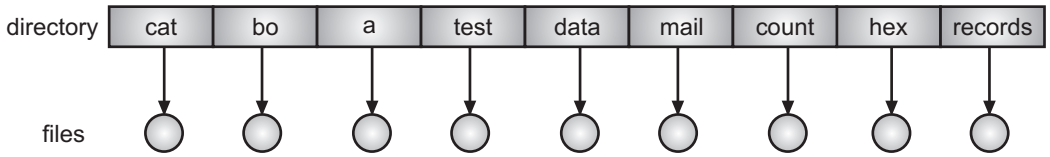


Fig. 2.8: Single-level Directory

Advantages:

1. It is a single directory, so its implementation are very easy and simple.
2. File operations like creation, searching, deletion is very simple since we have only one directory.

Disadvantages:

1. As all files are in the same directory, they must have unique names. We cannot have two files with the same name.
2. The directory may be very big therefore searching for a file may take so much time.
3. Users can not create their own directory.

2.3.2.2 Two-Level Directory

- In the two-level directory, each user has his/her own User File Directory (UFD). The UFDs have similar structure, but each lists only the files of a single user.
- When a user job starts or a user logs in, the system's Master File Directory (MFD) is searched.
- The MFD is indexed by user name or account number and each entry points to the UFD for the user.
- When a user refers to a particular file, only his UFD is searched. Different users may have files with the same name as long as all the file names within each UFD are unique.
- To create a file for the user, the operating system searches only that user's UFD to verify whether another file of that name exists.
- To delete a file, the operating system limits its search to the local UFD; thus it cannot accidentally delete another user's file that has the same name.

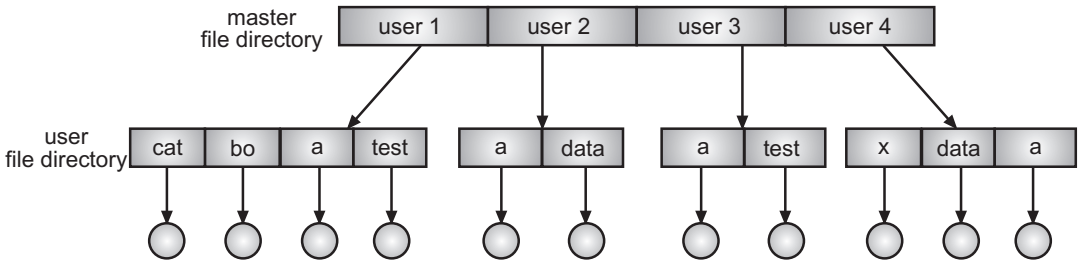


Fig. 2.9: Two-level Directory Structure

Advantages:

1. Different users can have the same directory as well as file name.
2. Searching of files becomes easier due to path names and user-grouping.
3. This method isolates one user from another and protects user's files.

Disadvantages:

1. As this structure isolates one user from another, sharing of tasks and accessing one another's files is not possible.
2. If access is to be permitted, the user must have the ability to name a file in another user's directory (i.e. user must know the pathname of the file desired).
3. In the case of system's files, which are required by all users, it has to be copied in each user's directory. It leads to wastage of memory.

2.3.2.3 Tree-Structured Directories [April 16, 17, 18]

- The generalization of two level directory structure is an arbitrary tree structure directory.
- This generalization allows users to create their own subdirectories and to organize their files accordingly.
- A tree structure directory is the most common directory structure. The tree has a root directory, and every file in the system has a unique path name.
- A directory (or subdirectory) contains a set of files or subdirectories. It is simply another file, but it is treated in a special way.
- All directories have the same internal format. One bit in each directory entry defines the entry as a file (0) or as a sub directory (1).
- Special system calls are used to create, delete and change directories.
- Path names can be of two types i.e., absolute and relative.
  1. **An absolute path name** begins at the root and follows the path down to the specified file, giving the directory names on the path.
  2. **A relative path name** defines a path from the current directory to that specified file.
- For example, in the tree-structured file system of Fig. 2.10. If the current directory is root/spell/mail, then the relative path name prt/first/ refers to the same file as does the absolute path name root/spell/ mail/prt/first.

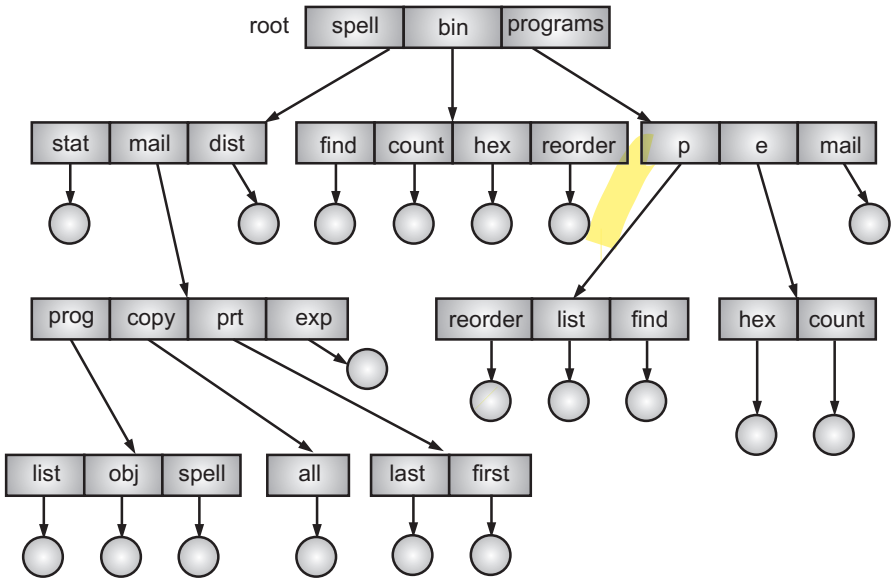


Fig. 2.10: Tree-structured Directory Structure

- If the directory is empty, its entry in the directory that contains it can be deleted. But if the directory to be deleted is not empty, then two approaches are taken:
  1. Operating systems such as MS-DOS will not delete a directory unless it is empty. Thus, to delete a directory, the user must first delete all the files in that directory. If sub-directories exist, then the same procedure must be applied recursively to them, so that they can be deleted. But it results in a significant amount of work.
  2. The other approach used by UNIX 'rm command' is to provide an option: when a request is made to delete a directory, all that directory and sub-directories are also to be deleted.

Advantages:

1. Users can create a directory as well as subdirectory.
2. Users can access the files of other users.
3. Very generalize, since full path names can be given.

Disadvantages:

1. The tree structure can create duplicate copies of the files.
2. The users could not share files or directories.
3. It is inefficient, because accessing a file may go under multiple directories.

2.3.2.4 Acyclic-Graph Directories [Oct. 17, April 18]

- The tree structured directory system doesn't allow the same file to exist in multiple directories therefore sharing is a major concern in tree structured directory systems.

- We can provide sharing by making the directory an acyclic graph. A graph without cycles is called an acyclic graph.
- An acyclic graph directory is a generalization of the tree-structured directory. Acyclic graph directories allow sharing of files.
- An acyclic graph that is, a graph with no cycles allows directories to share subdirectories and files as shown in the Fig. 2.11.

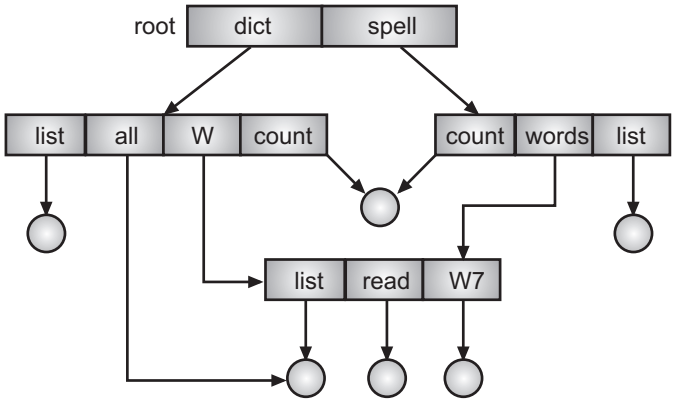


Fig. 2.11: Acyclic-graph Directory Structure

Implementation of Shared Files:

- It is important to note that a shared file (or directory) is not the same as two copies of the file.
- With two copies, each programmer can view a copy rather than the original one, but if a programmer changes the file, the changes will not appear in others copies. So inconsistency is big problem with copies and memory wastage.
- With a shared file, only one actual file exists, so any changes made by one person are immediately visible to the other.
- Sharing is particularly important for subdirectories; a new file created by one person will automatically appear in all the shared subdirectories.
- But also with a single copy, several concurrent updates to a file may result in the user obtaining incorrect information, and the file being left in an incorrect state.
- Shared files and subdirectories can be implemented in several ways:
  1. A common way is to create a new directory entry called a link. A link is effectively a pointer to another file or subdirectory. For example, a link may be implemented as an absolute or a relative path name. When a reference to a file is made, we search the directory. In the directory, entry is marked as link, and then the link is resolved by using that path name to locate the real file.
  2. Another common approach to implementing shared files is simply to duplicate all information about them in both sharing directories. Thus both entries are identical and equal.

Deletion of Shared File:

- Another problem involves deletion of shared files. One possibility is to remove the file whenever anyone deletes it, but this action may leave a dangling pointer to the non-existent file.
- To overcome this problem, a reference count is maintained for shared files. Whenever a user shares a file, reference count is incremented and whenever a user deletes a file, it is decremented.
- When reference count becomes zero, at that time only the file is actually deleted.

Advantages:

1. It is more flexible than tree-structure directories.
2. It allows sharing of files and directories.
3. Implementation of the algorithm is simple to traverse the graph.

Disadvantages:

1. It is difficult to ensure that there are no cycles.
2. Implementation of shared files is complicated.

2.3.2.5 General Graph Directory

- When we add links to the existing tree structure directories, the tree structure is destroyed, resulting in a simple graph directory as shown in Fig. 2.12.
- In general graph directory structure, cycles are allowed within a directory structure where multiple directories can be derived from more than one parent directory.
- If cycles are allowed, then algorithm should take care of the following:
  - Not allowing the reference to the file to go in an infinite loop, because of cycle.

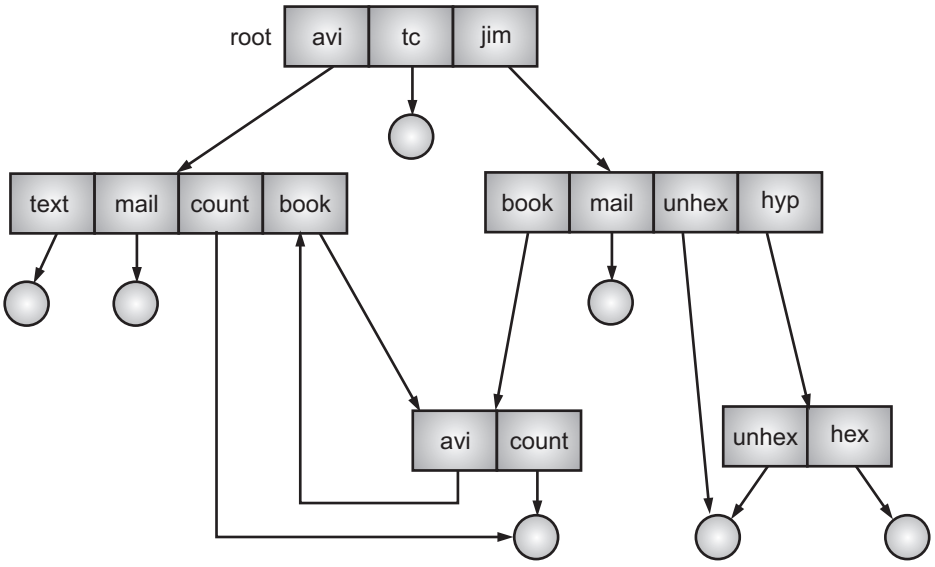


Fig. 2.12: General graph directory

- Garbage collection method should be implemented in case of deletion of shared files.
- Garbage collection involves traversing the entire files system, marking everything that can be accessed. Then, a second pass collects everything that is not marked onto a list of free space.

Advantages:

1. Allows sharing of files.
2. Directories are more flexible than acyclic graph structure.
3. It is more flexible than other directories structure.

Disadvantages:

1. It requires a separate algorithm for garbage collection.
2. Garbage collection for disk based file is time consuming.

2.4 ALLOCATION METHODS [April 16]

- The allocation methods define how the files are stored in the disk blocks. Selection of an appropriate allocation method will significantly affect the performance and efficiency of the system.
- Allocation method provides a way in which the disk will be utilized and the files will be accessed. The allocation methods define how the files are stored in the disk blocks.
- The direct-access nature of disks allows us flexibility in the implementation of files. The main problem is how to allocate space to these files so that disk space is utilized effectively and files can be accessed quickly.
- Three major methods of allocating disk space are contiguous, linked and indexed. Each method has its advantages and disadvantages. More commonly, a system uses one method for all files within file system type.
- The main idea behind these methods is to provide:
  - Efficient disk space utilization.
  - Fast access to the file blocks.

2.4.1 Contiguous Allocation

- In contiguous allocation, each file occupies a contiguous set of blocks on the disk.
- In contiguous allocation method, consecutive free blocks are allocated to a file. Directory contains name of the file, starting address and number of blocks.
- If the file is n blocks long and start at location b, then it requires blocks b, b + 1, b + n – 1 as shown in the Fig. 2.13.
- For sequential access of the record, the file system must remember the disk address of the last block read so that the next block is easily read.
- For direct access of block i of a file that starts at block b, we can directly access (b + i) block. Thus, this method supports sequential and direct access.

- It is used in IBM 370.

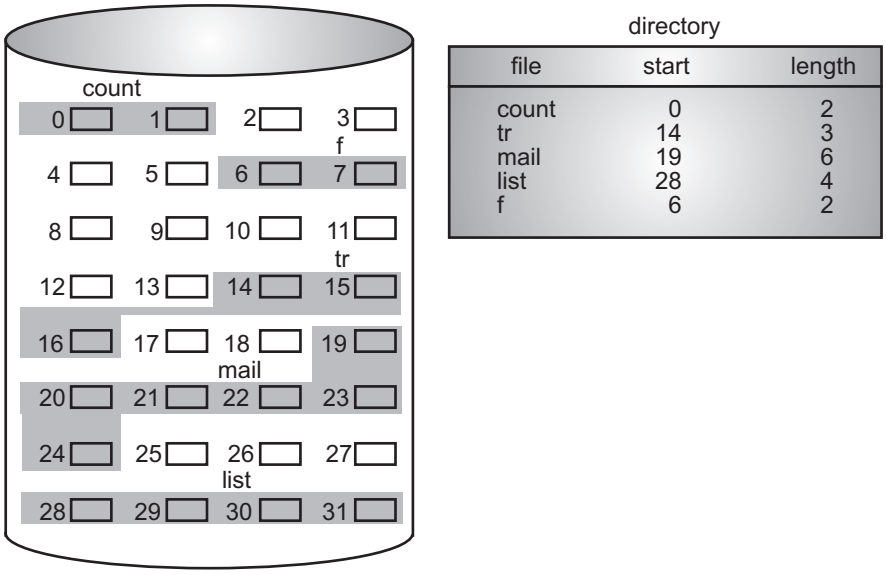


Fig. 2.13: Contiguous Allocation of Disk Space

Advantages:

1. Supports both sequential and direct access.
2. Reading all blocks belonging to each file is very fast.
3. Simple to implement.

Disadvantages:

1. Difficulty in searching space for new files.
2. To allocate a file, one needs to know the size of the file in advance.
3. Suffers from both internal and external fragmentation. This makes it inefficient in terms of memory utilization.

Dynamic Storage Allocation Problem in Contiguous Allocation: **[April 16]**

- Disk space is nothing but a large array of disk blocks. At any instance of time, few blocks are assigned to the file and some blocks are freed by the file.
- At any moment, we may find that allocated as well as free blocks are spread all over the disk. The unallocated (free) blocks are also called holes. There may be a number of consecutive holes.
- Now whenever a user has a request of size n, either of the following methods is used to search a hole big enough to satisfy the requirement of n units from the list of free holes.
  1. **First fit:** In this case, as soon as the first hole (that is big enough) is encountered, searching is stopped and memory is allocated for creating a file.
     - Most simple and less time consuming method to search a hole.
     - Poor memory utilization.
     - It creates internal fragmentation.

2. **Best fit:** In this case, the entire list is searched for and the smallest hole that satisfies the request is selected and is allocated for creating a file.
   - Memory utilization is the best.
   - Internal fragmentation is less.
   - As compared to the first and worst fit, this algorithm is more complex.
   - It is a time consuming process.
3. **Worst fit:** Again the entire list is scanned, and the biggest hole/block satisfying the requirement (i.e. biggest > n) is selected.
   - Time consuming.
   - Memory utilization is very poor.
   - Internal fragmentation is a big problem.

Compaction:

- Compaction is used to improve memory utilization, which is poor because of external fragmentation. Compaction is the process of collecting all free holes into one big hole.

2.4.2 Linked Allocation [April 13]

- Linked allocation is essentially a disk-based version of the linked list. In this allocation, each file is a linked list of disk blocks which need not be contiguous.
- In this method, a linked list of all blocks belonging to the file is maintained. These blocks may be scattered through the disk.
- Directory entry contains the name of the file, address of starting block and the last block. If we want to read a file, we simply follow the pointers.

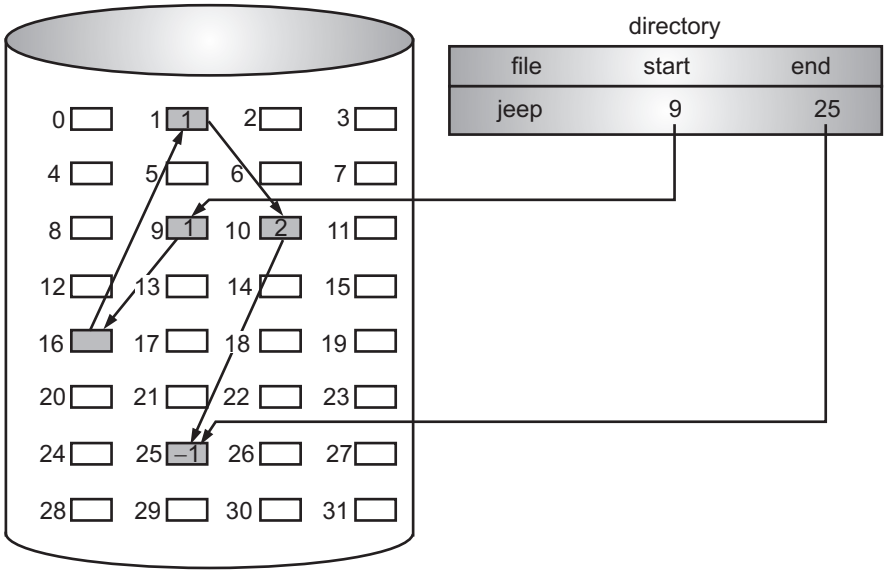


Fig. 2.14: Linked Allocation of Disk Space

- If we want to write to a file, any free block is removed from the free list and written to it. This block is appended at the end of the file with a pointer.
- It is used in DEC TOPS-10, Xerox Alto.
- The file 'jeep' in the Fig. 2.14 shows how the blocks are randomly distributed. The Fig. 2.14 shows file jeep of five blocks 9-16-1-10-25.

Advantages:

1. No external fragmentation because any free block is used to satisfy the request.
2. Files can grow any time as the free blocks are available.
3. No need to know the size of the file in advance.
4. No disk compaction.

Disadvantages:

1. Only sequential access is possible.
2. Memory is required for storing the pointers. Therefore, the file requires slightly more space.
3. Reliability is a big problem. Since disk blocks are linked by pointers, a single damaged pointer can make thousands of disk blocks inaccessible. So we cannot access any portion of the file or entire file.

- A File Allocation Table (FAT) is a variation of Linked allocation. It uses a separate disk area to hold the links. This method doesn't use space in data blocks. Many pointers may remain in memory.
- In FAT a file allocation table is maintained, which gathers all the disk block links. The table has one entry for each disk block and is indexed by block number. A FAT file system is used by MS-DOS.

2.4.3 Indexed Allocation [April 13, 15, Oct. 17]

- One disadvantage with the linked allocation method is that it does not support direct accessing since blocks are scattered all over the disk.
- This problem is solved by indexed allocation by placing all of the pointers together into an index block.
- In indexed allocation, a special block known as the Index block contains the pointers to all the blocks occupied by a file. Each file has its own index block.
- In this method of allocation, an index block is there for each file. It contains addresses of all blocks belonging to that file sequentially e.g. i^{th} entry in the index block of a file will contain the address of i^{th} block belonging to that file.
- A directory entry contains the name of the file and address of the index block. In case of large files, a single index block holds the addresses of all index blocks.
- These indexes may contain addresses of a few more indexed blocks or addresses of actual blocks belonging to the file. This is called as the level of indices.

- In the Fig. 2.15 Block number 19th contains address, 9th, 16th, 1st 10th and 25th blocks. Address of index block 19th is present in the directory entry.
- It is used in DEC VMS, Nachos.

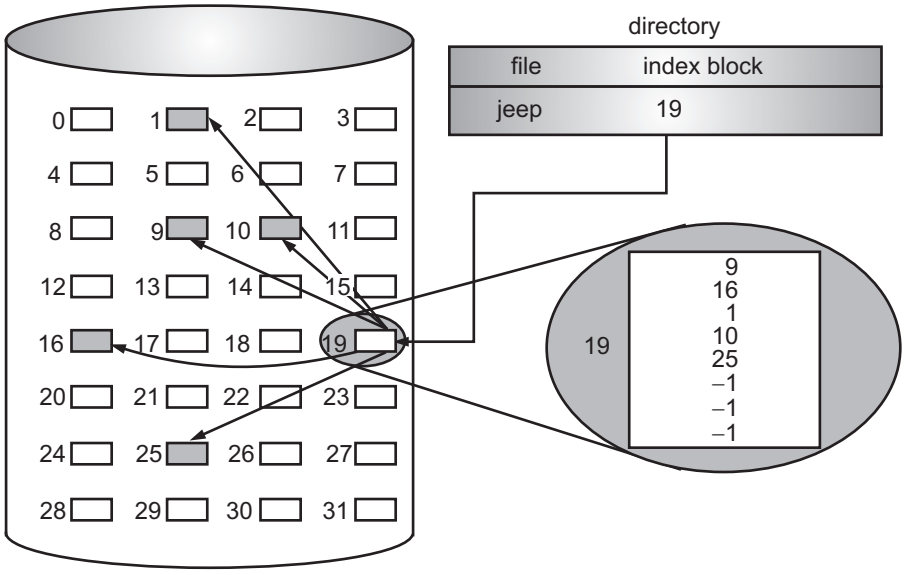


Fig. 2.15: Indexed Allocation of Disk Space

Advantages:

1. Direct access possible.
2. No need for the user to know the size of the file in advance.
3. No external fragmentation.

Disadvantages:

1. Overhead of index blocks is not feasible for very small files.
2. Overhead of the index block is not feasible for very big files also. It is also very difficult to manage levels of indices.
3. File access time is increased because, for any operation of the file, an index block has to be accessed, which increases input / output.
4. Keeping indexes in memory requires space.

- A question is of how big the index block should be, and how it should be implemented. There are several approaches:
  1. **Linked Scheme**: An index block is one disk block, which can be read and written in a single disk operation. The first index block contains some header information, the first N block addresses, and if necessary a pointer to additional linked index blocks.
  2. **Multi-Level Index**: The first index block contains a set of pointers to secondary index blocks, which in turn contain pointers to the actual data blocks.

3. **Combined Scheme:** This is the scheme used in UNIX inodes, in which the first 12 or so data block pointers are stored directly in the inode, and then singly, doubly, and triply indirect pointers provide access to more data blocks as needed.

- Let us summarize briefly three allocation methods.

Factors	Contiguous Allocation Method	Linked Allocation Method	Indexed Allocation Method
Allocation method	Array of disk blocks.	Linked list of disk blocks which are scattered throughout the disk.	All pointers to blocks are placed together in one block known as an index block.
Access method	Sequential and direct.	Only sequential	Sequential and direct.
Directory entry	File name, start location of file and size of file.	File name, start block and end block.	File name and address of index block.
File Size	Should be known in advance and size is fixed.	No need to know in advance and file size can grow at any time.	No need to know in advance and file size can grow at any time.
Fragmentation	Suffers from external and internal fragmen-tation.	No external fragmentation.	No external fragmentation.
Reliable	Yes.	No, the problem of dangling pointers.	Yes.

2.5 FREE SPACE MANAGEMENT [April 16, Oct. 16]

- To store a file on the disk, the operating system needs to know which blocks on the disk are free and which are not. The operating system maintains a free space list to keep the track of free disk space.
- When a file is created, the required amount of the space is searched from the free space list and allocated to the file.
- This space is then removed from the free space list. When a file is deleted, the space which was allocated to a file is freed and added to the free space list.

- Following are some methods which manage free-space list information.

1. Bitmap / Bit Vector: **[Oct. 18]**

- A Bitmap or Bit Vector is a series or collection of bits where each bit corresponds to a disk block. The bit can take two values namely, 0 and 1. The 0 indicates that the block is allocated and 1 indicates a free block.
- This is a method of keeping track of allocated and unallocated blocks. Each block is represented by one bit. If the block is free, the bit is set to 1.
- If the block is allocated, the bit is set to 0.
- For example, consider a disk where blocks 4,5,8,9,10,12,13,15,18,19,25 are free, the free space bitmap or bit vector is: 00011001110110100110000001.
- Fig. 2.16 shows the bitmap representation of a disk.
- The bitmap method for managing the free-space list is easy and simple. For example, if a file requires four free blocks using a contiguous allocation method, free blocks 12, l3, 14 and 15 (the first four free blocks on the disk that are adjacent to each other) may be allocated.
- However, for the same file using linked or indexed allocation, the file system may use free blocks 2, 4, 6, and 8 for allocation to the file.

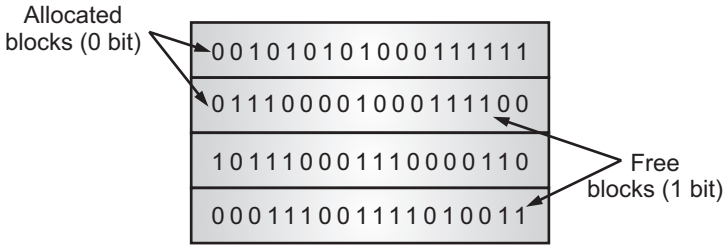


Fig. 2.16: Bit Map

2. Linked List:

- In linked list free space management technique, the free disk blocks are linked together i.e. a free block contains a pointer to the next free block.
- The block number of the very first disk block is stored at a separate location on disk and is also cached in memory.
- The linked list method creates a linked list of all the free blocks on the disk. A pointer to the first free block is kept in a special location on the disk and is cached in the memory.
- This first block contains a pointer to the next free block, which contains a pointer to the next free block and so on.
- Fig. 2.17 shows a linked list of free blocks in which the pointer is at block 2 which is the first free block. Block 2 would contain pointers to block 3 and so on.

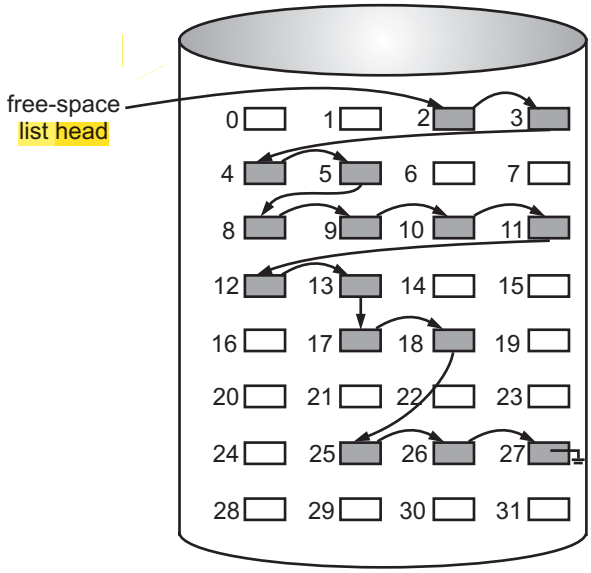


Fig. 2.17: Linked Free-space List on Disk

- However, this scheme is not efficient, since traversing the list is time consuming and requires reading of each block which requires substantial I/O.

3. **Grouping:**
   - To resolve the problem in the linked free space list, the addresses of n free blocks in the first free block.
   - The first n - 1 of these blocks is actually free.
   - The last block contains the disk addresses of other n free blocks and so on.
   - In this method, the addresses of a large number of free blocks can be found quickly.
4. **Counting:**
   - When many contiguous free blocks are present, rather than keeping a list of n free disk addresses, we can keep the address of the first free block and count of the next n free contiguous blocks.
   - Each entry in the free space list then consists of a disk address and a count.
5. **Space Map:**
   - A space map is a log of the free/allocated space for a disk region. The ZFS a file system recently developed by Sun Microsystems.
   - ZFS (previously: Zettabyte File System) tracks free space with space maps. Sun's ZFS file system was designed for HUGE numbers and sizes of files, directories, and even file systems.

- The resulting data structures could be VERY inefficient if not implemented carefully. For example, freeing up a 1 GB file on a 1 TB file system could involve updating thousands of blocks of free list bitmaps if the file was spread across the disk.
- ZFS divides the space on each virtual device into a few hundred regions called metaslabs. Each metaslab has an associated space map, which describes that metaslab's free space.
- ZFS uses a combination of techniques, starting with dividing the disk up into (hundreds of) metaslabs of a manageable size, each having their own space map.
- Free blocks are managed using the counting technique, but rather than write the information to a table, it is recorded in a log-structured transaction record. Adjacent free blocks are also coalesced into a larger single free block.
- An in-memory space map is constructed using a balanced tree data structure, constructed from the log data.
- The combination of the in-memory tree and the on-disk log provide for very fast and efficient management of these very large files and free blocks.
- The design of space maps makes the free space information of the file system extremely compact allowing it to be stored in main memory at all times.

PRACTICE QUESTIONS

Q. I Multiple Choice Questions:

1. Which is a named collection of related information that is recorded on secondary storage?
   (a) file (b) directory
   (c) file system (d) None of the mentioned
2. Collection of files is a file is called as,
   (a) file (b) directory
   (c) file system (d) None of the mentioned
3. Which specifies the characteristics of the files such as type, size, location etc.?
   (a) File operations (b) File contents
   (c) File attributes (d) All of the mentioned
4. Which specifies the task or actions that can be performed on a file?
   (a) File operations (b) File contents
   (c) File attributes (d) All of the mentioned

5. By using the specific system call, we can
   (a) open and close the file (b) read and write the file
   (c) create and delete file (d) All of the mentioned
6. Which specifies the logical method of file storage in a computer system?
   (a) File operations (b) File systems
   (c) File attributes (d) All of the mentioned
7. Which is used to identify a storage location in the file system?
   (a) filename (b) filetype
   (c) filelocation (d) All of the mentioned
8. Repositioning in a file is also known as file,
   (a) reposition (b) relocation
   (c) seek (d) float
9. In which file access method files can be accessed in random for read and write operations.
   (a) sequential (b) random or relative
   (c) Index sequential (d) All of the mentioned
10. A file has a certain defined structure according to its, type
   (a) type (b) attribute
   (c) size (d) name
11. Which organizes and provides information about all the files in the system.
   (a) file (b) file attribute
   (c) file operation (d) directory
12. In which file allocation method a single continuous set of blocks is allocated to a file at the time of file creation.
   (a) Indexed (b) Linked
   (c) Continuous (d) float
13. Which file access method constructs an index for the file?
   (a) Index sequential (b) Random or relative
   (c) Sequential (d) All of the mentioned
14. In which file access allocation, each file is considered as the linked list of disk blocks.
   (a) Indexed (b) Linked
   (c) Continuous (d) float

15. Which a graph with no cycles and allows directories to share subdirectories and files.
    (a) Acyclic graph (b) wait-for graph
    (c) Both (a) and (b) (d) None of the mentioned
16. Which is series or collection of bits where each bit corresponds to a disk block?
    (a) linked list (b) counting
    (c) grouping (d) Bit vector

Answers

1. (a)	2. (b)	3. (c)	4. (a)	5. (d)	6. (b)	7. (a)	8. (c)	9. (b)	10. (a)
11. (d)	12. (c)	13. (a)	14. (b)	15. (a)	16. (d)				

Q. II Fill in the Blanks:

1. A _____ is a collection of related information that is recorded on secondary storage like disks.
2. Collection of files is a file _____ .
3. Files are typically organized in a file _____, which tracks file locations on the disk and enables user access.
4. File _____ describe all the information regarding particular file.
5. _____ operation a file makes the file contents available to the program.
6. File access _____ specifies the access permissions related to a file such as read and write.
7. File _____ in operating system is used to manage files of computer system includes the process of creating, modifying and deleting the files.
8. To _____ from a file, the system call should specify the name and location of the file.
9. A file _____ should be according to a required format that the operating system can understand.
10. File _____ refers to the ability of the operating system to distinguish different types of file such as text files, source files, object files and binary files etc.
11. File _____ methods refers to the manner in which the records of a file may be accessed.
12. To _____ into a file, the system call should specify the name of the file and the contents that need to be written.

13. A _____ access is that in which the records are accessed in pre-defined sequence, i.e., the information in the file is processed in order, one record after the other.
14. In _____ file allocation method, an index block is for each file which contains addresses of all blocks belonging to that file sequentially.
15. A _____ contains the number of bits where each bit represents each block.
16. In _____ -level directory the entire system will contain only one directory which is supposed to mention all the files present in the file system.
17. _____ is free space management technique which stores the address of the first free disk block and a number n of free contiguous disk blocks that follow the first block.
18. In _____ -structured directory each user has its own directory and it cannot enter in the other user's directory.
19. In linked list file allocation, each file is considered as the _____ of disk blocks.
20. ZFS file system was designed for _____ numbers and sizes of files, directories, and even file systems.

Answers

1. file	2. directory	3. system	4. attributes
5. Open	6. permission	7. management	8. read
9. structure	10. type	11. access	12. write
13. sequential	14. indexed	15. bit map vector	16. single
17. Counting	18. tree	19. linked list	20. HUGE

Q. III State True or False:

1. Files on a computer can be created, moved, modified, grown, shrunk (truncated), and deleted.
2. Files are typically organized in a file system, which tracks file locations on the disk and enables user access.
3. A directory is collection of related information stored in the memory of computer system.
4. File access is a process that determines the way that files are accessed and read into memory.
5. A file structure needs to be predefined format in such a way that an operating system understands it.

6. Linked allocation or non-contiguous allocation each block contains a pointer to the next block in the chain (linked list).
7. In indexed sequential file access, an index is built for every file, with a direct pointer to different memory blocks. In this method, the Index is searched sequentially, and its pointer can access the file directly.
8. In general graph directory structure, cycles are allowed within a directory structure where multiple directories can be derived from more than one parent directory.
9. A directory is a container that is used to collection of files.
10. The grouping free space management, stores the address of the free blocks in the first free block.
11. In linked list free space management technique, the free disk blocks are linked together i.e. a free block contains a pointer to the next free block.
12. An acyclic graph is a graph with no cycle and allows to share subdirectories and files.
13. In two-level directory one directory for each user is maintained.
14. Random access file organization provides, accessing the records directly and each record has its own address on the file with by the help of which it can be directly accessed for reading or writing.
15. A file is a collection of related information that is recorded on secondary storage like disks.
16. A tree structure directory structure has a root directory, and every file in the system have a unique path.
17. A bitmap or bit vector the bit can take two values: 0 and 1. The 0 indicates that the block is allocated and 1 indicates a free block.
18. In acyclic-graph directory has is a natural generalization of the tree-structured directory.
19. File system is the part of the operating system which is responsible for file management. It provides a mechanism to store the data and access to the file contents including data and programs.
20. In ZFS, we can create file-system hierarchies.

Answers

1. (T)	2. (T)	3. (T)	4. (F)	5. (T)	6. (T)	7. (T)	8. (T)	9. (T)	10. (T)
11. (T)	12. (T)	13. (F)	14. (T)	15. (T)	16. (T)	17. (T)	18. (T)	19. (T)	20. (T)

Q. IV Answer the following Questions:

(A) Short Answer Questions:

1. Define file.
2. Define is file attribute.
3. What is meant by file structure?
4. Define file operation.
5. What is acyclic graph?
6. Define free space management.
7. Give the purpose of file system?
8. Define directory.
9. List types of directories.
10. State true or false, “an acyclic graph namely a graph with cycles allows directories to share subdirectories and files.
11. Define bit vector.
12. What is file allocation?
13. Justify: “newly created directory will have two entries automatically in it”?
14. List basic operations on a file.
15. “The single-level directory has only one root directory and no user is allowed to create subdirectories inside the root directory.” Justify this statement.
16. Which methods are used for file access?
17. Define file system management.
18. Give purpose of linked list in free space management.

(B) Long Answer Questions:

1. What is file? How to create, open, read and write it?
2. What is directory? What are its different types? Explain two of them in detail.
3. What is free space management technique? Explain linked list method with example.
4. What are sequential and direct access methods of file? Explain with diagrammatically. Also compare them.
5. What is meant by file allocation method? List its types. And also compare them.
6. With the help of diagram describe acyclic-graph directory in detail.
7. With the help of example describe bit vector.

8. Give the diagrammatic representation of single-level directory. Also list out the disadvantages of single-level directory structure.
9. What is counting and grouping? How they can e used in free space management?
10. What is file system? Describe in detail with diagram.
11. What is directory? What are its types? Explain two of them.
12. What are the operations performed on files?
13. With the help of diagram explain contiguous allocation. Also state its advantages and disadvantages.
14. Write a short note on: General graph directory.
15. Explain two-level directory structure with diagram, advantages and disadvantages.
16. What is file management system? Explain its function in detail.
17. Describe tree structure directory structure diagrammatically.
18. What space map? Explain in detail.

UNIVERSITY QUESTIONS AND ANSWERS

April 2016

1. Discuss the various techniques of free space management in file system **[5 M]**

Ans. Refer to Section 1.5.

2. Explain tree structured directories along with advantages and disadvantages. **[5 M]**

Ans. Refer to Section 2.3.2.3.

3. What are various dynamic allocation memory management methods? **[1 M]**

Ans. Refer to Pages 2.22 and 2.23.

4. Give any two disk allocation methods.

Ans. Refer to Section 2.4.

October 2016

1. List any four file attributes. **[1 M]**

Ans. Refer to Section 2.1.3.

2. Explain different methods for handling free-space list in file system. **[5 M]**

Ans. Refer to Section 2.5.

3. Explain file operations in detail. **[4 M]**

Ans. Refer to Section 2.1.4.

April 2017

1. List any four file attributes. **[1 M]**

Ans. Refer to Section 2.1.3.

2. Explain tree structured directories along with advantages and disadvantages. **[4 M]**

Ans. Refer to Section 2.3.2.3.

October 2017

1. Define acyclic graph directory. **[1 M]**

Ans. Refer to Section 2.3.2.4.

2. Explain different file access methods. **[4 M]**

Ans. Refer to Section 2.2.

April 2018

1. List any four file attributes. **[1 M]**

Ans. Refer to Section 2.1.3.

2. "Newly created directory will have two entries automatically in it". Comment. **[1 M]**

Ans. Refer to Section 2.3.

3. Explain tree-structured directories along with its advantages and disadvantages. **[5 M]**

Ans. Refer to Section 2.3.2.3.

October 2018

1. What is bit vector? **[1 M]**

Ans. Refer to Section 2.5, Point (1).

April 2019

1. Write file access methods. **[1 M]**

Ans. Refer to Section 2.2.

■■■

CHAPTER 3

Disk Scheduling

Objectives...

- To understand Concept of Disk
- To learn Disk Scheduling
- To study different Disk Scheduling Algorithms

3.0 INTRODUCTION

- File systems must be accessed in an efficient manner, especially with hard drives, which are the slowest part of a computer.
- As a computer deals with multiple processes over a period of time, a list of requests to access the disk builds up. For efficiency purposes, all requests (from all processes) are aggregated together.
- The technique that the operating system uses to determine which requests to satisfy first is called disk scheduling
- Disk scheduling is a technique used by the operating system to schedule multiple requests for accessing the disk. The algorithms used for disk scheduling are called as disk scheduling algorithms.
- In most system there are many processes that are running simultaneously. Often, many processes request I/O operations from / to the hard drive.
- The algorithm used to select which request is going to be satisfied first is called a disk scheduling algorithm.
- Various disk scheduling algorithms are used in accessing data and transferring data. These are disk scheduling algorithms are FCFS, SSTF, SCAN and LOOK.

3.1 OVERVIEW

- As we know, a process needs two types of time, CPU time and IO time. For I/O, it requests the operating system to access the disk.
- However, the operating system must be fair enough to satisfy each request and at the same time, the operating system must maintain the efficiency and speed of process execution.

- The technique that the operating system uses to determine the request which is to be satisfied next is called disk scheduling.
- The main purpose of the disk scheduling algorithm is to select a disk request from the queue of IO requests and decide the schedule when this request will be processed.
- Hard disk is magnetic storage medium for a computer. Hard disks are flat circular plates made of aluminum or glass and coated with a magnetic material.
- A floppy disk is a flexible disk with a magnetic coating on it. It is packaged inside a protective plastic envelope.
- The floppy disks are one of the oldest types of portable storage devices that could store up to 1.44 MB of data but now they are not used due to very less memory storage.

3.1.1 Disk Structure

- In modern computers, most of the secondary storage is in the form of magnetic disks. Hence, knowing the structure of a magnetic disk is necessary to understand how the data in the disk is accessed by the computer.

Structure of a Magnetic Disk:

- A magnetic disk contains several platters. Each platter is divided into circular shaped tracks. The length of the tracks near the centre is less than the length of the tracks farther from the centre.
- Each track is further divided into sectors, as shown in the figure. There may be thousands of concentric cylinders in a disk drive, and each track may contain hundreds of sectors. The storage capacity of common disk drives is measured in gigabytes.
- Tracks of the same distance from the centre form a cylinder. A read-write head is used to read data from a sector of the magnetic disk.

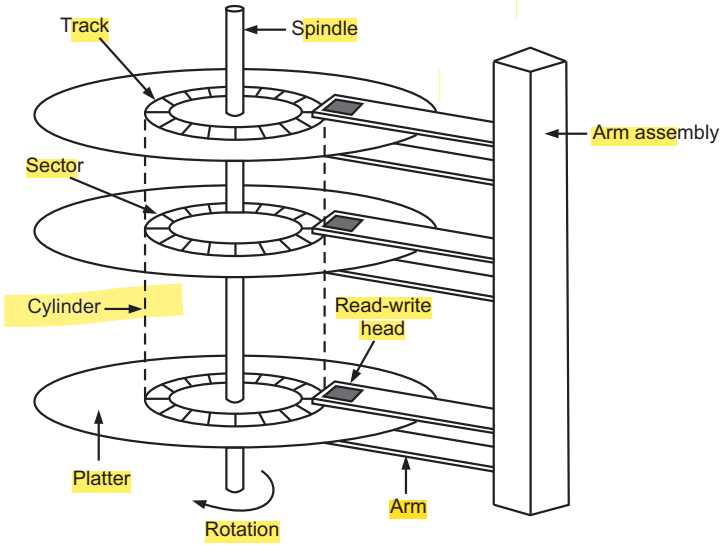


Fig. 3.1: Disk Structure

- The speed of the disk is measured as two parts:
  - **Transfer Rate** is the rate at which the data moves from disk to the computer.
  - **Random Access Time** is the sum of the seek time and rotational latency.
  - The **seek time** is the time for the disk arm to move the head to the required cylinder containing the desired track.
  - The **rotational latency** is the additional time for the disk to rotate the desired sector to the disk head.
  - The **disk bandwidth** is the total number of bytes transferred, divided by the total time between the first request for service and the completion of the last transfer.
  - A disk drive is attached to a computer by a set of wires called an I/O bus. Several kinds of buses are available, including Advanced Technology Attachment (ATA), Serial ATA (SATA), eSATA, Universal Serial Bus (USB), and Fiber Channel (FC).
  - The data transfers on a bus are carried out by special electronic processors called controllers. The **host controller** is the controller at the computer end of the bus.
  - A **disk controller** is built into each disk drive. To perform a disk I/O operation, the computer places a command into the host controller, typically using memory-mapped I/O ports.
- Even though the disk is arranged as sectors and tracks physically, the data is logically arranged and addressed as an array of blocks of fixed size. The size of a block can be 512 or 1024 bytes.
- The logical block is the smallest unit of transfer. The 1-dimensional array of logical blocks is mapped into the sectors of the disk sequentially. In this way, each sector in the disk will have a logical address. Sector 0 is the first sector of the first track on the outermost cylinder.
- Mapping proceeds in order through that track, then the rest of the tracks in that cylinder, and then through the rest of the cylinders from outermost to innermost.

3.2 DISK SCHEDULING

- On a typical multiprogramming system, there will usually be multiple disk access requests at any point of time. So those requests must be scheduled to achieve good efficiency.
- The OS is responsible for minimizing access time and maximizing bandwidth for disks. Access time has two major components – Seek time and Rotational latency.
- Whenever a process needs I/O to or from the disk, it issues a system call to the operating system.
- The request specifies several pieces of information like whether operation is input or output, disk address, memory address for the transfer, number of sectors to be transferred etc.

- Requests are serviced immediately if the disk is not busy, if disk is busy then requests are queued and disk scheduling is used to choose the next request to service.

3.2.1 Disk Performance Parameters

- When a disk drive is operating, the disk is rotating at constant speed. To read or write, the head must be positioned at the desired track and at the beginning of the desired sector on the track.
- Track selection involves moving the head in a movable-head system or electronically selecting one head on a fixed head system.
- On a movable-head system, the time taken to position the head at the track is known as seek time.
- Once the track is selected, the disk controller waits until the appropriate sector rotates to line up with the head.
- The time it takes to reach the beginning of the desired sector is known as rotational delay or rotational latency.
- The sum of the seek time, (for movable head system) and the rotational delay is termed as access time of the disk, the time it takes to get into appropriate position (track and sector) to read or write.
- Once the head is in position, the read or write operation is then performed as the sector moves under the head, and the data transfer takes place.

Performance parameters are as follows:

1. **Seek Time:**

- Seek time is the time required to move the disk arm to the required track. The seek time is approximated as,

$$T_s = m \times n + s$$

where, T_s = estimated seek time
n = number of tracks traversed
m = constant that depends on the disk drive
s = startup time

2. **Rotational Delay:**

- Disk drive generally rotates at 3600 rpm, i.e. to make one revolution it takes around 16.7 ms. Thus on the average, the rotational delay will be 8.3 ms.

3. **Transfer Time:**

- The transfer time to or from the disk depends on the rotational speed of the disk and it is estimated as,

$$T = \frac{b}{rN}$$

where, T = Transfer time

b = Number of bytes to be transferred

N = Numbers of bytes on a track

r = Rotational speed, in revolution per second.

4. Access Time:

Access Time = Seek Time + Rotational Delay + Transfer Rate

Thus, the total average access time can be expressed as,

$$T_a = T_s + \frac{1}{2r} + \frac{b}{rN}$$

where T_s is the average seek time.

5. Disk Response Time:

- Response Time is the average of time spent by a request waiting to perform its I/O operation. Average Response time is the response time of all requests.
- Variance Response Time is a measure of how individual requests are serviced with respect to average response time.

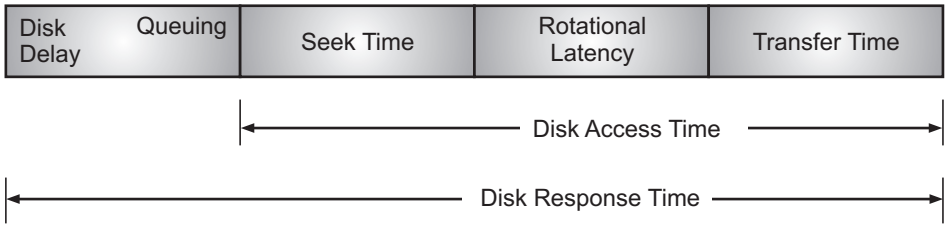


Fig. 3.2: Disk Performance Parameter

- Disks are a potential bottleneck for system performances and storage system reliability.
- The disk access time is relatively higher than the time required to access data from main memory and perform CPU operation.
- Also the disk drive contains some mechanical parts and it involves mechanical movement, so the failure rate is also high.
- The disk performance has been improving continuously; microprocessor performance has improved much more rapidly.
- A disk array is an arrangement of several disks, organized to increase performance and improve reliability of the resulting storage system. Performance is increased through data striping. Reliability is improved through redundancy.
- Disk arrays that implement a combination of data striping and redundancy are called Redundant Arrays of Independent Disks (RAID).

3.2.2 Scheduling Algorithms

- Disk scheduling is done by operating systems to schedule I/O requests arriving for disk.
- Disk scheduling is important because:
  1. Multiple I/O requests may arrive by different processes and only one I/O request can be served at a time by disk controller. Thus other I/O requests need to wait in the waiting queue and need to be scheduled.
  2. Two or more requests may be far from each other so can result in greater disk arm movement.
  3. Hard drives are one of the slowest parts of a computer system and thus need to be accessed in an efficient manner.
- Disk scheduling is similar to process scheduling. The disk scheduling algorithm that gives minimum average seek time, minimum rotational latency and minimum variance response time is better.
- Some of the disk scheduling algorithms are described in following sections.

3.2.2.1 FCFS

- The simplest form of disk scheduling is the First-Come, First-Served (FCFS) algorithm. In FCFS, the requests are addressed in the order they arrive in the disk queue.
- The FCFS algorithm is intrinsically fair to all processes, but it generally does not provide the fastest service.
- For example, consider, for example, a disk queue with requests for I/O to block on cylinders 95, 180, 34, 119, 11, 123, 62, 64 with the Read-write head initially at the cylinder 50 and the tail cylinder being at 199.
- All incoming requests are placed at the end of the queue. If the disk head is initially at cylinder 50, it will first move from 50 to 95, 95 then to 180, 34, 119, 11, 123, 62 and finally to 64.

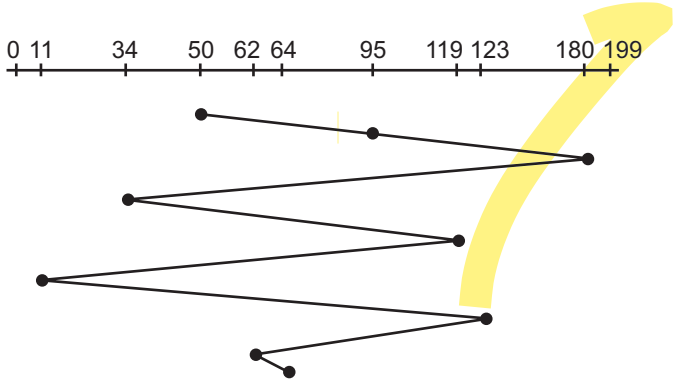


Fig. 3.3: FCFS Scheduling

- To determine the number of head movements you would simply find the number of tracks it took to move from one request to the next.

  Total head Movement = (95-50)+(180-95)+(180-34)+(119-34)+(119-11)+
  (123-11)+(123-62)+(64-62)
  = 45+85+146+85+108+112+61+2
  = 644

- The disadvantage of this algorithm is noted by the oscillation from track 50 to track 180 and then back to track 11 to 123 then to 64. As we will soon see, this is the worst algorithm that one can use.

Advantages:

1. Every request gets a fair chance.
2. No indefinite postponement.

Disadvantages:

1. Algorithm does not try to optimize seek time.
2. It may not provide the best possible service.

3.2.2.2 SSTF

- The SSTF (Shortest Seek Time First) algorithm selects the request with the minimum seek time from the current head position.
- In other words, SSTF chooses the pending request closest to the current head position. SSTF scheduling is essentially a form of Shortest-Job-First (SJF) scheduling; and like SJF scheduling.
- So, the seek time of every request is calculated in advance in the queue and then they are scheduled according to their calculated seek time. As a result, the request near the disk arm will get executed first.
- For example, consider a disk queue with requests for I/O to block on cylinders 95, 180, 34, 119, 11, 123, 62, 64 with the Read-write head initially at the cylinder 50 and the tail cylinder being at 199.
- In this case the request is serviced according to the next shortest distance. Starting at 50, the next shortest distance would be 62 instead of 34 since it is only 12 tracks away from 62 and 16 tracks away from 34. The process would continue until all the processes are taken care of.
- For example, the next case would be to move from 62 to 64 instead of 34 since there are only 2 tracks between them and not 18 if it were to go the other way.

  Total Head Movement = (64 – 50) + (64 – 11) + (180 – 11) = 14 + 53 + 169 = 236 tracks.

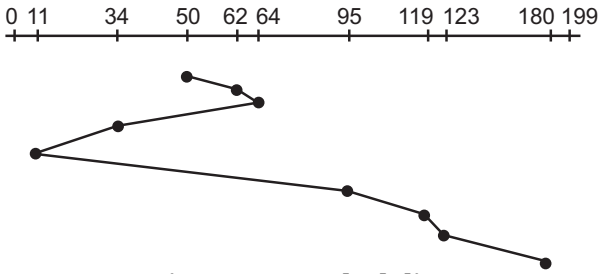


Fig. 3.4: SSTF Scheduling

Advantages:

1. It decreases average response Time.
2. It increases throughput.

Disadvantages:

1. Overhead to calculate seek time in advance.
2. It can cause starvation for a request if it has higher seek time as compared to incoming requests.
3. High variance of response time as SSTF favors only some requests.

3.2.2.3 SCAN

- In the SCAN algorithm, the disk arm moves into a particular direction and services the requests coming in its path and after reaching the end of the disk, it reverses its direction and again services the request arriving in its path.
- So, this algorithm works like an elevator and hence is also known as elevator algorithm. It moves the Read/Write (R/W) head back-and-forth to the innermost and outermost track.
- As it scans the tracks from end to end, it processes all the requests found in the direction it is headed.
- This will ensure that all track requests, whether in the outermost, middle or innermost location, will be traversed by the access arm thereby finding all the requests.
- For example, consider a disk queue with requests for I/O to block on cylinders 95, 180, 34, 119, 11, 123, 62, 64 with the Read-write head initially at the cylinder 50 and the tail cylinder being at 199. Assume direction towards 0.

  Total Head Movement = (50-0) + (180-0) = 50 + 180 = 230.

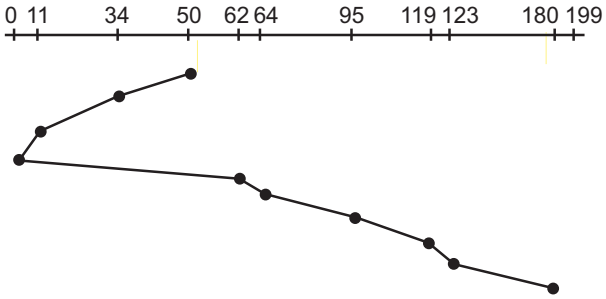


Fig. 3.5: SCAN Scheduling

Advantages:

1. It gives high throughput.
2. It provides low variance of response time.
3. It provides average response time.

Disadvantages:

1. Long waiting time for requests for locations just visited by disk arm.

C-SCAN:

- In the SCAN algorithm, the disk arm again scans the path that has been scanned, after reversing its direction.
- So, it may be possible that too many requests are waiting at the other end or there may be zero or few requests pending at the scanned area. These situations are avoided in the C-SCAN algorithm.
- In C-SCAN, the head moves from one end of the disk to the other servicing requests as it goes.
- When it reaches the other end, however, it immediately returns to the beginning of the disk, without servicing any requests on the return trip.
- It treats the cylinders as a circular list that wraps around from the last cylinder to the first one.
- For example, consider a disk queue with requests for I/O to block on cylinders 95, 180, 34, 119, 11, 123, 62, 64 with the Read-write head initially at the cylinder 50 and the tail cylinder being at 199. Assume direction towards 0.

Total Head Movement = (50 – 0) + (199 – 62) = 50 + 137 = 187.

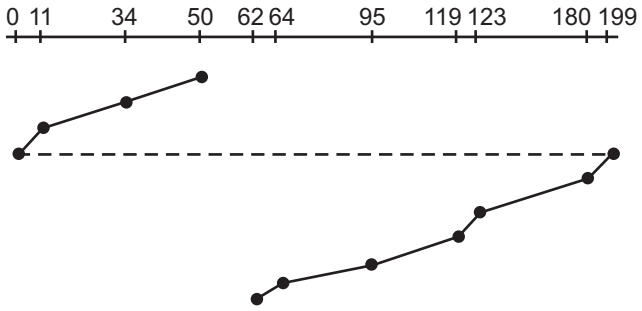


Fig. 3.6: C-SCAN Scheduling

Advantage: Provides more uniform wait time compared to SCAN.

3.2.2.4 LOOK

- It is similar to the SCAN disk scheduling algorithm except the difference is that the disk arm in spite of going to the end of the disk goes only to the last request to be serviced in front of the head and then reverses its direction from there only.
- Thus, it prevents the extra delay which occurred due to unnecessary traversal to the end of the disk.

- The disk arm starts at the first I/O request on the disk, and moves toward the last I/O request on the other end, servicing requests until it gets to the other extreme I/O request on the disk, where the head movement is reversed and servicing continues.
- It moves in both directions until both last I/O requests; more inclined to serve the middle cylinder requests.
- For example, consider a disk queue with requests for I/O to block on cylinders 95, 180, 34, 119, 11, 123, 62, 64 with the Read-write head initially at the cylinder 50 and the tail cylinder being at 199. Assume direction towards 0.

  Total head Movement = (50 – 11) + (180 – 11)

  = 39 + 169 = 208

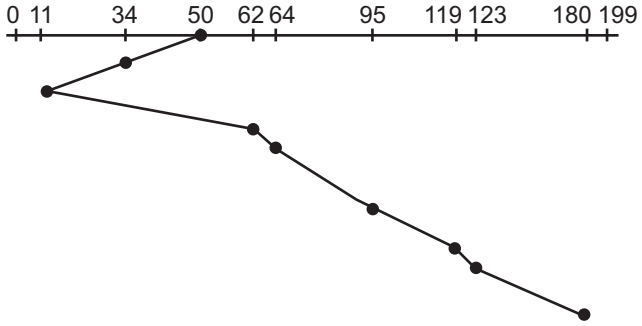


Fig. 3.7: LOOK Scheduling

Advantages:

1. Better than the SCAN algorithm, in terms of head movement.

Disadvantages:

1. It doesn't provide uniform waiting time.

C-LOOK:

- C-LOOK is similar to the C-SCAN disk scheduling algorithm.
- In C-LOOK, the disk arm in spite of going to the end goes only to the last request to be serviced in front of the head and then from there goes to the other end's last request.
- Thus, it also prevents the extra delay which occurred due to necessary traversal to the end of the disk.
- In general, Circular versions are fairer but pay with a larger total seek time. Scan versions have a larger total seek time than the corresponding Look versions.
- For example, consider a disk queue with requests for I/O to block on cylinders 95, 180, 34, 119, 11, 123, 62, 64 with the Read-write head initially at the cylinder 50 and the tail cylinder being at 199. Assume direction towards 0.

  Total Head Movement = (50 – 11) + (180 – 62) = 39 + 118 = 157 tracks

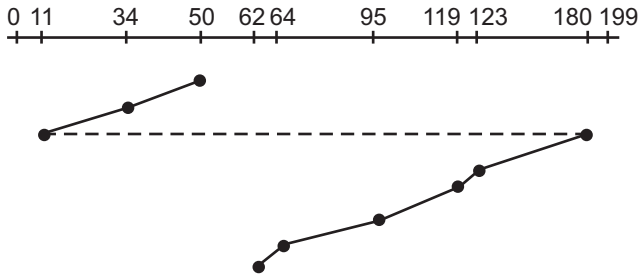


Fig. 3.8: C-Look Scheduling

- When selecting a Disk Scheduling algorithm, performance depends on the number and types of requests. SSTF is common and has a natural appeal. SCAN gives better performance than FCFS and SSTF.
- The disk-scheduling algorithm should be written as a separate module of the operating system, allowing it to be replaced with a different algorithm if necessary. Either SCAN or C-LOOK is a reasonable choice for the default algorithm.

Solved Problems

Problem 1: Consider Work Queue: 23, 89, 132, 42, 187 and schedule by algorithms:

1. FCFS (FIFO)
2. SSTF
3. SCAN
4. LOOK
5. C-SCAN
6. C-LOOK

There are 200 cylinders numbered from 0-199. The disk head starts at number 100.

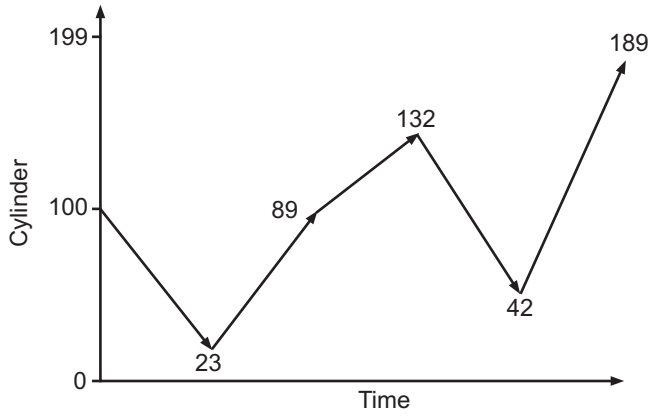


Fig. 3.9

1. FCFS (FIFO):

Total time is estimated by total arm motion

$|100 - 23| + |23 - 89| + |89 - 132| + |23 - 132| + |132 - 42| + |42 - 187|$

$= 77 + 66 + 43 + 109 + 90 + 145$

$= 530$

2. SSTF:

$|100 - 89| + |89 - 132| + |132 - 187| + |187 - 42| + |42 - 23|$

$= 11 + 43 + 55 + 145 + 19 = 273$

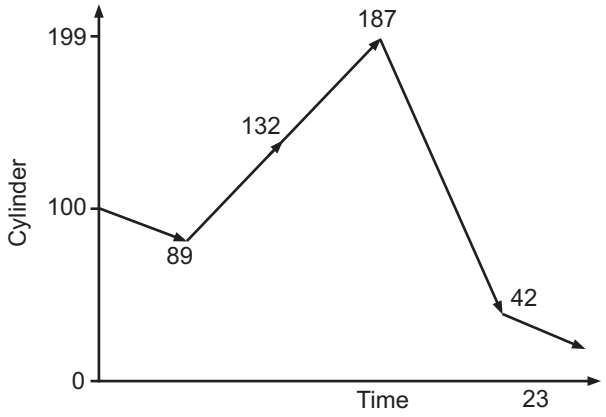


Fig. 3.10

3. SCAN:

Assume we are going inwards
(i.e., towards 0).

$|100 - 89| + |89 - 42| + |42 - 23| + |23 - 0| + |0 - 132| + |132 - 187|$

$= 11 + 47 + 19 + 23 + 132 + 55$

$= 287$

Refer Fig. 3.11.

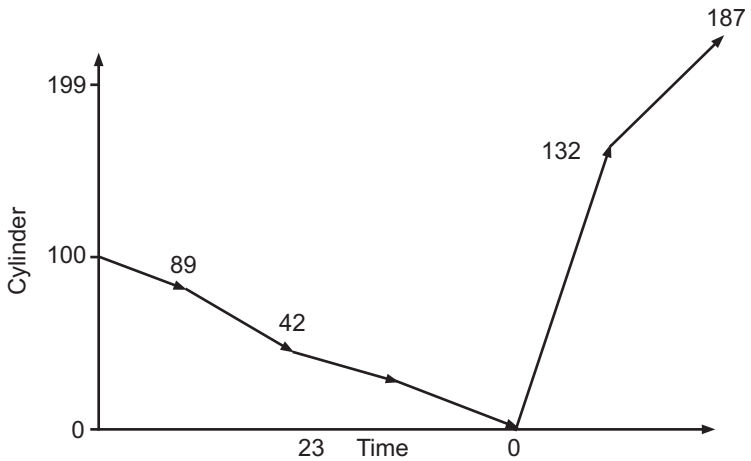


Fig. 3.11

4. LOOK:

$|100 - 89| + |89 - 42| + |42 - 23| + |23 - 132| + |132 - 187|$

$= 11 + 47 + 19 + 109 + 55 = 241$

Reduce variance compared to SCAN.

Refer Fig. 3.12.

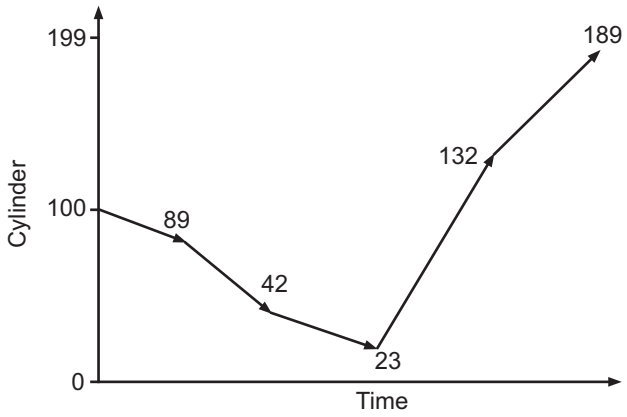


Fig. 3.12

5. C-SCAN:

$|100 - 89| + |89 - 42| + |42 - 23| + |23 - 0| + |0 - 199| + |199 - 187| + |187 - 132|$

$= 11 + 47 + 19 + 23 + 199 + 12 + 55 = 366$

Refer Fig. 3.13.

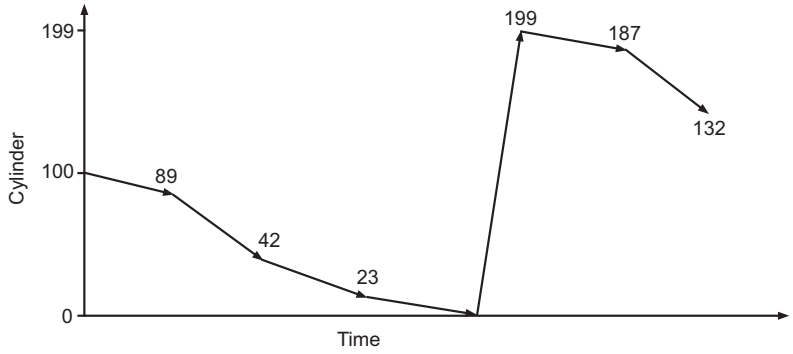


Fig. 3.13

6. C-LOOK:

$|100 - 89| + |89 - 42| + |42 - 23| + |23 - 187| + |187 - 132|$

$= 11 + 47 + 19 + 164 + 55 = 296$

Refer Fig. 3.14.

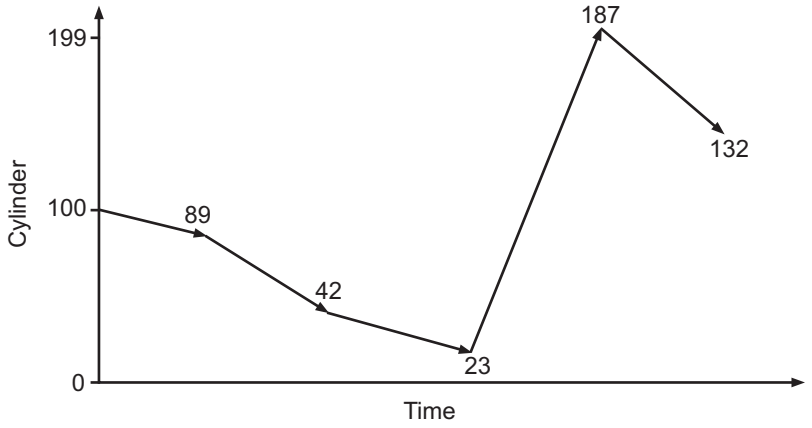


Fig. 3.14

3.3 DISK MANAGEMENT

- The operating system is responsible for several other aspects of disk management, too such as disk initialization, booting from disk and bad-block recovery.
- A hard drive needs to be initialized by the operating system before it can be used. Initializing disk is a must-be process after we install a new hard drive on the computer.
- Initializing disk is a process of building or rebuilding MBR (Master Boot Record) the first sector of a hard drive. The process of initializing the hard drive involves setting the partition style.
- The partition style defines how the hard drive will store the partition information so that the operating system knows which sectors belong to each partition, and which partition is bootable.

Disk Formatting:

- A new magnetic disk is a blank slate meaning it is just a platter of a magnetic recording material.
- Before a disk can store data, it must be divided into sectors that the disk controller can read and write and this process is called low-level formatting or physical formatting.
- Physical formatting fills the disk with a special data structure for each sector. The data structure for a sector typically consists of a header, a data area (usually 512 bytes in size) and a trailer.
- The header and trailer contain information used by the disk controller such as a sector number and an ECC (Error Correcting Code).
- When the disk controller writes a sector of data during normal I/O, the ECC is updated with a value calculated from all the bytes in the data area. When the sector is read, the ECC is recalculated and compared with the stored value.

- If the stored and calculated numbers are different, this mismatch indicates that the data area of the sector has become corrupted and that the disk sector may be bad.
- The ECC contains enough information, if only a few bits of data have been corrupted, to enable the disk controller to identify which bits have changed and calculate what their correct values should be.
- ECC then reports recoverable soft errors. The disk controller automatically does the ECC processing whenever a sector is read or written.
- For a numbers of hard disks are low-level/physical formatted at the factory as a part of the manufacturing process.
- This formatting enables the manufacturer to test the disk and to initialize the mapping from logical block numbers to defect-free sectors on the disk.
- For number of hard disks, when the disk controller is instructed to low-level-format the disk, it can also be told how many bytes of data space to leave between the header and trailer of all sectors. It is usually possible to choose among a few sizes, such as 256, 512 and 1,024 bytes.
- Formatting a disk with a larger sector size means that fewer sectors can fit on each track; but it also means that fewer headers and trailers are written on each track and more space is available for user data. Some operating systems can handle only a sector size of 512 bytes.
- Before it can use a disk to hold files, the operating system still needs to record its own data structures on the disk. It does so in following two steps:

  Step 1: To partition the disk into one or more groups of cylinders. The operating system can treat each partition as though it were a separate disk. For example, one partition can hold a copy of the operating system's executable code, while another holds user files.

  Step 2: The logical formatting or creation of a file system. In this step, the operating system stores the initial file-system data structures onto the disk. These data structures may include maps of free and allocated space (a FAT or inodes) and an initial empty directory.
- To increase efficiency, a number of file systems group blocks together into larger chunks, frequently called clusters.
- Disk I/O is done via blocks, but file system I/O is done via clusters; effectively assuring that I/O has more sequential-access and fewer random-access characteristics.
- Some operating systems give special programs the ability to use a disk partition as a large sequential array of logical blocks, without any file-system data structures. This array is sometimes called the raw disk, and I/O to this array is termed raw I/O.
- For example, some database systems prefer raw I/O because it enables them to control the exact disk location where each database record is stored.
- Raw I/O bypasses all the file-system services, such as the buffer cache, file locking, pre-fetching, space allocation, file names, and directories.

- We can make number of applications more efficient by allowing them to implement their own special-purpose storage services on a raw partition, but most applications perform better when they use the regular file-system services.

Boot Block:

- For a computer to start running - for example, when it is powered up - it must have an initial program to run known as bootstrap program.
- The bootstrap program initializes all aspects of the system, from CPU registers to device controllers and the contents of main memory and then starts the operating system.
- To do its job, the bootstrap program finds the operating-system kernel on disk, loads that kernel into memory, and jumps to an initial address to begin the operating-system execution.
- For a number of computers, the bootstrap is stored in ROM (Read Only Memory). This location is convenient, because the ROM needs no initialization and is at a fixed location that the processor can start executing when powered up or reset.
- And since ROM is read only, it cannot be infected by a computer virus. The problem is that changing this bootstrap code requires changing the ROM hardware chips.
- For this reason, number of systems stores a tiny bootstrap loader program in the boot ROM whose only job is to bring in a full bootstrap program from disk.
- The full bootstrap program can be changed easily. A new version is simply written onto the disk.
- The full bootstrap program is stored in the "boot blocks" at a fixed location on the disk. A disk that has a boot partition is called a system disk or boot disk.
- The code in the boot ROM instructs the disk controller to read the boot blocks into memory (at this point no device drivers are loaded) and then starts executing that code.
- The full bootstrap program is more sophisticated than the bootstrap loader in the boot ROM; it is able to load the entire operating system from a non-fixed location on disk and to start the operating system running. Even so, the full bootstrap code may be small.
- Fig. 3.15 shows the boot process in the Windows 2000 system with its boot code in the first sector on the hard disk (which it terms the MBR (Master Boot Record).
- Windows 2000 system allows a hard disk to be divided into one or more partitions; one partition, identified as containing the operating system and device drivers.
- Booting begins in Windows 2000 by running code that is resident in the system's ROM memory and this code directs the system to read the boot code from the MBR.
- In addition to containing boot code, the MBR contains a table listing the partitions for the hard disk and a flag indicating which partition the system is to be booted from, as shown in Fig. 3.15.

- Once the system identifies the boot partition, it reads the first sector from that partition (which is called the boot sector) and continues with the remainder of the boot process, which includes loading the various subsystems and system services.

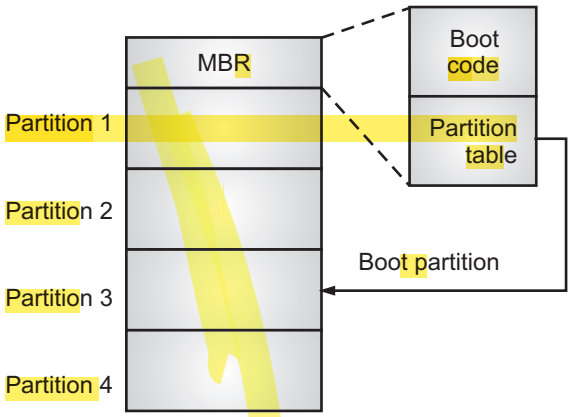


Fig. 3.15: Booting from Disk in Windows 2000 System

Bad Blocks:

- Because disks have moving parts and small tolerances, they are prone to failure. Sometimes the failure is complete; in this case, the disk needs to be replaced and its contents restored from backup media to the new disk.
- More frequently, one or more sectors become defective. Most disks even come from the factory with bad blocks.
- Depending on the disk and controller in use, these blocks are handled in a variety of ways.
- On simple disks, like some disks with IDE (Integrated Development Environment) controllers, bad blocks are handled manually.
- For example, the MS-DOS format command performs logical formatting and, as a part of the process, scans the disk to find bad blocks.
- If format finds a bad block, it writes a special value into the corresponding FAT (File Allocation Table) entry to tell the allocation routines not to use that block.
- If blocks go bad during normal operation, a special program like 'chkdsk' must be run manually to search for the bad blocks and to lock them away. Data that resided on the bad blocks usually are lost.
- The SCSI (Small Computer System Interface) disks used in high-end PCs and most workstations and servers, are smarter about bad-block recovery.
- The disk controller maintains a list of bad blocks on the disk. The list is initialized during the low-level/physical formatting at the factory and is updated over the life of the disk.
- Physical formatting also sets aside spare sectors not visible to the operating system. The disk controller can be told to replace each bad sector logically with one of the spare sectors known as forwarding or sector sparing.

- An alternative to sector some controllers can be instructed to replace a bad block by sector slipping.
- The replacement of a bad block generally is not totally automatic because the data in the bad block are usually lost.
- Soft errors may trigger a process in which a copy of the block data is made and the block is spared or slipped.
- An unrecoverable hard error, however, results in lost data. Whatever file was using that block must be repaired (for example, by restoration from a backup tape) and that requires manual intervention.
- Depending on the disk and controller in use, these blocks are handled in a variety of ways such as sector sparing or forwarding (controller can be told to replace each bad sector logically with one of the spare sectors), sector slipping (all sectors between the bad sector and the replacement sector are moved down by one) and so on.

PRACTICE QUESTIONS

Q. I Multiple Choice Questions:

1. Which is done by operating systems to schedule I/O requests arriving for the disk?
   (a) Disk scheduling (b) Disk formatting
   (c) Disk management (d) None of the mentioned
2. In which algorithm, the requests are served in the order they come. Those who come first are served first.
   (a) SSTF (b) SCAN
   (c) FCFS (d) LOOK
3. Which is a storage device that uses a magnetization process to write, rewrite and access data?
   (a) memory (b) drive
   (c) port (d) disks
4. In which algorithm, the disk arm moves in a particular direction till the last request is found in that direction and serves all of them found in the path, and then reverses its direction and serves the requests found in the path again up to the last request found.
   (a) SCAN (b) LOOK
   (c) SSTF (d) FCFS
5. SSTF stands for,
   (a) Shortest Seek Time First (b) Simplest Seek Time First
   (c) Shortest Speed Time First (d) None of the mentioned
6. In which algorithm the disk arm moves into a particular direction and services the requests coming in its path and after reaching the end of disk, it reverses its direction and again services the request arriving in its path.
   (a) LOOK (b) SSTF
   (c) SCAN (d) FCFS

7. A disk is divided into,
   (a) tracks (b) cylinders
   (c) sectors (d) All of the mentioned
8. In which, the disk arm in spite of going to the end goes only to the last request to be serviced in front of the head and then from there goes to the other end's last request.
   (a) C-LOOK (b) C-SACN
   (c) Both (a) and (b) (d) None of the mentioned
9. Disk management of the operating system includes:
   (a) Disk formatting (b) Booting from disk
   (c) Bad block recovery (d) All of the mentioned
10. Which is the time taken by the desired sector of disk to rotate into a position so that it can access the read/write heads?
    (a) Disk Access Time (b) Seek Time
    (c) Rotational Latency (d) Disk Response Time

Answers

1. (a)	2. (c)	3. (d)	4. (b)	5. (a)	6. (c)	7. (d)	8. (a)	9. (d)	10. (c)

Q. II Fill in the Blanks:

1. Nowadays, the popular secondary non-volatile storage/memory used in computers is magnetic _____ which is used to store all programs and files.
2. The disks surface is divided into concentric series of circles called _____.
3. Disk _____ algorithm manages those requests and decides the order of the disk access given to the requests.
4. A magnetic disk primarily consists of a rotating magnetic surface called as _____ and a mechanical arm that moves over it.
5. Hard disks are a type of_____ storage, retaining stored data even when powered off.
6. _____ time is the time taken by the arm to move to the required track. Rotational latency is defined as the time taken by the arm to reach the required sector in the track.
7. A _____ head is used to read data from a sector of the magnetic disk.
8. The main purpose of disk scheduling algorithm is to select a disk request from the queue of I/O _____ and decide the schedule when this request will be processed.
9. Disk _____ time is the average of time spent by each request waiting for the IO operation.
10. _____ disk scheduling algorithm services the IO requests in the order in which they arrive. A sector is the smallest unit of data that can be read or written from a disk.

11. When the computer is turned on or restarted, the program stored in the initial _____ ROM finds the location of the OS kernel from the disk, loads the kernel into memory, and runs the OS. start.
12. The _____ disk scheduling algorithm selects the disk I/O request which requires the least disk arm movement from its current position regardless of the direction. Access time is the amount of time taken to retrieve data from the hard drive.
13. _____ disk scheduling algorithm is also called as Elevator algorithm in which the disk arm moves into a particular direction till the end, satisfying all the requests coming in its path and then it turns backend moves in the reverse direction satisfying requests coming in its path.
14. Disks are error-prone because moving parts have small _____.
15. in _____ scheduling algorithm, the arm of the disk stops moving inwards (or outwards) when no more request in that direction exists.
16. In _____ disk scheduling algorithm, the arm of the disk moves in a particular direction servicing requests until it reaches the last cylinder, then it jumps to the last cylinder of the opposite direction without servicing any request then it turns back and start moving in that direction servicing the remaining requests.
17. A _____ error triggers the data recovery process.
18. Disk _____ is a process to configure the data-storage devices such as hard-drive, floppy disk and flash drive when we are going to use them for the very first time
19. In _____ the disk arm in spite of going to the end goes only to the last request to be serviced in front of the head and then from there goes to the other end's last request.
20. Disk scheduling algorithms are the algorithms that are used for scheduling a _____.

Answers

1. hard disk	2. tracks	3. scheduling	4. platter
5. non-volatile	6. Seek	7. read-write	8. requests
9. response	10. FCFS	11. bootstrap	12. SSTF
13. SCAN	14. tolerances	15. LOOK	16. C-SCAN
17. soft	18. formatting	19. C-LOOK	20. disk

Q. III State True or False:

1. A hard disk is volatile storage in computer system.
2. In FCFS, the requests are addressed in the order they arrive in the disk queue.
3. A hard disks is a collection of plates called as platters and each surface of disk is divided into circles called as tracks, each track is divided into smaller pieces is called as sectors, a group of tracks that are positions on top of each other is called as cylinder.

4. Disk scheduling algorithms are needed because a process can make multiple I/O requests and multiple processes run at the same time.
5. In CSAN algorithm in which the disk arm instead of reversing its direction goes to the other end of the disk and starts servicing the requests from there. So, the disk arm moves in a circular fashion and this algorithm is also similar to SCAN algorithm and hence it is known as C-SCAN (Circular SCAN).
6. Unrecoverable hard errors may result in data loss.
7. In SSTF, after each request is received, the controller selects the requests that is closed to the current position.
8. Disk scheduling algorithms are used to schedule single requests for accessing the disk.
9. In SCAN algorithm the disk arm moves into a particular direction and services the requests coming in its path and after reaching the end of disk, it reverses its direction and again services the request arriving in its path.
10. Response time is the average of time spent by a request waiting to perform its I/O operation.
11. Disk scheduling is done by operating systems to schedule I/O requests arriving for the disk.
12. In SSTF (Shortest Seek Time First), requests having shortest seek time are executed last.
13. In CLOOK, the disk arm in spite of going to the end goes only to the last request to be serviced in front of the head and then from there goes to the other end's last request.

Answers

1. (F)	2. (T)	3. (T)	4. (T)	5. (T)	6. (T)	7. (F)	8. (T)	9. (T)	10. (T)
11. (T)	12. (F)	13. (T)							

Q. IV Answer the following Questions:

(A) Short Answer Questions:

1. Define disk scheduling.
2. What is disk management?
3. Define rotational latency.
4. Define sector and tracks in hard disks.
5. Give the purpose of SCAN.
6. Define disk scheduling algorithm.
7. Give the purpose of disk scheduling.
8. What is meant by LOOK?
9. What are bad blocks?

10. Define disk formatting.
11. "The SCAN is a Elevator algorithm" Justify this statement.
12. What is C-LOOK?

(B) Long Answer Questions:

1. What is disk scheduling? Explain in detail.
2. What is disk? Explain structure of disk with diagram.
3. What is disk scheduling algorithm? Enlist them.
4. Explain SCAN disk scheduling algorithm with example.
5. Describe SSTF algorithm with example.
6. With the help of example describe FCFS.
7. Explain the disk performance parameters in detail.
8. Discuss advantages and disadvantages of FCFS, SSTF, SCAN and LOOK algorithms.
9. What C-SCAN and C-LOOK? Compare them.
10. With the help of diagram describe FCFS. Also states its advantages and disadvantages.
11. Differentiate between SCAN, LOOK and FCFS disk scheduling algorithms.
12. What is disk formatting? Describe in detail.
13. Suppose the order of requests are 70, 140, 50, 125, 30, 25, 160 and the initial position of the Read-Write head is 60. Calculate seek time for FCFS, SSTF, SCAN, LOOK, C-SCAN and C-LOOK disk scheduling algorithms.
14. Suppose the request queue sequence is 82, 170, 43, 140, 24, 16, 190 respectively, for a disk with 200 tracks. Consider the position of R/W head pointer is 50. Calculate the number of R/W head movements in cylinders, using FCFS, SCAN, SSTF, LOOK, C-SCAN and C-LOOK scheduling algorithms.
15. Suppose the request sequence is 176, 79, 34, 60, 92, 11, 41, 114 and initial head position is 50. Give seek time for FCFS, SSTF, SCAN and LOOK disk scheduling algorithms.

■■■

CHAPTER 4

Introduction to Distributed Operating Systems and Architecture

Objectives...

- To understand Concept of Distributed Systems and Distributed Operating Systems
- To learn Design Goals and Types of Distributed Systems
- To study different Architectural Styles and System Architectures

4.0 INTRODUCTION

- A distributed system is basically a computer network in which two or more autonomous computers are connected via their hardware and software interconnections, to facilitate communication using message passing.
- A message is a collection of data objects consisting of a fixed size header and a variable or constant length body, which can be managed by a proceed and delivered to is destination.
- A distributed system contains multiple nodes that are physically separate but linked together using the network.
- All the nodes in the distributed system communicate with each other and handle processes in tandem. Each of these nodes contains a small part of the distributed operating system software.
- A distributed system is composed of several computers that do not share a common memory and these computers communicate with each other by passing messages over a communication network.
- A distributed operating system is system software over a collection of independent, networked, communicating and physically separate computational nodes. They handle jobs which are serviced by multiple CPUs.
- A distributed operating system is generally designed to support system-wide sharing of resources, such as I/O devices, files, and computational capacity.

- A distributed operating system is an integration of system services, presenting a transparent view of multiple computer systems with distributed resources and control.
- An operating system that manages many computers but presents an interface of single computer to the user is called distributed OS.
- In this chapter we will study concepts of distributed systems and distributed operating systems.

4.1 WHAT IS DISTRIBUTED SYSTEM? [April 18]

- Advances in computer and communication technologies have revolutionized our modern society. The use of computers in managing information and automation has become an integral part of our daily life.
- Distributed systems with multiple computers connected through communication channels to facilitate information sharing and cooperative work are commonplace.
- Recently we have also experienced a significant growth of applications that require networking and distributed processing.
- As a result, various computer networks and distributed systems have been developed and put into practical use.
- A distributed system is a collection of loosely coupled processors interconnected by a communication network.
- A distributed system is a system consisting of a collection of autonomous machines connected by communication networks and equipped with software systems designed to produce an integrated and consistent computing environment.
- A distributed system is a system composed of several computers which communicate through a computer network, hosting execution of distributed activities.
- A computer network is a set of computers interconnected through communication links of possibly diverse media and topology and using a common set of communication protocols.
- A distributed computer system consists of multiple software components that are on multiple computers, but run as a single system.
- The key **purposes of distributed systems** are examined below:
  1. **Openness**: The openness of distributed systems is achieved by specifying the key software interface of the system and making it available to software developers so that the system can be extended in many ways.
  2. **Scalability:** A distributed system running on a collection of a small number of machines can be easily extended to a large number of machines to increase the processing power.

3. **Transparency:** Distributed systems can provide many forms of transparency such as Location transparency (which allows local and remote information to be accessed in a unified way), Failure transparency (which enables the masking of failures automatically) and Replication transparency (which allows duplicating software/data on multiple machines invisibly).
4. **Resource Sharing:** In a distributed system, the resources - hardware, software and data can be easily shared among users. For example, a printer can be shared among a group of users.
5. **Fault-Tolerance:** Machines connected by networks can be seen as redundant resources, a software system can be installed on multiple machines so that in the face of hardware faults or software failures, the faults or failures can be detected and tolerated by other machines.
6. **Concurrency:** The processing concurrency can be achieved by sending requests to multiple machines connected by networks at the same time.

- A general structure of a distributed system is shown in Fig. 4.1.

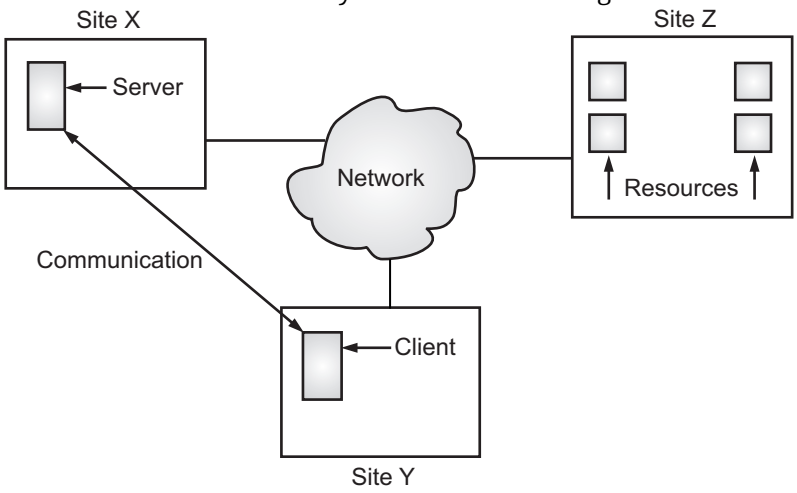


Fig. 4.1: A Typical Distributed System

- From the point of view of a specific processor in a distributed system, the rest of the processors and their respective resources are remote, whereas its own resources are local.
- The processors in a distributed system may vary in function and size. They may include small microprocessors, workstations, minicomputers and large general-purpose computer systems.
- These processors are referred to by a number of names, such as sites, nodes, computers, machines and hosts, depending on the context in which they are mentioned.
- We mainly use site to indicate the location of a machine and host to refer to a specific system at a site.

- Generally, one host at one site, the server, has a resource that another host at another site, the client (or user), would like to use.
- Any implementation of a distributed computing model (an abstract view of a system) must involve the implementation of processes, message links, routing schemes and timing.
- The most natural implementation of a distributed system is a network of computers, each of which runs more than one process.

4.1.1 Definition of Distributed System

- A distributed system is a system that organizes the computers on a network and communicates and coordinates through message passing.
- The main purpose of a distributed system is to enable users to access long-distance resources and share the resource like a text, a picture, a voice, a video and so on with other users in a controlled way.
- A distributed system is defined as, "a collection of autonomous computers that appear to the users of the system as a single computer."

Advantages of Distributed Systems:

1. **Improved Reliability and Availability:** Improved reliability and availability is provided by distributed systems because failure of few components of the system does not affect the availability of the rest of the systems. Distributed systems can also be made fault tolerant, through the replication of data and services (processes that provide functionality).
2. **Modular Expandability:** New resources either hardware or software can be easily added in a distributed system environment, without replacing it with the existing resources.
3. **Resource Sharing:** Whenever, a computer requests for a service from another computer, it simply transmits an appropriate request to that computer over a communication network. Such a communication takes place by sharing resources among computers over the network.
4. **Enhanced Performance:** Providing rapid response time and higher system throughput is the ability of distributed systems and this is mainly because of the fact that many tasks are executed concurrently at different computers. Moreover, to improve the response time, load distributing technique is used by the distributed system.
5. **Improved Efficiency:** Distributed systems are highly efficient as they involve multiple computers that save time for users. Also, they can provide higher performance as compared to centralized systems.

Disadvantages of Distributed Systems:

1. **Security:** Security of information resources available and maintained in a distributed system is very important. It is difficult to provide adequate security in distributed systems because the nodes as well as the connections need to be secured.
2. **Software:** This is the worst problem for a distributed system. At present, very few software are available for a distributed system, because designing, implementing and using such distributed software is a challenging task till date.
3. **High Set-up Cost:** The initial cost of installation and set-up is high due to many hardware and software devices. There are other maintenance costs associated with the system which adds to the total cost, making it even more expensive.
4. **Overloading:** Overloading may occur in the network if all the nodes of the distributed system try to send data at once.
5. **Data Loss:** This problem arises in a distributed system due to its communication network. When two or more devices are communicating, it is possible that they lose messages over the communication network.

4.1.2 Design Goals of Distributed System

- There are following major reasons for building/designing distributed systems namely, resource sharing, scalability, computation speedup, reliability, flexibility and communication.

Supporting Resource Sharing:

- Distributed systems allow users to share resources on geographically dispersed hosts connected via a computer network.
- An important goal of a distributed system is to make it easy for users and applications to access and share remote resources.
- Resources can be virtually anything, but typical examples include peripherals, storage facilities, data, files, services, and networks, to name just a few.
- When a group of users are working together they need to communicate with each other, to share data and resources.
- The requirement of a distributed system came from the fact of sharing resources among users. These resources could be hardware (such as disks and printers) or software (such as files, databases and data objects etc.).
- Resource sharing in a distributed system provides mechanisms for sharing resources like files at remote sites, processing information in a distributed database, printing files at remote sites, using remote specialized hardware devices (such as a high-speed array processor), and performing other operations.

- Connecting users and resources also makes it easier and simpler to collaborate and exchange information, as is illustrated by the success of the Internet with its simple protocols for exchanging files, mail, documents, images, audio and video.
- However, resource sharing in distributed systems is perhaps best illustrated by the success of file-sharing peer-to-peer networks such as BitTorrent. These distributed systems make it extremely simple for users to share files across the Internet.
- Peer-to-peer networks are often associated with distribution of media files such as audio and video.
- In other cases, the technology is used for distributing large amounts of data, as in the case of software updates, backup services, and data synchronization across multiple servers.

Being Scalable:

- Scalability has become one of the most important design goals for development of distributed systems. Scalability means that we can change the size and extent of a particular system.
- Scalability refers to the capability of a system to adapt to increased service load. It is inevitable that a distributed system will grow with time since it is very common to add new machines or an entire sub-network to the system to take care of increased workload or organizational changes in a company.
- Therefore, a distributed operating system should be designed to easily cope with the growth of nodes and uses in the system i.e., such growth should not cause serious disruption of service or significant loss of performance to users.
- Scalability of a system can be measured along at least following three dimensions:
  1. **Geographical Scalability:** A geographically scalable system is one in which the users and resources may lie far apart, but the fact that communication delays may be significant is hardly noticed.
  2. **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 without any noticeable loss of performance.
  3. **Administrative Scalability:** An administratively scalable system is one that can still be easily managed even if it spans many independent administrative organizations.

Computation Speedup:

- There is a need for speeding up the computation. If a particular computation can be partitioned into sub-computations that can run concurrently, then a distributed system allows us to distribute the sub-computations among the various sites; the sub-computations can be run concurrently and thus provide computation speedup.

- In addition, if a particular site is currently overloaded with jobs, some of them can be moved to other, lightly loaded sites. This movement of jobs is called load sharing.
- Automated load sharing, in which the distributed operating system automatically moves jobs, is not yet common in commercial systems.

Making Distribution Transparent:

- An important design goal of a distributed system is to hide the fact that its processes and resources are physically distributed across multiple computers, possibly separated by large distances.
- In other words, it tries to make the distribution of processes and resources transparent that is invisible to end users and applications.
- The concept of transparency can be applied to several aspects of a distributed system. Some of them are explained below:

1. **Access Transparency**: Access transparency refers to the ability to access both local and remote system objects in a uniform way. The physical separation of system objects is concealed from the user.
2. **Location Transparency:** It refers to the fact that users cannot tell where an object is physically located in the system. Location transparency means that users have no awareness of object locations. Objects are mapped and referred to by logical names. This transparency is sometimes called name transparency.
3. **Replication Transparency:** It deals with hiding the fact that several copies of a resource exist. Replication transparency exhibits consistency of multiple instances (or partitioning) of files and data. It is closely related to concurrency transparency but is treated separately since files and data are special objects.
4. **Relocation Transparency:** It refers to being moved by the distributed system.
5. **Migration Transparency:** It is offered by a distributed system when it supports the mobility of processes and resources irritated by users, with-out affecting ongoing communication and operations. Migration transparency is an added property of location transparency where an object is not only referred to by its logical name but can also be moved to a different physical location without changing the name. Migration transparency is also called location independence.
6. **Concurrency Transparency:** It is important that each user does not notice that the other is making use of the same resource. This phenomenon is called concurrency transparency. It allows the sharing of objects without interference; it is similar to the time-sharing concept in a broader sense.
7. **Failure Transparency**: This transparency provides fault tolerance such that failures in the system can be transformed into graceful system performance degradation rather than disruptions and damage to the users is minimized.
8. **Size Transparency**: This transparency is related to modularity and scalability. It allows incremental growth of a system without the user's awareness. The size of the system has no effect on the user‘s perception of the system.

Reliability:

- An important design goal of a distributed system is reliability. Reliability denotes the ability of a distributed system to deliver its services even when one or several of its software or hardware components fails.
- If one site fails in a distributed system, the remaining sites can continue operating, giving the system better reliability.
- If the system is composed of multiple large autonomous installations (that is, general-purpose computers), the failure of one of them should not affect the rest.
- If, however, the system is composed of small machines, each of which is responsible for some crucial system function (such as terminal character I/O or the file system), then a single failure may halt the operation of the whole system.
- In general, with enough redundancy (in both hardware and data), the system can continue operation; even if some of its sites have failed.

Being Open:

- The important goal of distributed systems is openness. An open distributed system is essentially a system that offers components that can easily be used by, or integrated into other systems.
- At the same time, an open distributed system itself will often consist of components that originate from elsewhere.
- To be open means that components should adhere to standard rules that describe the syntax and semantics of what those components have to offer (i.e., which service they provide).
- A general approach is to define services through interfaces using an Interface Definition Language (IDL).

Interoperability, Portability and Extensibility:

- Interoperability characterizes the extent by which two implementations of systems or components from different manufacturers can co-exist and work together by merely relying on each other's services as specified by a common standard.
- Portability characterizes to what extent an application developed for a distributed system X can be executed, without modification, on a different distributed system Y that implements the same interfaces as X.
- The important goal for an open distributed system is that it should be easy to configure the system out of different components.
- Also, it should be easy to add new components or replace existing ones without affecting those components that stay in place.
- In other words, an open distributed system should also be extensible. For example, in an extensible system, it should be relatively easy to add parts that run on a different operating system, or even to replace an entire file system.

Flexibility:

- It is an important goal in the design of distributed systems. Flexibility is the most important feature for open distributed systems.
- The design of a distributed operating system should be flexible due to the following reason:
  1. **Easy Modification:** From the experience of system designers, it has been found that some parts of the design often need to be replaced/modified either because some bug is detected in the design or because the design is no longer suitable for the changed system environment or new-user requirements. Therefore, it should be easy to incorporate changes in the system in a user-transparent manner or with minimum interruption caused to the users.
  2. **Easy Enhancement:** In every system, new functionalities have to be added from time to time to make it more powerful and easy to use. Therefore, it should be easy to add new services to the system. Furthermore, if a group of users do not like the style in which a particular service is provided by the operating system, they should have the flexibility to add and use their own service that works in the style with which the users of that group are more familiar and feel more comfortable.
- To achieve flexibility in open distributed systems, it is crucial that the system be organized as a collection of relatively small and easily replaceable or adaptable components.

Communication:

- Communication is an important goal in the design of distributed systems. When multiple sites are connected to one another by a communication network, users at the various sites have the opportunity to exchange information.
- At a low level, messages are passed between systems, much as messages are passed between processes in the single-computer message system.
- Given message passing, all the higher-level functionality found in standalone systems can be expanded to encompass the distributed system. Such functions include file transfer, login, mail and Remote Procedure Calls (RPCs).

4.2 TYPES OF DISTRIBUTED SYSTEMS

- A distributed system is a collection of loosely coupled processors interconnected by a computer network. The various types of distributed systems are explained below:

High Performance Distributed Computing:

- An important type of distributed systems is the one used for high-performance computing tasks.
- Roughly speaking, one can make a distinction between two subgroups namely, cluster computing and grid computing.

- In **cluster computing** the underlying hardware consists of a collection of similar workstations or PCs, closely connected by means of a high-speed local-area network. In addition, each node runs the same operating system.
- The **grid computing** subgroup consists of distributed systems that are often constructed as a federation of computer systems, where each system may fall under a different administrative domain, and may be very different when it comes to hardware, software, and deployed network technology. From the perspective of grid computing, a next logical step is to simply outsource the entire infrastructure that is needed for compute-intensive applications. In essence, this is what **cloud computing** is all about: providing the facilities to dynamically construct an infrastructure and compose what is needed from available services.

- Grid computing is distinguished from conventional high-performance computing systems such as cluster computing in that grid computers have each node set to perform a different task/application.
- Grid computers also tend to be more heterogeneous and geographically dispersed (thus not physically coupled) than cluster computers.

Cluster Computing:

- Cluster computing is an approach to achieving high performance, high reliability or high throughput computing by using a collection of interconnected computer systems.
- Cluster computing systems became popular when the price/performance ratio of personal computers and workstations improved.
- At a certain point, it became financially and technically attractive to build a supercomputer using off-the-shelf technology by simply hooking up a collection of relatively simple computers in a high-speed network.
- In virtually all cases, cluster computing is used for parallel programming in which a single (compute intensive) program is run in parallel on multiple machines.
- A cluster is a type of parallel or distributed processing system, which consists of a collection of interconnected stand-alone computers working together as a single, integrated computing resource.
- A cluster generally refers to two or more computers (nodes) connected together.
- Cluster computing works on the distributed system with the networks. Cluster computing depicts a system that consists of two or more computers or systems, often known as nodes.
- These nodes work together for executing applications and performing other tasks. The users using nodes have an apprehension that only a single system responds to them, creating an illusion of a single resource called virtual machines.

- A computer cluster is a set of computers that work together so that they can be viewed as a single system. Computer clusters have each node set to perform the same task, controlled and scheduled by software.
- Computer cluster is a group of computers connected together (mainly and mostly in LAN) to do a task so that they whole together appear as a single computer.
- Fig. 4.2 shows example of a cluster computer formed by Linux-based Beowulf clusters. Each cluster consists of a collection of corrupter nodes that are controlled and accessed by means of a single master node.
- The master typically handles the allocation of nodes to a particular parallel program, maintains a batch queue of submitted jobs, and provides an interface for the users of the system.
- As such, the master actually runs the middleware needed for the execution of programs and management of the cluster, while the compute nodes are equipped with a standard operating system extended with typical middleware functions for communication, storage, fault tolerance, and so on. Apart from the master node, the compute nodes are thus seen to be highly identical.
- The components of a cluster are usually connected to each other through fast local area networks, with each node (computer used as a server) running its own instance of an operating system.

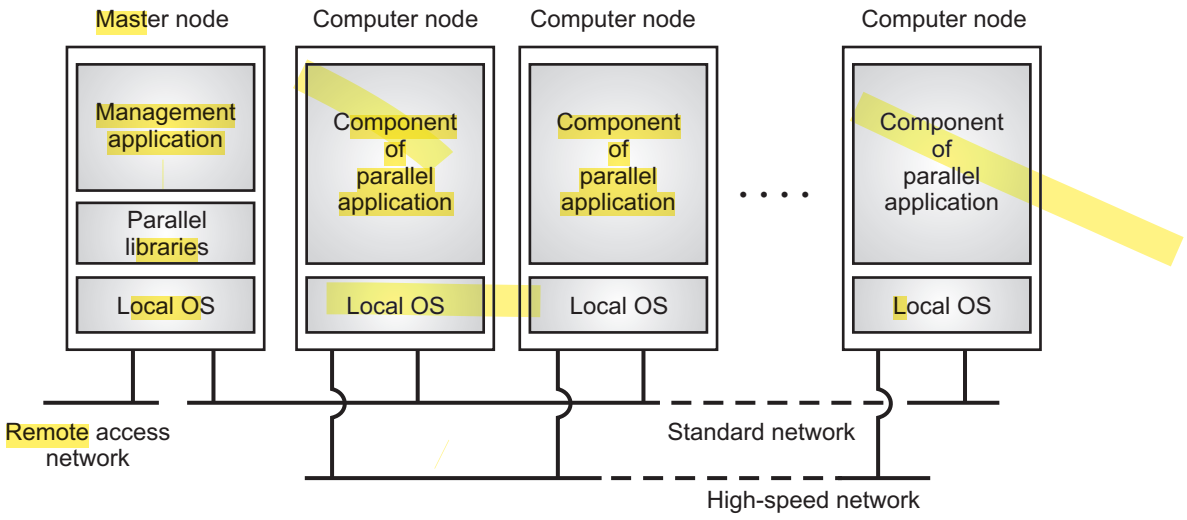


Fig. 4.2: Example of Cluster Computing System

- A cluster is a group of interconnected computers that work together to perform computationally intensive tasks. In a cluster, each computer is referred to as a "node".

- Fig. 4.3 shows a cluster computer network.

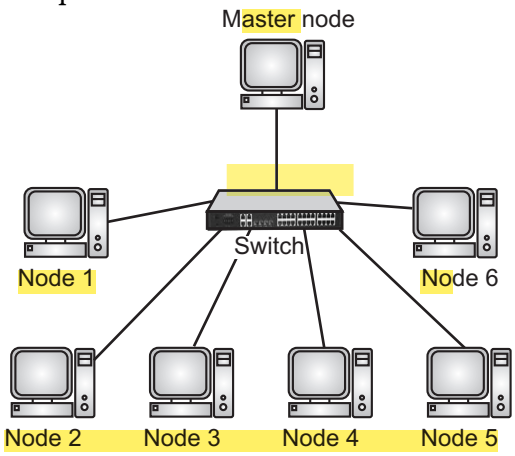


Fig. 4.3: Typical Cluster Computing Network

Advantages of Cluster Computing:

1. **Cost-effectiveness:** Compared with the mainframe systems, cluster computing is considered to be much more cost-effective. These computing systems offer enhanced performance with respect to the mainframe computer devices.
2. **Processing Speed:** The processing speed of cluster computing is justified with that of the mainframe systems and other supercomputers present in the world.
3. **Expandability:** Scalability and expandability are another set of advantages that cluster computing offers. Cluster computing represents an opportunity for adding any number of additional resources and systems to the existing computing network.
4. **High Availability of Resource:** Availability plays a vital role in cluster computing systems. Failure of any connected active node can be easily passed on to other active nodes on the server, ensuring high availability.
5. **Improved Flexibility:** In cluster computing, superior specifications can be upgraded and extended by adding new nodes to the existing server.
6. **Reliability:** As the cluster computing system continues to operate even in case of failures in parts of it, obviously with lower performance.

Disadvantages of Cluster Computing:

1. **Cost is High:** Since the cluster needs good hardware and a design, it will be costly compared to a non-clustered server management design.
2. **Hard to Monitor:** Since clustering needs more servers and hardware to establish one, monitoring and maintenance is hard.
3. **Difficult to Manage and Organize:** In cloud computing a large number of computers are difficult to manage and organize.

- Some of the critical applications of cluster computers are Google Search Engine, Petroleum Reservoir Simulation, Earthquake Simulation, Weather Forecasting and so on.

Grid Computing:

- Grid computing is an emerging environment to solve large scale complex problems. Grid computing is a form of distributed computing whereby a "super and virtual computer" is composed of a cluster of networked, loosely coupled computers, acting in concert to perform very large tasks.
- Grid computing includes concepts of distributed computing, high performance computing, and disposable computing, depending upon the exact nature and scale of the application of the grid.
- Grid Computing refers to distributed computing, in which a group of computers from multiple locations are connected with each other to achieve a common objective/goal.
- Grid computing refers to a network of same or different types of computers whose target is to provide an environment where a task can be performed by multiple computers together on need basis. Each computer can work independently as well.
- Grid computing is a distributed computing approach where the end user will be ubiquitously offered any of the services of a “grid” or a network of computer systems located either in a Local Area Network (LAN) or in a Wide Area Network (WAN) in a spread of geographical area.
- It aims at dynamic or “runtime” selection, allocation and control of computational resources such as processing power, disk storage or specialized software systems or even data, according to the demands of the end-users.
- A grid is a collection of distributed computing resources over a local or wide area network that appears to an end-user or application as one large virtual computing system.
- Fig. 4.4 shows a grid computer network.
- A key issue in a grid-computing system is that resources from different organizations are brought together to allow the collaboration of a group of people from different institutions, indeed forming a federation of systems. Such collaboration is realized in the form of a virtual organization.
- The processes belonging to the same virtual organization have access rights to the resources that are provided to that organization.
- Typically, resources consist of computer servers (including supercomputers, possibly implemented as cluster computers), storage facilities, and databases.
- In addition, special networked devices such as telescopes, sensors etc., can be provided as well.

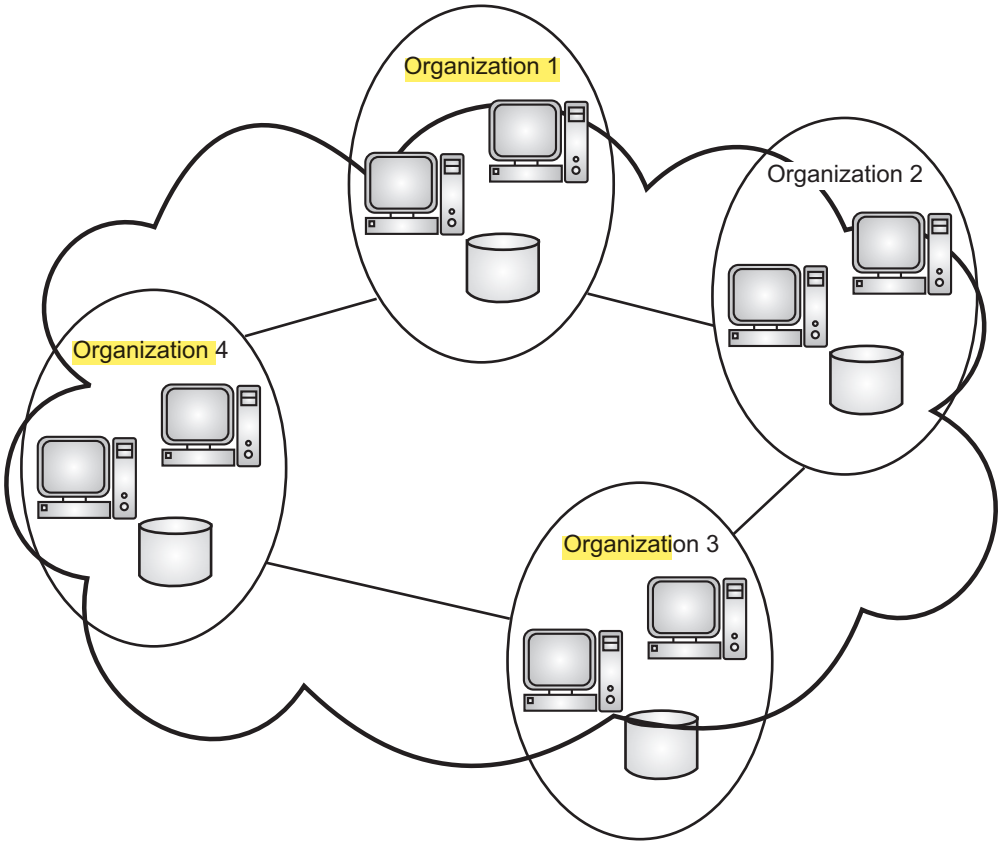


Fig. 4.4: Typical Grid Computing Network

Layered Architecture of Grid Computing:

- Grid computing is the use of widely distributed computer resources to reach a common goal.

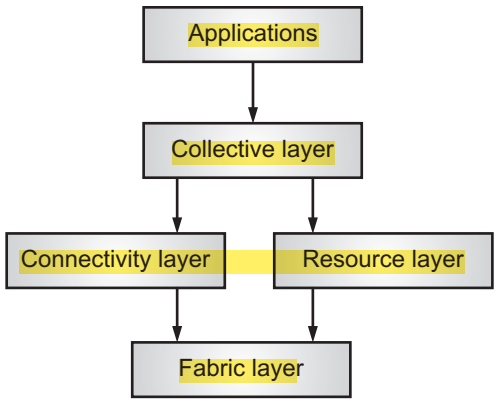


Fig. 4.5: A Layered Architecture of Grid Computing

- An architecture initially proposed by Foster is shown in Fig. 4.5, which still forms the basis for many grid computing systems.

- Typically the collective, connectivity, and resource layer form the heart of what could be called a grid middleware layer.
- These layers jointly provide access to and management of resources that are potentially dispersed across multiple sites.
- The layered architecture of grid computing system consists of following layers:
  - **Fabric Layer:** The lowest fabric layer provides interfaces to local resources at a specific site. Note that these interfaces are tailored to allow sharing of resources within a virtual organization. Typically, they will provide functions for querying the state and capabilities of a resource, along with functions for actual resource management (e.g., locking resources).
  - **Connectivity Layer:** The connectivity layer consists of communication protocols for supporting grid transactions that span the usage of multiple resources. For example, protocols are needed to transfer data between resources, or to simply access a resource from a remote location. In addition, the connectivity layer will contain security protocols to authenticate users and resources.
  - **Resource Layer:** The resource layer is responsible for managing a single resource. It uses the functions provided by the connectivity layer and calls directly the interfaces made available by the fabric layer. For example, this layer will offer functions for obtaining configuration information on a specific resource, or, in general, to perform specific operations such as creating a process or reading data. The resource layer is thus seen to be responsible for access control, and hence will rely on the authentication performed as part of the connectivity layer.
  - **Collective Layer:** It deals with handling access to multiple resources and typically consists of services for resource discovery, allocation and scheduling of tasks onto multiple resources, data replication, and so on. Unlike the connectivity and resource layer, each consisting of a relatively small, standard collection of protocols, the collective layer may consist of many different protocols reflecting the broad spectrum of services it may offer to a virtual organization.
  - **Application Layer:** It consists of the applications that operate within a virtual organization and which make use of the grid computing environment.

Advantages of Grid Computing:

1. Grid computing environments are very modular in performance.
2. Grid computing ensures easy scaling of applications.
3. Grid computing adopts the use of open source, trust, transparency, and technology.
4. Grid computing allows seamless sharing and distribution of computing resources across networks.

5. Grid computing is also capable of making better use of the hardware that already exists.
6. Grid computing has also proved to be useful in combining with other organizations.
7. The failure rate of grid computing is low.

Disadvantages of Grid Computing:

1. Grid computing requires robust and fast interconnection between resources.
2. It suffers from proprietary approaches.

Cloud Computing:

- Cloud computing is the delivery of on-demand computing services from applications to storage and processing power over the internet and on a pay-as-you-go basis.
- Organizations have moved to cloud platforms for better scalability, mobility and security.
- Cloud computing is the on-demand availability of computer system resources, especially data storage (cloud storage) and computing power, without direct active management by the user.
- Cloud computing is on-demand access, via the internet, to computing resources - applications, servers (physical servers and virtual servers), data storage, development tools, networking capabilities, and more - hosted at a remote data center managed by a Cloud Services Provider (CSP).
- Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction.
- Forrester defines cloud computing as, "a pool of abstracted, highly scalable and managed compute infrastructure capable of hosting end-customer applications and billed by consumption".
- Cloud Computing provides us means of accessing the applications as utilities over the Internet. It allows us to create, configure, and customize the applications online.

What is Cloud?

- The term cloud refers to a network or the internet. It is a technology that uses remote servers on the internet to store, manage, and access data online rather than local drives. The data can be anything such as files, images, documents, audio, video, and more.
- Cloud can be defined as a Wide Area Network (WAN) community that houses data centers and their associated software/hardware applications, other Information Technology (IT) resources and infrastructures that can be accessed using internet connectivity.

- A cloud is a combination of servers, storages, network devices (like switches etc.) networked together from where computing facilities are delivered to users.
- Cloud facility is developed by activating this specially designed network of commodity computing components with advanced software to power them up for the task.

Types of Cloud:

- Various types of clouds are explained below:

1. Public Cloud:

- Public cloud is open to all to store and access information via the Internet using the pay-per-usage method.
- The public cloud allows systems and services to be easily accessible to the general public.
- In the public cloud, computing resources are managed and operated by the Cloud Service Provider (CSP).
- **Example:** Amazon elastic compute cloud (EC2), IBM SmartCloud Enterprise, Microsoft, Google App Engine, Windows Azure Services Platform.

2. Private Cloud:

- The private cloud is private in nature and allows systems and services to be accessible within an organization. It is more secured because of its.
- Private cloud is also known as an internal cloud or corporate cloud. It is used by organizations to build and manage their own data centers internally or by the third party.
- It can be deployed using Opensource tools such as Openstack and Eucalyptus.

3. Hybrid Cloud:

- Hybrid cloud is a combination of the public cloud and the private cloud.
- Hybrid cloud is partially secure because the services which are running on the public cloud can be accessed by anyone, while the services which are running on a private cloud can be accessed only by the organization's users.
- **Example:** Google Application Suite (Gmail, Google Apps, and Google Drive), Office 365 (MS Office on the Web and OneDrive), Amazon Web Services.

4. Community Cloud:

- Community cloud allows systems and services to be accessible by a group of several organizations to share the information between the organization and a specific community.
- It is owned, managed, and operated by one or more organizations in the community, a third party, or a combination of them.
- **Example:** Health Care community cloud.

- Cloud computing refers to manipulating, configuring, and accessing the hardware and software resources remotely. It offers online data storage, infrastructure, and application.

Characteristics of Cloud Computing:

- Cloud computing characteristics are typically distinctive features and qualities that differentiate such systems from other seemingly-related systems.
- These characteristics are explained below:
  1. **Broad Network Access:** Real-time access and use of cloud services via the internet using any platform (laptops, mobile phones and so on) is made possible anytime regardless of the geographical location.
  2. **Reliability:** The use of multiple redundant sites is an approach being adopted to attain reliability. The higher the reliability the lower the cloud service usage risks. It also increases the confidence of potential cloud consumers and users to adopt cloud for business-critical tasks and disaster recovery.
  3. **Efficient Resource Utilization:** The cloud architecture ensures that resources that are delivered are efficiently utilized and made available only for the period needed.
  4. **On-Demand Self-Service:** Cloud services are automatically made available without human interference when required by a potential consumer. These services include, but are not limited to processing power, web applications, server time, storage and networks.
  5. **High Performance:** Cloud computing has very large storage and high computing specifications that characterize high-performance computing environments.
  6. **Network Access:** Cloud allows resources and services to be accessed using any internet-ready devices like the portable digital assistants, laptops, personal computers and so on using protocols like the HyperText Transfer Protocol (HTTP), internet and transmission control technologies.
  7. **Customization:** Cloud provides a highly flexible and reconfigurable environment that allows you to customize a set of cloud services, infrastructures and applications to meet users' specifications.
  8. **High Scalability:** cloud computing architecture allows us to add new nodes and servers with no reconfiguration and re-modification requirements to cloud software and infrastructure.
  9. **Green Technology:** Cloud computing is an energy-efficient technology that does not require huge power consumption.
- Cloud computing is a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) network on demand over the internet.

- The Cloud Computing Ecosystem (CCE) is a dynamic and complex community of the cloud computing system components and the stakeholders.
- The interdependent components of the CCE include cloud consultants, cloud service providers, cloud end-users (customers), cloud product manufacturers, cloud engineers, cloud partners, high speed network, the cloud management environment as well as the cloud computing infrastructures and IT resources that are provisioned as services.
- Fig. 4.6 shows a cloud computing network.

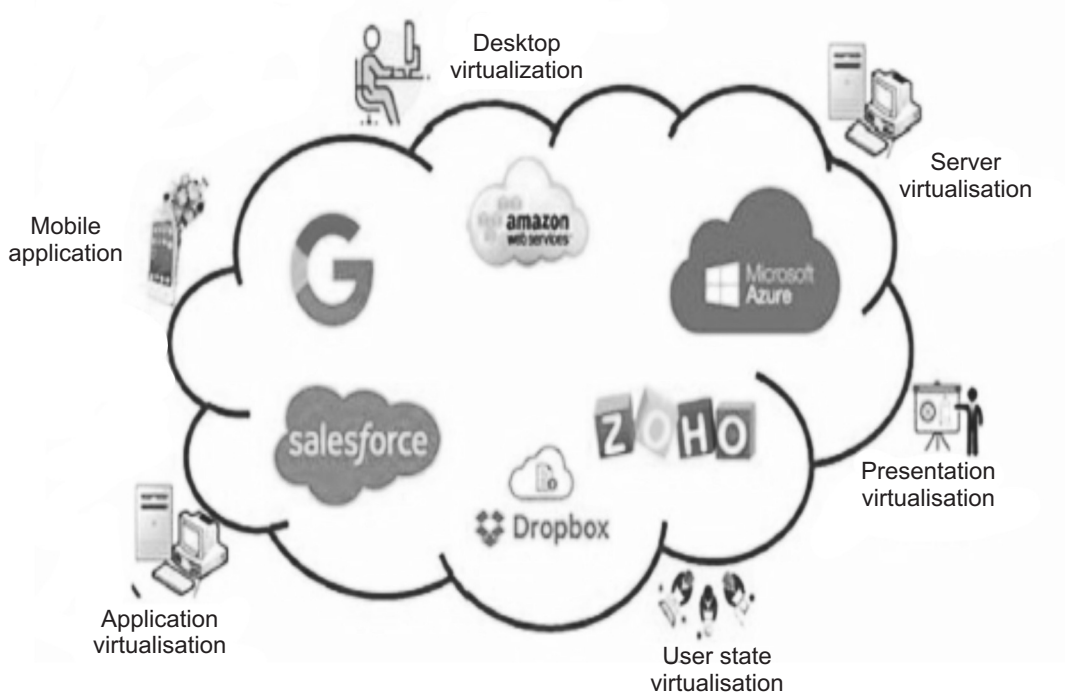


Fig. 4.6: Typical Cloud Computing Network

Layered Architecture of Cloud Computing:

- Fig. 4.7 shows the layered architecture of cloud computing.
- Cloud computing is characterized by an easily usable and accessible pool of virtualized resources.
- Which and how resources are used can be configured dynamically, providing the basis for scalability (if more work needs to be done, a customer can simply acquire more resources).
- The link to utility computing is formed by the fact that cloud computing is generally based on a pay-per-use model in which guarantees are offered by means of customized Service Level Agreements (SLAs).

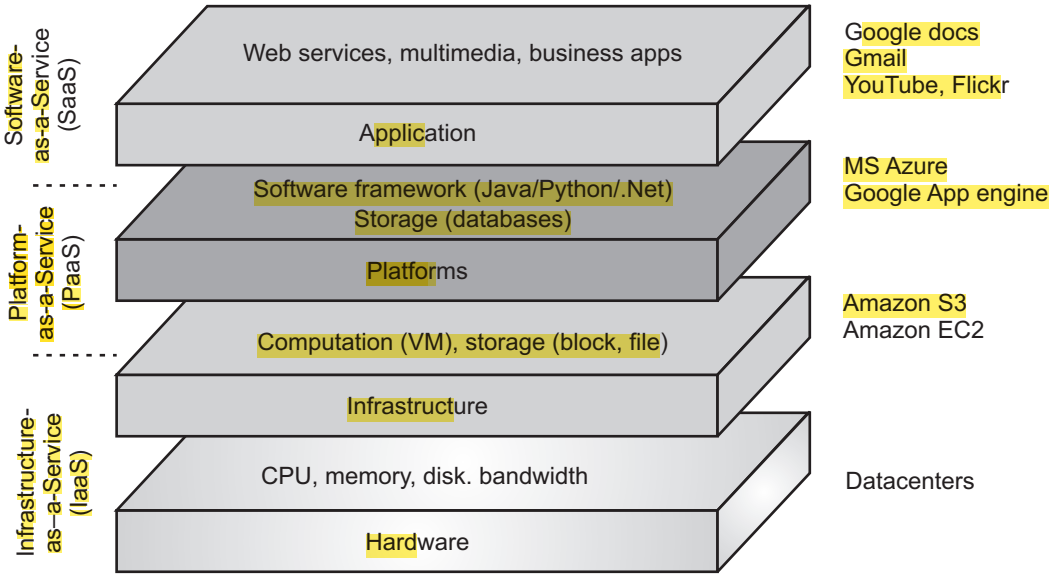


Fig. 4.7: Organization of Clouds/Layered Architecture of Cloud Computing

- The layered architecture of cloud computing consists of following layers:
  - **Hardware Layer:** The lowest layer is formed by the means to manage the necessary hardware such as processors, routers, but also power and cooling systems. It is generally implemented at data centers and contains the resources that customers normally never get to see directly.
  - **Infrastructure Layer:** This is an important layer forming the backbone for most cloud computing platforms. It deploys virtualization techniques to provide customers an infrastructure consisting of virtual storage and computing resources. Indeed, nothing is what it seems: cloud computing evolves around allocating and managing virtual storage devices and virtual servers.
  - **Platform Layer:** One could argue that the platform layer provides to a cloud-computing customer what an operating system provides to application developers, namely the means to easily develop and deploy applications that need to run in a cloud. In practice, an application developer is offered a vendor-specific API, which includes calls to uploading and executing a program in that vendor's cloud. In a sense, this is comparable to the Unix exec family of system calls, which take an executable file as a parameter and pass it to the operating system to be executed.
  - **Application Layer:** Actual applications run in this layer and are offered to users for further customization. Well-known examples include those found in office suites (text processors, spreadsheet applications, presentation applications, and so on). It is important to realize that these applications are again executed in the vendor's cloud. As before, they can be compared to the traditional suite of applications that are shipped when installing an operating system.

- Cloud computing providers offer these layers to their customers through various interfaces (including command-line tools, programming interfaces and Web interfaces), leading to following three different types of services:
  1. **Infrastructure-as-a-Service (IaaS)** covering the hardware and infrastructure layer. The IaaS provides access to fundamental resources such as physical machines, virtual machines, virtual storage, etc. IaaS provides on-demand access to fundamental computing resources–physical and virtual servers, networking, and storage-over the internet on a pay-as-you-go basis.
  2. **Platform-as-a-Service (PaaS)** covering the platform layer. PaaS provides the runtime environment for applications, development and deployment tools, etc. PaaS provides software developers with on-demand platforms - hardware, complete software stack, infrastructure and even development tools - for running, developing, and managing applications without the cost, complexity, and inflexibility of maintaining that platform on-premises.
  3. **Software-as-a-Service (SaaS)** in which their applications are covered. The SaaS model allows using software applications as a service to end-users. SaaS also known as cloud-based software or cloud applications - is application software that's hosted in the cloud and that we access and use via a web browser, a dedicated desktop client, or an API that integrates with our desktop or mobile operating system.

Advantages of Cloud Computing:

1. **Cost Reduction**: Cloud computing is seen as an incremental investment, companies can save money in the long term by obtaining resources.
2. **Storage Increase**: Instead of purchasing large amounts of storage before the need, organizations can increase storage incrementally, requesting additional disk space on the service provider when the need is recognized.
3. **Resource Pooling:** in the IT industry, this feature is also known as Multi-tenancy, where many users / clients share a type and varied level of resources.
4. **Highly Automated**: As the software and hardware requirements are hosted on a cloud provider, IT departments sites no longer have to worry about keeping the things up-to-date and available.
5. **Greater Mobility:** Once the information is stored in the cloud, access is quite simple, just you have an Internet connection, regardless of where they are located.
6. **Change the IT Focus**: Once the responsibility of the computing environment has, essentially shifted to the cloud provider, IT departments can now focus more on the organization's needs and the development of strategic applications and tactics and not on operational needs of the day-to-day.

7. **Towards Green IT:** By releasing the physical space, virtualization of applications and servers contributes to the reduction of equipment as well as the need for air conditioning, consequently, less energy waste.
8. **Unlimited Storage:** Cloud provides access to very large storage space based on users' requirements.
9. **Backup and Recovery:** Cloud offers backup and recovery services for the most traditional physical computing infrastructures. It is relatively much easier and safer to backup and restore important data than to store data on tertiary devices.
10. **Automatic Software Integration:** Cloud ensures real-time and seamless integration of its applications to the users' environment and system.
11. **Quick Deployment:** The cloud is ready for use once the relevant cloud functionalities and parameter settings are completed.

Disadvantages of Cloud Computing:

1. **Lack of Standards:** Most documented interfaces of cloud have no associated standards which might later pose interoperability issues. This challenge is currently being pursued by the open grid forum.
2. **Prone to Attack:** Cloud is more potentially prone to security threats and external hack attacks as there is a large amount of sensitive information on the cloud.
3. **Continuously Evolving:** Cloud is continuously evolving and so are the user requirements and the requirements for storage, interfaces and networking.
4. **Service Migration:** User lock-in to a particular provider is a problem after service migration to the cloud. This is due to non-existence of a standardized external interface in cloud computing.
5. **Security and Privacy:** Cloud computing still suffers from some security and privacy challenges. The major concerns are data protection and confidentiality, disaster and data breach as well as user authentication. Data protection is often managed using data encryption with defined roles and privileges rim data encryption keys management.

Differences between Cluster Computing, Grid Computing and Cloud Computing:

Sr. No.	Characteristics	Cluster Computing	Grid Computing	Cloud Computing
1.	Ownership	Single.	Multiple.	Single.
2.	Service pricing	Limited.	Private or Public assigned.	Utility/Large user discount.
3.	Virtualization	Half.	Half.	Yes.

contd. ...

4.	Resource management	Centralized resource.	Distributed resource.	Both.
5.	Scalable size	100s.	1000s.	100 to 1000s.
6.	Standardized	Yes.	Yes.	No.
7.	Interoperability	Yes.	Yes.	Not full.
8.	Speed/ Interconnected network	Dedicated high end with low latency and high bandwidth.	Mostly internet with high latency and low bandwidth.	Dedicated high end with low latency and high bandwidth.
9.	Self-service	No.	Yes.	Yes.
10.	Single system image	Yes.	No.	Yes/Optional included.
11.	Multi-tenancy	No.	Yes.	Yes.
12.	Service negotiation	Limited.	Yes, SLA-based.	Yes, SLA-based.
13.	Membership discovery	Membership service discovery.	Decentralized information services and centralized indexing.	Membership service discovery.
14.	Operating system	Windows/Linux.	Any standard but dominated by Unix.	Uses a hypervisor.
15.	Application drivers	Business, data centers, enterprise computing.	Collaborative scientific and high-throughput applications.	Web App, content delivery, dynamic provisioning.
16.	Standards/ Interoperability	Virtual Interface Architecture (VIA).	Some open grid forum.	Web services (SOAP and REST).
17.	Scalable	No.	Half.	Yes.
18.	Failure management	Limited (often failed task)/application and restarted.	Limited (often failed task)/application restarted.	Failover, content replication, virtual machine migration from one node to another supported.

contd. …

19.	Capacity	Stable and guaranteed capacity.	Varies, but high capacity.	Provisioned on-demand capacity
20.	Security	Traditional login/ password-based.	Public/private pair-based authentication and mapping of a user to an account.	Each user and/or application is provided with a virtual machine.
21.	Privacy	Medium level of privacy depends on user privileges.	Limited support for privacy.	High security/ privacy is guaranteed. There is support for file Access Control List (ACL) settings.
22.	Population	Commodity computers.	High-end computing systems (including clusters and servers).	Commodity PCs, high-end servers' network, attached storage.
23.	End-user presentation	Presented as a dynamic and diversified system.	Presented as a single system image.	Presented as a self-services-based usage model.

Distributed Information Systems:

- In a number of cases, a networked application simply consists of a server running that application (often including a database) and making it available to remote programs, called clients.
- Such clients send a request to the server for executing a specific operation, after which a response is sent back.
- Integration at the lowest level allows clients to wrap a number of requests, possibly for different servers, into a single larger request and have it executed as a distributed transaction. The key idea is that all or none of the requests are executed.
- As networked applications became more sophisticated and were gradually separated into independent components (notably distinguishing database components from processing components), it became clear that integration should also take place by letting applications communicate directly with each other.
- This has now led to a huge industry that concentrates on Enterprise Application Integration (EAI).

Distributed Transaction Processing:

- In this topic, we will concentrate on database applications. In practice, operations on a database are carried out in the form of transactions.
- A distributed transaction is a set of operations on data that is performed across two or more data repositories (especially databases).
- Programming using transactions requires special primitives that must either be supplied by the underlying distributed system or by the language runtime system.
- Typical examples of transaction primitives are shown in Fig. 4.8. The exact list of primitives depends on what kinds of objects are being used in the transaction.
- In a mail system, there might be primitives to send, receive, and forward mail. In an accounting system, they might be quite different.
- The READ and WRITE are typical examples, however. Ordinary statements, procedure calls, and so on are also allowed inside a transaction.
- In particular, Remote Procedure Calls (RPCs), that is, procedure calls to remote servers, are often also encapsulated in a transaction, leading to what is known as a transactional RPC.

Primitive	Description
BEGIN_TRANSACTION	Mark the start of a 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.

Fig. 4.8: Primitive Transaction

- The BEGIN_TRANSACTION and END_TRANSACTION are used to delimit the scope of a transaction. The operations between them from the body of the transaction.
- The characteristic feature of a transaction is either all of these operations are executed or none are executed.
- These may be system calls, library procedures, or bracketing statements in a language, depending on the implementation.
- This all-or-nothing property of transactions is one of the four characteristic properties that transactions have. More specifically, transactions adhere to the so-called ACID properties.
- The ACID properties are explained below:
  - **Atomic (A):** To the outside world, the transaction happens indivisibly.
  - **Consistent (C):** The transaction does not violate system invariants.
  - **Isolated (I):** Concurrent transactions do not interfere with each other.
  - **Durable (D):** Once a transaction commits, the changes are permanent.

- In distributed systems, transactions are often constructed as a number of sub-transactions, jointly forming a nested transaction as shown in Fig. 4.9.
- The top-level transaction may fork off children that run in parallel with one another, on different machines, to gain performance or simplify programming.
- Each of these children may also execute one or more sub-transactions, or fork off its own children.
- Nested transactions are important in distributed systems, for they provide a natural way of distributing a transaction across multiple machines. They follow a logical division of the work of the original transaction.
- For example, a transaction for planning a trip by which three different flights need to be reserved can be logically split up into three sub-transactions. Each of these sub-transactions can be managed separately and independently of the other two.

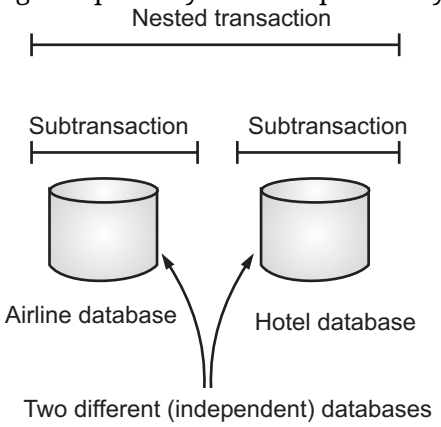


Fig. 4.9: Nested Transaction

- In the early days of enterprise middleware systems, the component that handled nested (or distributed) transactions formed the core for integrating applications at the server or database level.
- This component was called a transaction processing monitor (or TP monitor). Its main task was to allow an application to access multiple server/databases by offering it a transactional programming model, as shown in Fig. 4.10.
- Essentially, the TP monitor coordinated the commitment of sub-transactions following a standard protocol known as distributed commit.
- An important observation in Fig. 4.10 is that applications wanting to coordinate several sub-transactions into a single transaction did not have to implement this coordination themselves.
- By simply making use of a TP monitor, this coordination was done for them. This is exactly where middleware comics come into play.

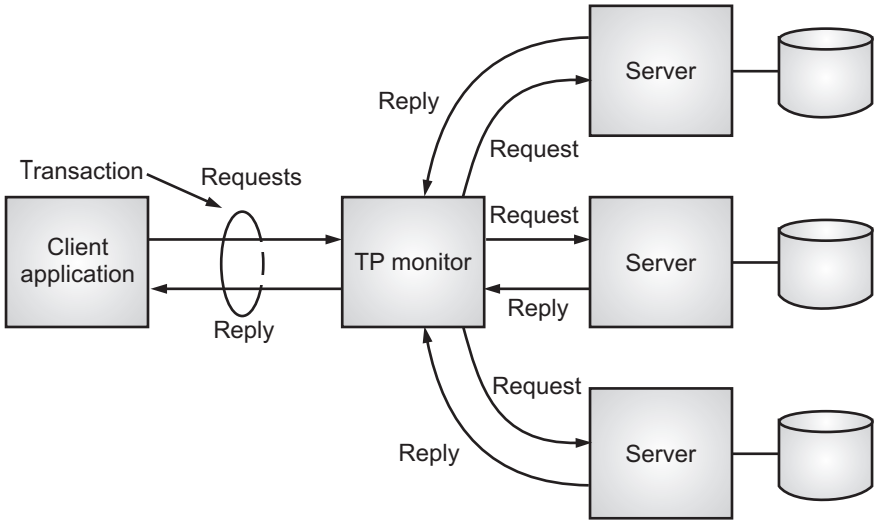


Fig. 4.10: Role of Transaction Processing (TP) Monitor in Distributed Systems

- The middleware implements services that are useful for many applications avoiding that such service have to be re-implemented over and over again by application developers.

Enterprise Application Integration:

- The more applications became decoupled from the databases they were built upon, the more evident it became that facilities were needed to integrate applications independently from their databases.
- In particular, application components should be able to communicate directly with each other and not merely by means of the request/reply behavior that was supported by transaction processing systems.
- This need for inter-application communication led to many different communication models.
- The main idea was that existing applications could directly exchange information, as shown in Fig. 4.11.
- Application integration is the process of enabling independently designed applications to work together.
- Several types of communication middleware exist in distributed systems. With RPC, an application component can effectively send a request to another application component by doing a local procedure call, which results in the request being packaged as a message and sent to the callee.
- Likewise, the result will be sent back and returned to the application as the result of the procedure call.
- As the popularity of object technology increased, techniques were developed to allow calls to remote objects, leading to what is known as Remote Method Invocations (RMI).

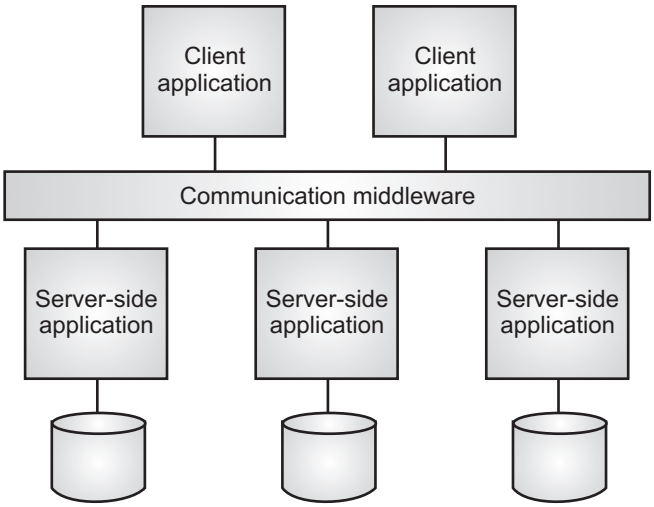


Fig. 4.11: Middleware as a Communication in Enterprise Application Integration

- An RMI is essentially the same as an RPC, except that it operates on objects instead of functions.
- RMI and RPC have the drawback that the caller and callee both need to be up and running at the time of communication. In addition, they need to know exactly how to refer to each other.
- This tight coupling is often experienced as a serious drawback, and has led to what is known as Message Oriented Middleware (MOM).
- In MOM case, applications send messages to logical contact points, often described by means of a subject.
- Likewise, applications can indicate their interest for a specific type of message, after which the communication middleware will take care that those messages are delivered to those applications.
- These so-called subscribe/publish systems form an important and expanding class of distributed systems.

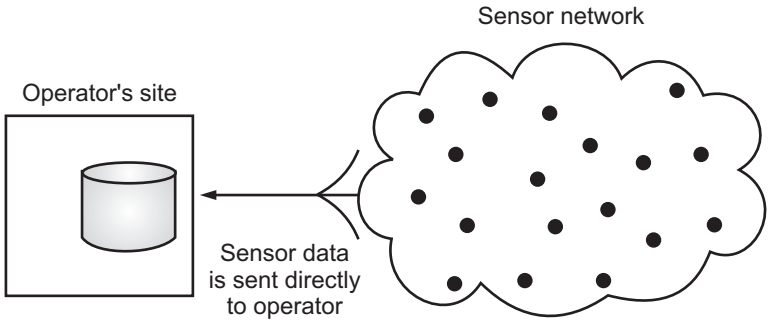
Pervasive Systems:

- The distributed systems largely characterized by their stability means nodes are fixed and have a more or less permanent and high-quality connection to a network.
- To a certain extent, this stability is realized through the various techniques for achieving distribution transparency.
- For example, there are many ways we can create the illusion that only occasionally components may fail.
- Likewise, there are all kinds of means to hide the actual network location of a node, effectively allowing users and applications to believe that nodes stay put.
- However, matters have changed since the introduction of mobile and embedded computing devices, lending to what are generally referred to as pervasive systems.

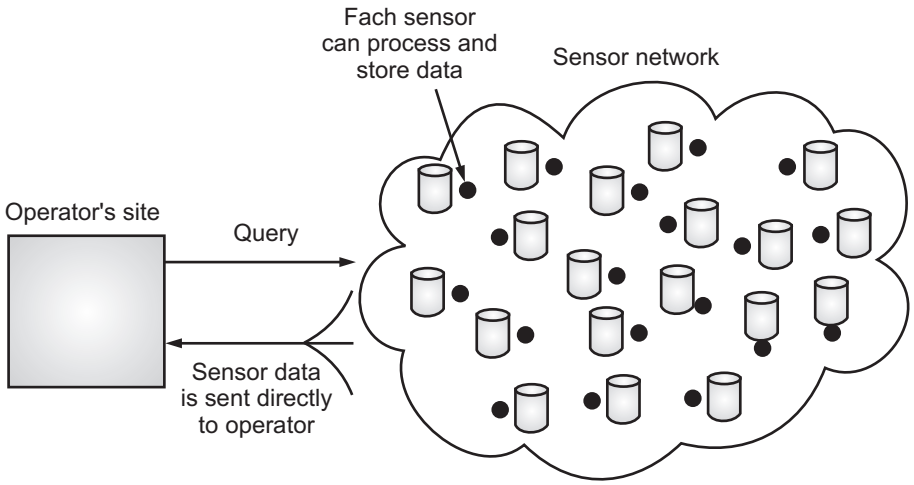
- As its name suggests, pervasive systems are intended to naturally blend into our environment. They are naturally also distributed systems, and certainly meet the characterization.
- Numbers of devices in pervasive systems are characterized by being small, battery-powered, mobile, and having only a wireless connection, although not all these characteristics apply to all devices.
- These are not necessarily restrictive characteristics, as is illustrated by smartphones and their role in what is now coined as the Internet of Things (IoT).
- An example of pervasive computing is an Apple Watch that alerts the user to a phone call and allows the call to be completed through the watch.
- The three different types of pervasive systems are sensor networks, mobile computing systems and ubiquitous computing systems.

1. Sensor Networks:

- With the rapid development of sensors, sensor networks are a vital part of the IoT (Internet of Things) and the modern world.
- Sensor networks in many cases form part of the enabling technology or pervasiveness and we see that many solutions for sensor networks return in pervasive applications.
- A sensor network consists of multiple detection stations called sensor nodes, each of which is small, lightweight and portable.
- Sensor nodes often collaborate to efficiently process the sensed data in an application-specific manner, making them very different from. For example, traditional computer networks.
- A sensor node in sensor networks monitors the data collected by the sensor and transmits this to other sensor nodes.
- A sensor network generally consists of tens to hundreds or thousands of relatively small nodes, each equipped with one or more sensing devices.
- A sensor network generally consists of tens to hundreds or thousands of relatively small nodes, each equipped with one or more sensing devices.
- Similar to other networked computer systems, additional support is needed to effectively deploy sensor network applications.
- To organize a sensor network as a distributed database, there are essentially two extremes as shown in Fig. 4.12. These two extremes are explained below:
  (i) sensors do not cooperate but simply send their data to a ·centralized database located at the operator's site.
  (ii) to forward queries to relevant sensors and to let each compute an answer, requiring the operator to aggregate the responses.



(a) Only at the Operators Site



(b) Only at the sensors

Fig. 4.12: Organization of Sensor Network Database (Storing and Processing Data)

2. Mobile Computing Systems:

- Mobility often forms an important component of pervasive systems, and many, if not all aspects that we have just discussed also apply to mobile computing.
- Mobile computing is human – computer interaction in which a computer is expected to be transported during normal usage, which allows for the transmission of data, voice and video.
- Mobile computing involves mobile communication, mobile hardware and mobile software.
  - M**obile communication** in this case, refers to the infrastructure put in place to ensure that seamless and reliable communication goes on.
  - M**obile hardware** includes mobile devices or device components such as portable laptops, smartphones, tablet Pc's, Personal Digital Assistants (PDAs) that receive or access the service of mobility.
  - M**obile software** is the actual program that runs on the mobile hardware.

- Mobile computing refers to a broad set of computing operations that allow a user to access information from portable devices such as laptop computers, PDAs, cell phones, handheld computers, music players, portable game devices, and so on.
- Mobile computing can be defined as, a computing environment of physical mobility.
- The user of a mobile computing environment will be able to access data, information or other logical objects from any device in any network while on the move.
- A mobile computing system allows a user to perform a task from anywhere using a computing device in the public (the Web), corporate (business information) and personal information spaces (medical record, address book).
- Mobile computing technology offers a quick and easy way to increase efficiency, productivity and profitability while gaining better control of the operations.
- Mobility is the biggest benefit that mobile computing devices offer. With the advent of mobile computing, it became possible for people to carry around computing devices with great capabilities.
- Mobile computing enables improvements in information accessibility. The advent of portable computers and laptops, PDA, tablets and smartphones, has in turn made mobile computing very convenient.
- The portability of these devices ensures and enables the users to access all services from any location.
- Location flexibility in mobile computing has enabled users to work from anywhere as long as there is an internet connection established.
- There are several issues that set mobile computing aside to pervasive systems as explained below:
    - The devices that form part of a (distributed) mobile system may vary widely. Typically, mobile computing is now done with devices such as smartphones and tablet computers. However, completely different types of devices are now using the Internet Protocol (IP) to communicate, placing mobile computing in a different perspective. Such devices include remote controls, pagers, active badges, car equipment, various GPS-enabled devices, and so on. A characteristic feature of all these devices is that they use wireless communication. Mobile implies wireless so it seems (although there are exceptions to the rules).
    - In mobile computing the location of a device is assumed to change over time. A changing location has its effects on many issues. For example, if the location of a device changes regularly, so will perhaps the services that are locally available. As a consequence, we may need to pay special attention to dynamically discovering services, but also letting services announce their presence. In a similar vein, we often also want to know where a device actually is. This may mean that we need to know the actual geographical coordinates of a device such as in tracking and tracing applications, but it may also require that we are able to simply detect its network position.

3. Ubiquitous Computing Systems:

- Ubiquitous computing is the new trend in computer technology, involving improvement in mobile technology and pervasive computing.
- Ubiquitous computing is an advanced computing concept where computation occurs anywhere using any computer, everywhere and in any medium.
- In contrast to desktop computing, ubiquitous computing can occur using any device, in any location, and in any format.
- A user interacts with the computer, which can exist in many different forms like laptop, tablets, terminals, smartphones, etc.
- In a ubiquitous computing system we go one step further 'the system is pervasive and continuously present'.
- The latter means that a user will be continuously interacting with the system, often not even being aware that interaction is taking place.

Characteristics of Ubiquitous Computing:

(i) Distribution: In a ubiquitous computing system devices are networked, distributed and accessible in a transparent manner. Distributed computing involves connectivity, storage and open accessibility of devices/systems.

(ii) Interaction: Interaction between users and devices in ubiquitous computing is highly unobtrusive.

(iii) Context Awareness: The system is aware of a user's context in order to optimize interaction. Context awareness which means that devices need to be informed of the ecological context to improve the system operation.

(iv) Autonomy: Devices in ubiquitous computing operate autonomously without human intervention and are thus highly self-managed.

(v) Intelligence: The system as a whole can handle a wide range of dynamic actions and interactions.

- Ubiquitous computing, is the growing trend of embedding computational capability into everyday objects to make them effectively communicate and perform useful tasks in a way that minimizes the end user's need to interact with computers as computers.
- Ubiquitous computing can be located in a sense and accessible to clients when appropriate, rather than rendering computational resources open to all computers worldwide.
- Data privacy is an important requirement for safeguarding privacy in ubiquitous computing.

4.3 ARCHITECTURAL STYLES

- The organization of distributed systems is mostly about the software components that constitute the system.
- These software architectures tell us how the various software components are to be organized and how they should interact.
- In this section we will see architectural styles toward organizing (distributed) computer systems.
- An architectural style is formulated in terms of components, it is the way in which these components are connected together and how data is exchanged between the components.
  - The **components** are modular units with well-defined interfaces. In addition, they have characteristics of being replaceable and reusable.
  - The **connectors** are communication links between modules that coordinate between components.
- Therefore, the idea behind distributed architectures is to have components presented in different ways, where the components can communicate with each other over a network.
- The basic idea of architectural style is to organize logically different components, and distribute those computers over the various machines.
- Important styles of architecture for distributed systems include Layered Architecture, Object-based Architecture and Resource-centered architecture.

4.3.1 Layered Architecture

- In the layered architecture style of distributed systems, components are organized in layers.
- Each layer in layered architecture style uses the previous layer to implement new functionality that is exported to the layer above.
- The layered architecture separates layers of components from each other, giving it a much more modular approach.
- In layered architecture each interaction is sequential where a layer will contact the adjacent layer and this process continues, until the request is catered to.
- The layers on the bottom provide a service to the layers on the top. The request flows from top to bottom, whereas the response is sent from bottom to top.
- The advantage of using a layered approach is that the calls always follow a predefined path and that each layer can be easily replaced or modified without affecting the entire architecture.

- The basic idea of a layered architecture style is shown in Fig. 4.13.

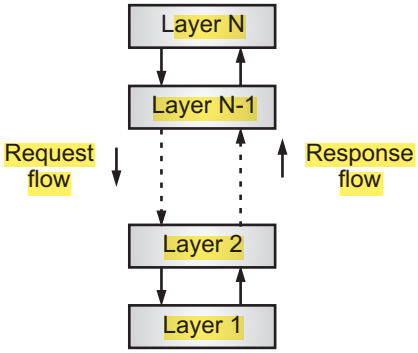


Fig. 4.13: Layered Architecture Style

- The basic idea in layered architecture style is simple components are organized in a layered format where a component in a layer can do a downcall of a lower-layer component and usually waits for response. Only in exceptional cases a component can do an upcall to a higher level component'.
- The three common cases are shown in Fig. 4.14. Fig. 4.14 (a) shows a standard organization in which only downcalls to the next lower layer are made. This organization is commonly deployed in the case of network communication.

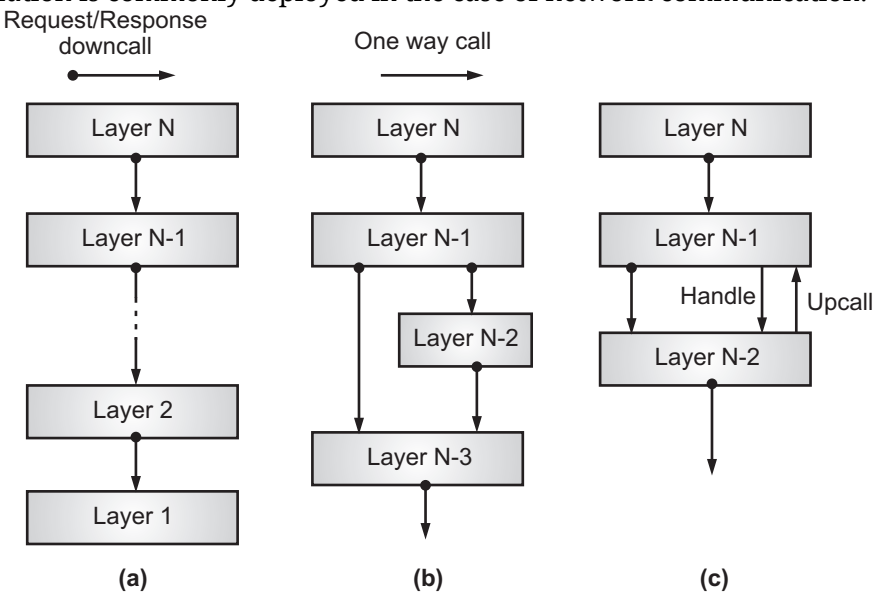


Fig. 4.14: (a) Pure Layered Organization (b) Mixed Layered Organization (c) Layered Organization with Upcalls

- In a number of situations we also encounter the organization shown in Fig. 4.14 (b). For example, consider an application A that makes use of a library L_{os} to interface to an operating system. At the same time, the application uses a specialized mathematical library L_{math} that has been implemented by also making use of L_{os}.

- In this case, referring to Fig. 4.14 (b), A is implemented at layer N – 1, L_{math} at layer N - 2 and L_{os} which is common to both of them, at layer N – 3.
- A special situation is shown in Fig. 4.14 (c). In some cases, it is convenient to have a lower layer do an upcall to its next higher layer.
- A typical example is when an operating system signals the occurrence of an event, to which end it calls a user-defined operation for which an application had previously passed a reference (typically referred to as a handle).

Layered Communication Protocol:

- A well-known and ubiquitously applied layered architecture of distributed systems is that of so-called communication protocol stack.
- In communication-protocol stacks, each layer implements one or several communication services allowing data to be sent from a destination to one or several targets. To this end, each layer offers an interface specifying the functions that can be called.
- In principle, the interface should completely hide the actual implementation of a service.
- Another important concept in the rise of communication is that of a (communication) protocol, which describes the rules that parties will follow in order to exchange information.
- It is important to understand the difference between a service offered by a layer, the interface by which that service is made available, and the protocol that a layer implements to establish communication.

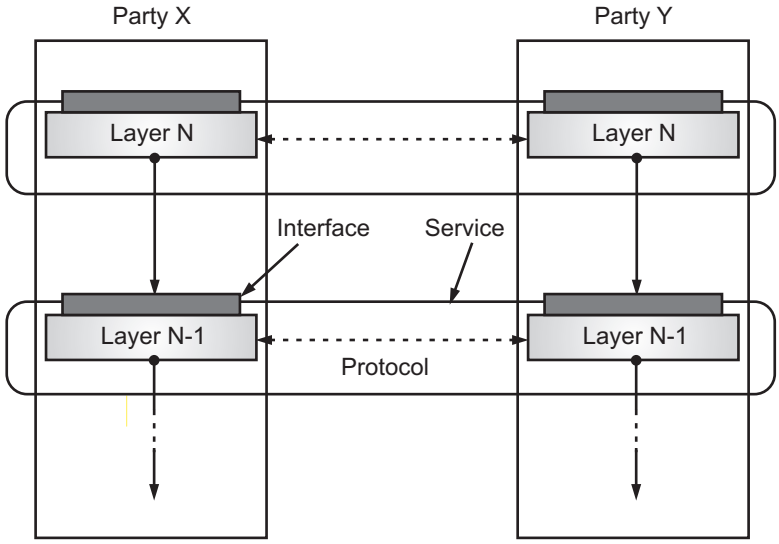


Fig. 4.15: A Layered Communication-Protocol Stack showing the difference between a Service, its Interface, and the Protocol it Deploys

- Fig 4.15 shows this distinction. To make this distinction clear, consider a reliable, connection-oriented service, which is provided by many communication systems.
- In this case, a communicating party first needs to set up a connection to another party before the two can send and receive messages.
- Being reliable means that strong guarantees will be given that sent messages will indeed be delivered to the other side, even when there is a high risk that messages may be lost.
- In addition such services generally also ensure that messages are delivered in the same order as that they were sent. This kind of service is realized on the Internet by means of the TCP (Transmission Control Protocol).
- The TCP protocol specifies which messages are to be exchanged for setting up or tearing down a connection, what needs to be done to preserve the ordering of transferred data and what both parties need to do to detect and correct data that was lost during transmission.
- The service is made available in the form of a relatively simple programming interface containing calls to set up a connection, send and receive messages and to tear down the connection again.
- In fact, there are different interfaces available, often dependent on the operating system or programming language used. Likewise, there are many different implementations of the protocol and its interfaces.

Application Layering:

- In this topic we focus on the logical layering of applications. Considering that a large class of distributed applications is targeted ·toward supporting user or application access to databases, many people have advocated a distinction between three logical levels, essentially following a layered architecture style:
  1. The application interface level,
  2. The processing level and
  3. The data level.
- In line with this layering architecture, we see that applications can often be constructed from roughly following three different pieces/parts:
  - a part that handles interaction with a user or some external application.
  - a part that operates on a database or file system.
  - a middle part that generally contains the core functionality of the application. This middle part is logically placed at the processing level.
- In contrast to user interfaces and databases, there are not many aspects common to the processing level. Therefore, we shall give a number of examples to make this level clearer.

- Consider an example, Internet search engine (See Fig. 4.16). Ignoring all the animated banners, images and other fancy window dressing, the user interface of a search engine can be very simple, a user types in a string of keywords and is subsequently presented with a list of files of Web pages. The backend is formed by a huge database of Web pages that have been pre-fetched and indexed. The core of the search engine is a program that transforms the user's string of keywords into one or more database queries.
- It subsequently ranks the results into a list and transforms that list into a series of HTML pages and this information retrieval part is typically placed at the processing level.

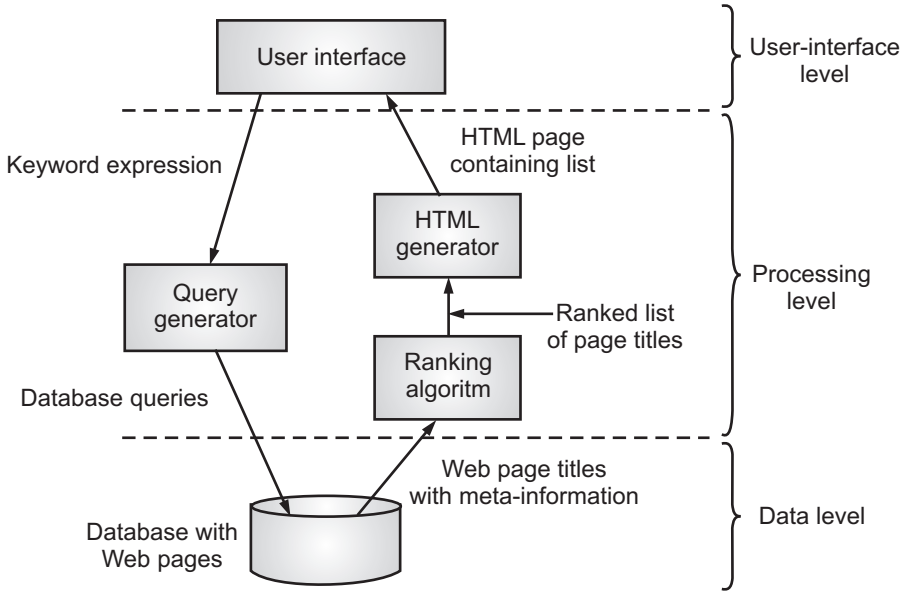


Fig. 4.16: Simplified Organization of an Internet Search Engine into Three different Layers

4.3.2 Object-based Architecture

- This architecture style is based on loosely coupled arrangement of objects. This has no specific architecture like layers.
- Like in layers, this does not have a sequential set of steps that needs to be carried out for a given call.
- Each of the components is referred to as objects, where each object can interact with other objects through a given connector or interface.
- These are much more direct where all the different components can interact directly with other components through a direct method call.
- In essence, each object corresponds to what we have defined as a component and these components are connected through procedure call mechanism.

- In the case of distributed systems, .a procedure call can also take place over a network, i.e., the calling object need not be executed on the same ·machine as the called object.
- As shown in the Fig. 4.17, communication between objects happens as method invocations. These are generally called Remote Procedure Calls (RPC).
- Some popular examples are Java RMI and Web Services.

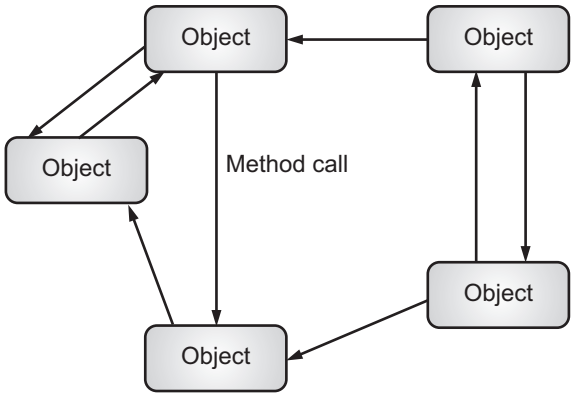


Fig. 4.17: Object Based Architecture Style

- Object-based architecture styles are attractive because they provide a natural way of encapsulating data (called an object's state) and the operations that can be performed on that data (which are referred to as an object's methods) into a single entity.
- The interface offered by an object conceals implementation details, essentially meaning that we, in principle, can consider an object completely independent of its environment.
- As with components, this also means that if the interface is clearly defined and left otherwise untouched, an object should be replaceable with one having exactly the same interface.
- This separation between interfaces and the objects implementing these interfaces allows us to place an interface at one machine, while the object itself resides on another machine.
- This organization, which is shown in Fig. 4.18, is commonly referred to as a distributed object.
- A characteristic, but somewhat counterintuitive feature of most distributed objects is that their state is not distributed means it resides at a single machine.
- Only the interfaces implemented by the object are made available on other machines. Such objects are also referred to as remote objects.
- In a general distributed object, the state itself may be physically distributed across multiple machines, but this distribution is also hidden from clients behind the object's interfaces.

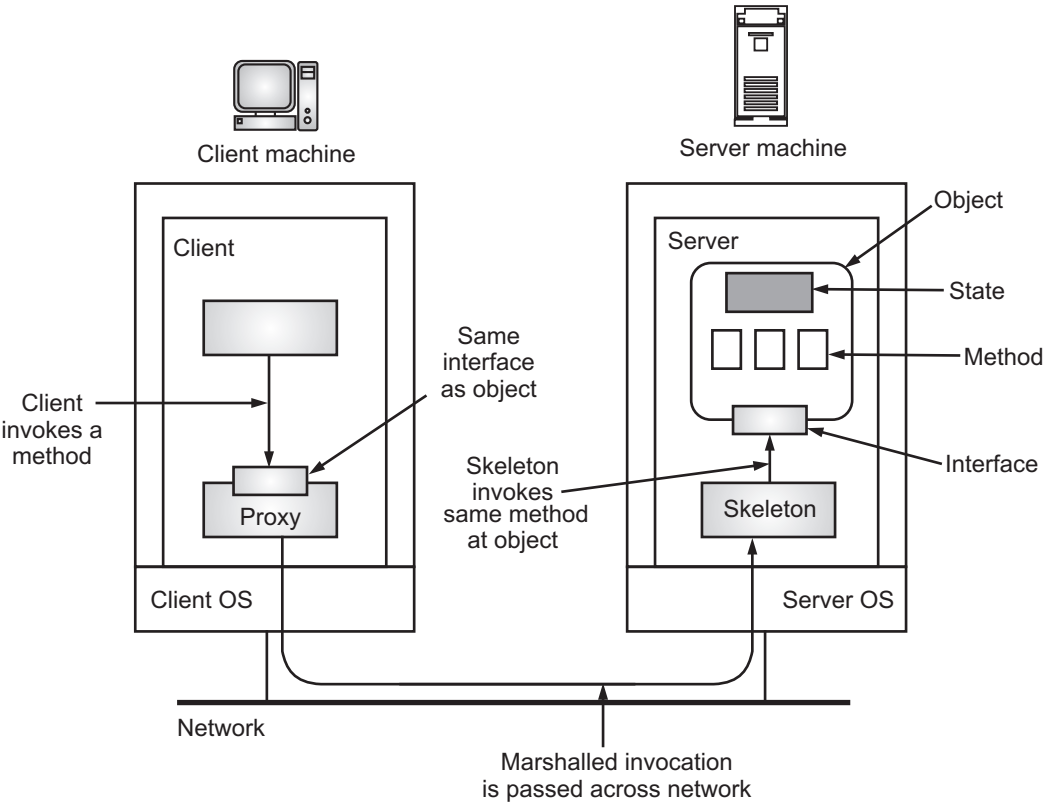


Fig. 4.18: Organization of a remote Object with Client-side Proxy

- When a client binds to a distributed object, an implementation of the object's interface, called a proxy is then loaded into the client's address space.
- A proxy is analogous to a client stub in RPC systems. The only thing it does is marshal method invocations into messages and un-marshal reply messages to return the result of the method invocation to the client.
- The actual object resides at a server machine, where it offers the same interface as it does on the client machine.
- Incoming invocation requests are first passed to a server stub, which un-marshals them to make ·method invocations at the object's interface at the server.
- The server stub is also responsible for marshaling replies and forwarding reply messages lo the client-side proxy.
- The server-side stub is often referred to as a skeleton as it provides the bare means for letting the server middleware access the user-defined objects.
- In practice, it often contains incomplete code in the form of a language-specific class that needs to be further specialized by the developer.
- The object-based architectures form the foundation of encapsulating services into independent units.

- Encapsulation is the keyword here (the service as a whole is realized as a self-contained entity, although it can possibly make use of other services).
- By clearly separating various services such that they can operate independently, we are paving the road toward Service-Oriented Architectures (SOAs).
- In a SOA, a distributed application or system is essentially constructed as a composition of many different services.
- Not all of these services ·may belong to the same administrative organization. We already came across this phenomenon when discussing cloud computing (it may very well be that an organization running its business application makes use of storage services offered by a cloud provider.
- These storage services are logically completely encapsulated into a single unit, of which an interface is made available to customers).

4.3.3 Resource-centered Architecture

- One of the problems with service composition is that connecting multiple components can easily lead to an integration nightmare.
- Alternatively, someone can view a distributed system as a large set of features that can be individually managed by components.
- Resources can be added or removed by applications, and can also be retrieved or modified.
- This approach has been widely adopted on the Web and is known as REST. RESTful architecture has become popular due to its simplicity.
- As an alternative, one can also view a distributed system as a huge collection of resources that are individually managed by components.
- Resources may be added or removed by (remote) applications, and likewise can be retrieved or modified. This approach has now been widely adopted for the Web and is known as REpresentational State Transfer (REST).
- There are following four key characteristics of what are known as RESTful architectures:
  1. Resources are identified through a single naming scheme.
  2. All services offer the same interface, consisting of at most four operations, as shown in Table 4.1.
  3. Messages sent to or from a service are fully self-described.
  4. After executing an operation at a service, that component forgets everything about the caller.

Table 4.1: The Four Operations available is RESTful Architectures

Operation	Description
PUT	Create a new resource.
GET	Retrieve the state of a resource in some representation.
DELETE	Delete a source.
POST	Modify a resource by transferring a new state

- As an increasing number of services became available over the Web and the development of distributed systems through service composition became more important, researchers started to rethink the architecture of mostly Web-based distributed systems.
- One of the problems with service composition is that connecting various components can easily turn into an integration nightmare.
- RESTful architecture has become popular because of its simplicity. However, holy wars are being fought over whether RESTful services are better than where services are specified by means of service-specific interfaces.
- In particular, the simplicity of RESTful architectures can easily prohibit easy solutions to intricate communication schemes.
- One example is where distributed transactions are needed, which generally requires that services keep track of the state of execution.

4.4 SYSTEM ARCHITECTURE

- A distributed system is a system whose components are located on different networked computers, which communicate and coordinate their actions by passing messages to one another from any system.
- Now that we have briefly discussed some commonly applied architectural styles, let us take a look at how many distributed systems are actually organized by considering· where software components are placed.
- Deciding on software components, their interaction and their placement leads to an instance of a software architecture also known as a system architecture.
- In this section, we will discuss centralized and decentralized organizations, as well as various hybrid forms.
- The nodes in the distributed systems can be arranged in the form of client/server systems or peer to peer systems.
- The two major system level architectures that we use today are Client-Server and Peer-to-Peer (P2P).

Client/Server Distributed Systems:

- The client/server architecture is a prerequisite to the proper development of client/server systems.

- The client/server architecture is based on hardware and software components that interact to form a distributed system.
- In client/server systems, the client requests a resource and the server provides that resource.
- A server may serve multiple clients at the same time while a client is in contact with only one server.
- Both the client and server usually communicate via a computer network and so they are a part of distributed systems.
- The client/server systems that partitions tasks or workloads between the providers of a resource or service, called servers and service requesters, called clients.
- The client/server architecture has two major components namely, the client and the server.
- The server is where all the processing, computing and data handling is happening, whereas the client is where the user can access the services and resources given by the server (remote server).
- The clients can make requests from the server and the server will respond accordingly.

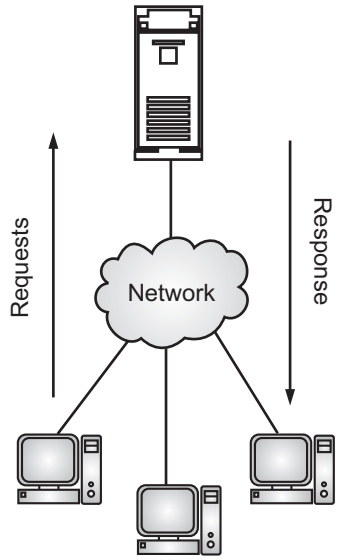


Fig. 4.19: Client/Server System

Advantages of Client/Server Distributed Systems:

1. A client/server system provides better performance at a reduced cost for hardware and software than alternative mini or mainframe solutions.
2. A client/server system has the ability to distribute the computing workload between client workstations and shared servers.
3. A client/server system allows the end user to use a microcomputer's graphical user interfaces, thereby improving functionality and simplicity.

Disadvantages Client/Server Distributed Systems:

1. A client/server system may suffer from security problems as the number of users and processing sites increases.
2. The maintenance cost of a client/server system is greater than that of an alternative mini or mainframe solution.
3. In a client/server system, the operating system software is distributed over many machines rather than a single system, thereby increasing complexity.

Peer to Peer Distributed Systems:

- The peer-to-peer architecture is a good way to structure a distributed system so that it consists of many identical software processes or modules, each module running on a different computer or node.
- The different software modules stored at different sites communicate with each other to complete the processing required for the execution of distributed applications.
- Peer-to-peer architecture provides both client and server functionalities on each computer. Therefore, each node can access services from other nodes as well as providing services to other nodes.
- In contrast with the client/server architecture, in a peer-to-peer distributed system each node provides user interaction facilities as well as processing capabilities.
- Peer-to-peer (P2P) computing or networking is a distributed application architecture that partitions tasks or workloads between peers.
- Peers are equally privileged, equipotent participants in the application. They are said to form a peer-to-peer network of nodes.
- The peer-to-peer systems contain nodes that are equal participants in data or resource sharing. All the tasks are equally divided between all the nodes (also called peers).
- The nodes interact with each other as required to share resources. This is done with the help of a network.
- Peers make a portion of their resources, such as processing power, disk storage or network bandwidth, directly available to other network participants, without the need for central coordination by servers or stable hosts.
- Peers are both suppliers and consumers of resources, in contrast to the traditional client–server model in which the consumption and supply of resources is divided.

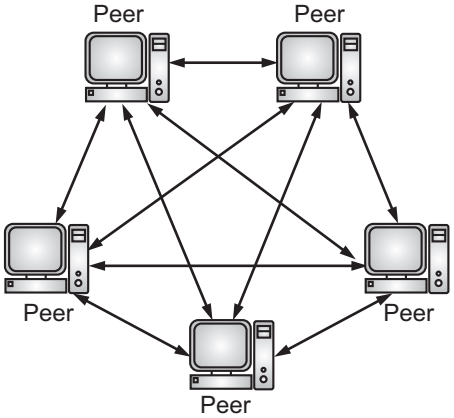


Fig. 4.20: P2P System

4.4.1 Centralized Organizations

- Despite the lack of consensus on many distributed systems issues, there is one issue that many researchers agree upon, 'thinking in terms of clients that request services from servers' helps understanding and managing the complexity of distributed systems'.

Simple Client/Server Architecture:

- In the basic client/server model processes in a distributed system are divided into two (possibly overlapping) groups namely, server and client.
  - A server is a process implementing a specific service, for example, a file system service or a database service.
  - A client is a process that requests a service from a server by sending it a request and subsequently waiting for the server's reply.
- This client-server interaction, also known as request-reply behavior, is shown in the Fig. 4.21 in the form of a message sequence chart.
- Communication between a client and a server can be implemented by means of a simple connectionless protocol when the underlying network is fairly reliable as in many Local Area Networks (LANs).
- When a client requests a service, it simply packages a message for the server, identifying the service it wants, along with the necessary input data.
- The message is then sent to the server. The latter, in turns, will always wait for an incoming request, subsequently process it, and package the results in a reply message that is then sent to the client.
- Using a connectionless protocol has the obvious advantage of being efficient. As long as messages do not get lost or corrupted, the request/reply protocol just sketched works fine.

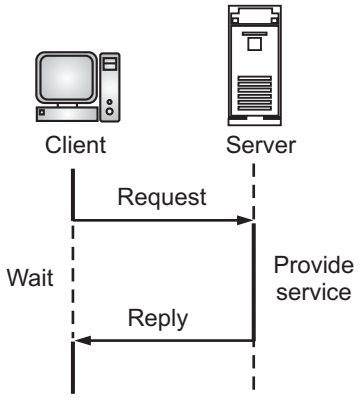


Fig. 4.21: Interaction between a Client and a Server

- As an alternative, many client/server systems use a reliable connection-oriented protocol.

- Although this solution is not entirely appropriate in a LAN due to relatively poor performance, it works perfectly fine in wide-area systems/networks i.e., WAN in which communication is inherently unreliable.

Multi-tiered Client/Server Architectures:

- The three logical levels suggest a number of possibilities for physically distributing a client/server application across several machines.
- The simplest organization is to have only two types of machines i.e. a client machine (containing only the programs implementing (part of) the user-interface level) and server machine (containing the rest, that is, the programs implementing the processing and data level).
- In simple client/server architecture everything is handled by the server while the client is essentially no more than a dumb terminal, possibly with only a convenient graphical interface.
- There are, however, many other possibilities. Many distributed applications are divided into the three layers namely, user interface layer, processing layer and data layer.
- One approach for organizing clients and servers is then to distribute these layers across different machines as shown in Fig. 4.22.
- As a first step, we make a distinction between client machines and server machines, leading to what is also referred to as a (physically) two-tiered architecture.
- Fig. 4.22 (a) shows one0 possible organization is to have only the terminal-dependent part of the user interface on the client machine and give the applications remote control over the presentation of their data.
- Fig. 4.22 (b) shows an alternative is to place the entire user-interface software on the client side.
- In such cases, we essentially divide the application into a graphical front end, which communicates with the rest of the application (residing at the server) through an application-specific protocol.
- In this model, the front end (the client software) does no processing other than necessary for presenting the application's interface.
- We may also move part of the application to the front end as shown in Fig. 4.22 (c). An example where this makes sense is where the application makes use of a form that needs to be filled in entirely before it can be processed.
- The front end can then check the correctness and consistency of the form and where necessary interact with the user.
- Another example of the organization of Fig. 4.22 (c), is that of a word processor in which the basic editing functions execute on the client side where 'they operate on locally cached or memory data, but where the advanced support· tools such as checking the spelling and grammar execute on the server side.

- The client/server environments, the organizations in Fig. 4.22 (d) and Fig. 4.22 (e) are particularly popular.
- These organizations are used where the client machine is a PC or workstation connected through a network to a distributed file system or database.
- Number of the application is running on the client machine, but all operations on files or database entries go to the server.
- For example, number of banking applications run on an end-user's machine where the user prepares transactions and such. Once finished, the application contacts the database on the bank's server and uploads the transactions for further processing.

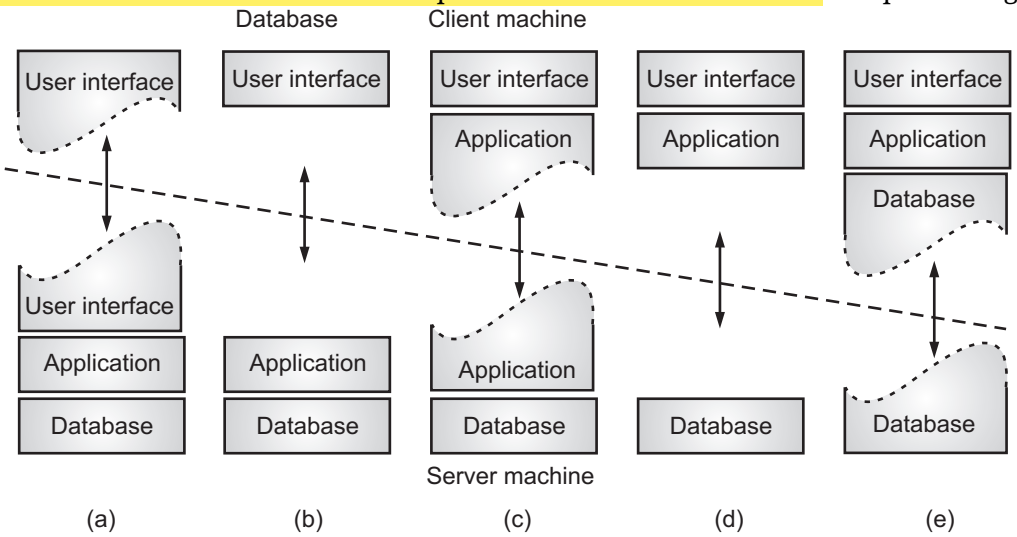


Fig. 4.22: Client/Server Organizations in Two-Tiered Architecture

- Fig. 4.23 shows three-tiered architecture. The three-tier architecture (also referred to as a multi-tier architecture) emerged to overcome the limitations of the two-tier architecture.
- In the three-tier architecture, a middle tier was added between the user system interface client environment and the database management server environment.
- In three-tier architecture, traditionally programs that form part of the processing layer are executed by a separate server, but may additionally be partly distributed across the client and server machines.
- A typical example or when a three-tiered architecture is used is in transaction processing.
- A separate process, called the transaction processing monitor, coordinates all transactions across possibly different data servers.
- Another, example of a three-tiered architecture is in the organization of Web sites. In this case, a Web server acts as an entry point to a site, passing requests to an application server where the actual processing takes place. This application server, in turn, interacts with a database server.

- For example, an application server may be responsible for running the code to inspect the available inventory of some goods as offered by an electronic bookstore. To do so, it may need to interact with a database containing raw inventory data.

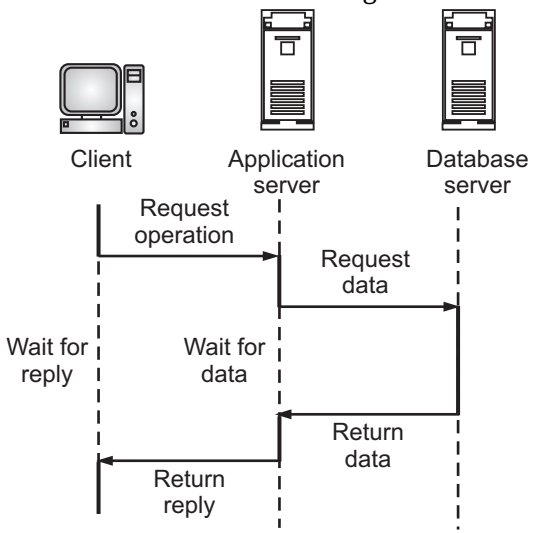


Fig. 4.23: Three-Tiered Architecture (a Server acting as Client)

4.4.2 Decentralized Organizations: Peer-to-Peer Systems

- Multi-tiered client/server architectures are a direct consequence of dividing distributed applications into a user interface, processing components and data management components.
- In multi-tiered client/server architecture the different tiers correspond directly with the logical organization of applications.
- In a number of business environments, distributed processing is equivalent to organizing a client/server application as a multi-tiered architecture. We refer to this type of distribution as vertical distribution.
- The characteristic feature of vertical distribution is that it is achieved by placing logically different components on different machines.
- The term is related to the concept of vertical fragmentation as used in distributed relational databases, where it means that tables are split column wise and subsequently distributed across multiple machines.
- A vertical distribution can help (functions are logically and physically split across multiple machines, where each machine is tailored to a specific group of functions).
- However, vertical distribution is only one way of organizing client/server applications. In modern architectures, it is often the distribution of the clients and the servers that counts, which we refer to as horizontal distribution.

- In horizontal distribution a client or server may be physically split up into logically equivalent parts, but each part is operating on its own share of the complete data set, thus balancing the load.
- In this section we will take a look at a class of modern system architectures that support horizontal distribution known as peer-to-peer systems.
- From a high-level perspective, the processes that constitute a peer-to-peer system are all equal means that the functions that need to be carried out are represented by every process that constitutes the distributed system.
- As a consequence, much of the interaction between processes is symmetric (each process will act as a client and a server at the same time (which is also referred to as acting as a servant)).
- The peer-to-peer architectures evolve around the question of how to organize the processes in an overlay network.
- An overlay network is a network in which the nodes are formed by the processes and the links represent the possible communication channels (which are often realized as TCP connections).
- A node may not be able to communicate directly with an arbitrary other node, but is required to send messages through the available communication channels. Two types of overlay networks exist namely, structured and unstructured.

Structured Peer-to-Peer Systems:

- In a structured peer-to-peer system the nodes (i.e., processes) are organized in an overlay that adheres to a specific, deterministic topology (a ring, a binary tree, a grid, etc.). This topology is used to efficiently look up data.
- Characteristic for structured peer-to-peer systems is that they are generally based on using a so-called semantic-free index.
- What this means is that each data item that is to be maintained by the system is uniquely associated with a key, and that this key is subsequently used as an index.

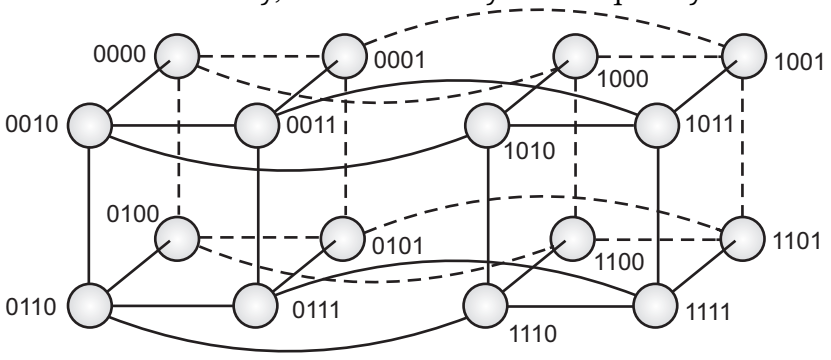


Fig. 4.24: A simple Peer-to-Peer System Organized as a Four-dimensional Hypercube

- In general, the nodes in a structured overlay network are formed in a logical ring, with nodes being connected to this ring. In this ring, certain nodes are responsible for certain services.

- A common approach that can be used to tackle the coordination between nodes is to use Distributed Hash Tables (DHTs).
- A traditional hash function converts a unique key into a hash value that will represent an object in the network. The hash function value is used to insert an object in the hash table and to retrieve it.
- In a DHT, each key is assigned to a unique hash, where the random hash value needs to be of a very large address space, in order to ensure uniqueness.
- A mapping function is being used to assign objects to nodes based on the hash function value.
- A lookup based on the hash function value, returns the network address of the node that stores the requested object.
  1. **Hash Function**: Takes a key and produces a unique hash value.
  2. **Mapping Function**: Map the hash value to a specific node in the system.
  3. **Lookup Table**: Return the network address of the node represented by the unique hash value.

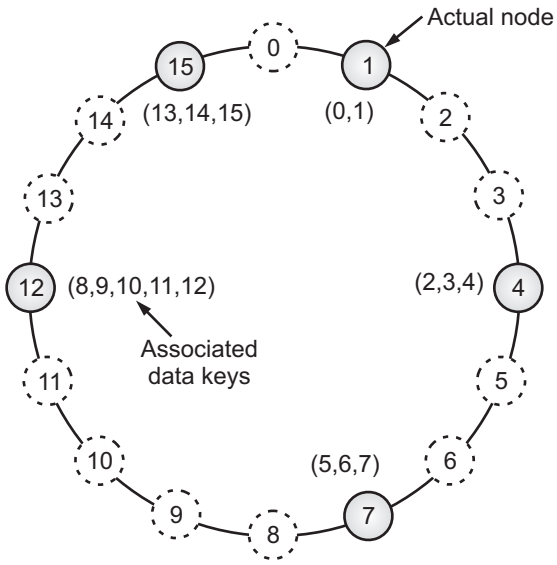


Fig. 4.25

Unstructured Peer-to-Peer Systems:

- Structured peer-to-peer systems attempt to maintain a specific, deterministic overlay network. An unstructured peer-to-peer system each node maintains an ad hoc list of neighbors.
- The resulting overlay resembles what is known as a random graph. A random graph in which an edge (u, v) between two nodes u and v exists only with a certain probability P[(u, v)].
- Ideally, this probability is the same for all pairs of nodes, but in practice a wide range of distributions is observed.

- In an unstructured peer-to-peer system, when a node joins it often contacts a well-known node to obtain a starting list of other peers in the system.
- This list can then be used to find more peers, and perhaps ignore others, and so on. In practice, a node generally changes its local list almost continuously.
- For example, a node may discover that a neighbor is no longer responsive and that it needs to be replaced.

Hierarchically Organized Peer-to-Peer Networks:

- In unstructured peer-to-peer systems, locating relevant data items can become problematic as the network grows this scalability problem.
- There are other situations in which abandoning the symmetric nature of peer-to-peer systems is sensible. Consider a collaboration of nodes that offer resources to each other.
- For example, in a collaborative Content Delivery Network (CON), nodes may offer storage for hosting copies of Web documents allowing Web clients to access pages nearby and thus to access them quickly.
- What is needed is a means to find out where documents can be stored best. In that case, making use of a broker that collects data on resource usage and availability for a number of nodes that are in each other's proximity will allow users to quickly select a node with sufficient resources.
- Nodes such as those maintaining an index or acting as a broker are generally referred to as super peers.
- As the name suggests, super peers are often also organized in a peer-to-peer network, leading to a hierarchical organization.
- A simple example of such an organization is shown in Fig. 4.26.
- In this organization, every regular peer, now referred to as a weak peer, is connected as a client to a super peer. All communication from and to a weak peer proceeds through that peer's associated super peer.
- In a number of cases, the association between a weak peer and its super peer is fixed (whenever a weak peer joins the network, it attaches to one of the super peers and remains attached until it leaves the network).

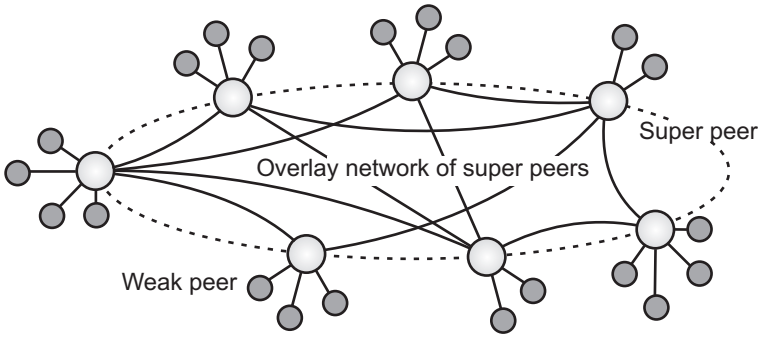


Fig. 4.26: A Hierarchical Organization of Nodes into a Super-peer Network

- The super peers are long-lived processes with high availability. To compensate for potential unstable behavior of a super peer, backup schemes can be deployed, such as pairing every super peer with another one and requiring weak peers to attach to both.
- The peer-to-peer networks offer a flexible means for nodes to join and leave the network.
- However, with super-peer networks a new problem is introduced, namely how to select the nodes that are eligible to become super peers. This problem is closely related to the leader-election problem.

4.4.3 Hybrid Architectures

- In previous sections we have focused on client/server architectures and a number of peer-to-peer architectures. Many distributed systems combine architectural features as we already came across in super-peer networks.
- Some specific classes of distributed systems in which client/server solutions are combined with decentralized architectures includes collaborative distributed systems and edge-server systems.

Collaborative Distributed Systems:

- Hybrid architectures are notably deployed in collaborative distributed systems. The main issue in many of these systems is to first get started, for which often a traditional client/server scheme is deployed.
- Once a node has joined the system, it can use a fully decentralized scheme for collaboration. To make matters concrete, let us consider the widely popular BitTorrent file-sharing system.
- BitTorrent is a peer'-to-peer file downloading system and its principal working is shown in Fig. 4.27.
- The basic idea is that when an end user is looking for a file, they download chunks of the file from other users until the downloaded chunks can be assembled together yielding the complete file. An important design goal was to ensure collaboration.
- To download a file, a user needs to access a global directory, which is generally just one of a few well-known Web sites. Such a directory contains references to what are called torrent files.
- A torrent file contains the information that is needed to download a specific file. In particular, it contains a link to what is known as a tracker, which is a server that is keeping an accurate account of active nodes (chunk of) the requested file.
- An active node is one that is currently downloading the file as well. Obviously, there will be many different trackers, although there will generally be only a single tracker per file (or collection of files).
- Once the nodes have been identified from where chunks to be downloaded, the downloading node effectively become active.

- At that point, it will be forced to help others, for example by providing chunks of the file it is downloading that others do not yet have.
- Clearly, BitTorrent combines centralized with de-centralized solutions. As it turns out, the bottleneck of the system is, not surprisingly, formed by the trackers.
- In an alternative implementation of BitTorrent, a node also joins a separate structured peer-to-peer system (i.e., a DHT) to assist in tracking file downloads.
- In effect, a central tracker's load is now distributed across the participating nodes, with each me acting as a tracker for a relatively small set of torrent files.
- The original function of the tracker coordinating the collaborative downloading of a file is retained.
- However, we note that in many BitTorrent systems used today, the tracking functionality has actually been minimized to a one-time provisioning of peers currently involved in downloading the file.

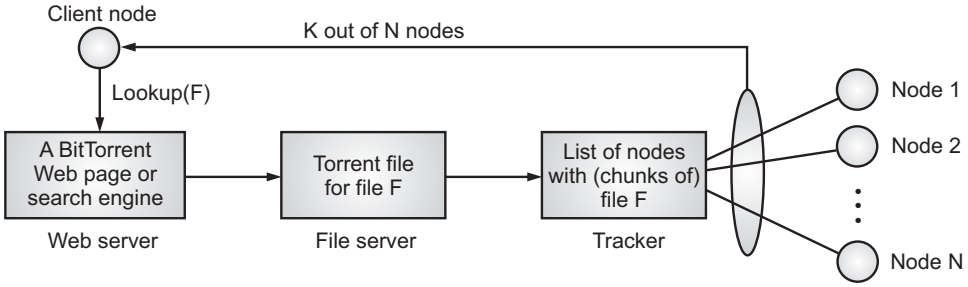


Fig. 4.27: Working Principle of BitTorrent

Edge-Server Systems:

- An important type of distributed systems that is organized according to a hybrid architecture is formed by edge-server systems.
- The edge-server systems are deployed on the Internet where servers are placed "at the edge" of the network.
- This edge is formed by the boundary between enterprise networks and the actual Internet, for example, as provided by an Internet Service Provider (ISP).
- Likewise, where end users at home connect to the Internet through their ISP, the ISP can be considered as residing at the edge of the Internet. This leads to a general organization like the one shown m Fig. 4.28.
- End users (or clients) in general, connect to the Internet by means of an edge server. The edge server's main purpose is to serve content, possibly after applying filtering and trans-coding functions.
- More interesting is the fact that a collection of edge servers can be used to optimize content and application distribution.
- The basic model is that for a specific organization, one edge server acts as an origin server from which all content originates. That server can use other edge servers for replicating Web pages.

- This concept of edge-server systems is now often taken a step further: taking cloud computing as implemented in a data center as the core, additional servers at the edge of the network are used to assist in computations and storage, essentially leading to distributed cloud systems.
- In the case of fog computing, even end-user devices form part of the system and are (partly) controlled by a cloud-service provider.

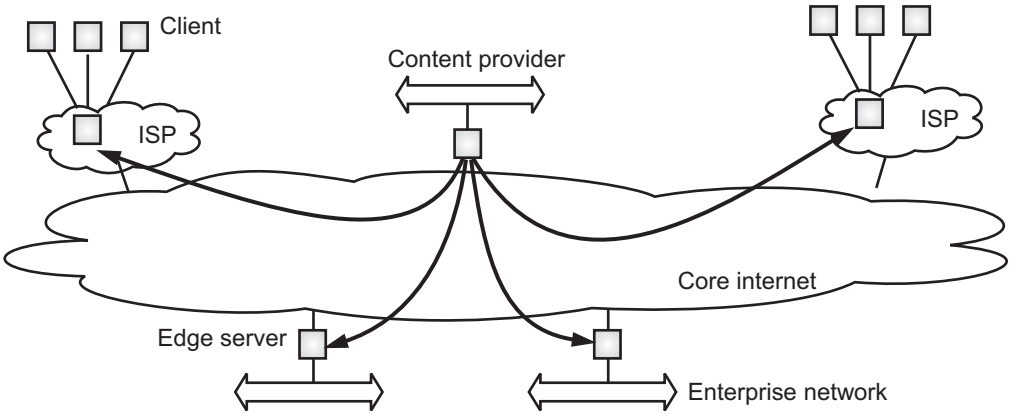


Fig. 4.28: Viewing the Internet as consisting of a Collection of Edge Servers

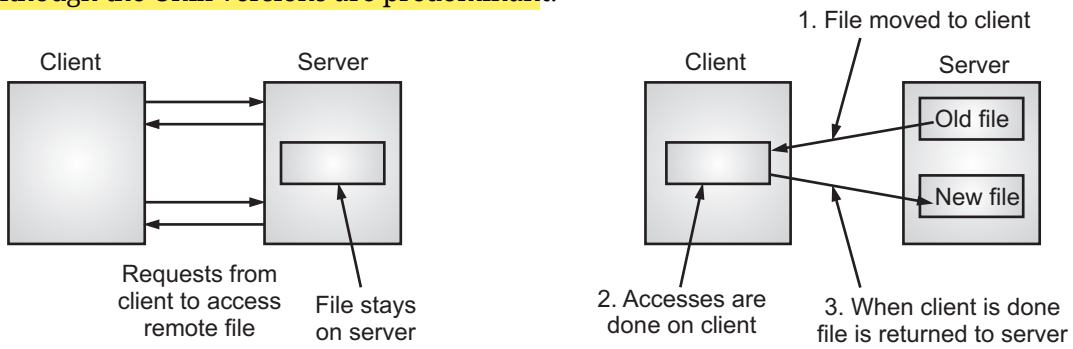
4.5 EXAMPLES ARCHITECTURES

- In this section we will study example architectures namely, Network File System (NFS) and Web-based distributed systems.

4.5.1 Network File System (NFS)

- Numbers of distributed files systems are organized along with client/server architectures, with Sun Microsystems's Network File System (NFS) being one of the most widely-deployed ones for Unix systems.
- NFS is a distributed file system protocol originally developed by Sun Microsystems in 1984, allowing a user on a client computer to access files over a computer network much like local storage is accessed.
- Here, we concentrate on NFSv3, the widely-used third version of NFS and NFSv4, the most recent, fourth version.
- The NFS is architecture of the client/server, which contains a client program, server program, and a protocol that helps for communication between the client and server.
- The basic idea behind NFS is that each file server provides a standardized view of its local file system.
- In other words, it should not matter how that local file system is implemented; each NFS server supports the same model. This approach has been adopted for other distributed files systems as well.

- NFS comes with a communication protocol that allows clients to access the files stored on a server, thus allowing a heterogeneous collection of processes, possibly running on different operating systems and machines, to share a common file system.
- The model underlying NFS and similar systems is that of a remote file service. In this model, clients are offered transparent access to a file system that is managed by a remote server.
- However, clients are normally unaware of the actual location of files. Instead, they are offered an interface to a file system that is similar to the interface offered by a conventional fecal file system.
- In particular, the client is offered only an interface containing various file operations, but the server is responsible for implementing those operations.
- This model is therefore also referred to as the remote access model as shown in Fig. 4.29 (a).
- In contrast, in the upload/download model a client accesses a file locally after having downloaded it from the server, as shown in Fig. 4.29 (b).
- When the client is finished with the file, it is uploaded back to the server again so that it can be used by another client.
- The Internet's FTP service can be used this way when a client downloads a complete file, modifies it, and then puts it back.
- NFS has been implemented for a large number of different operating systems, although the Unix versions are predominant.



(a) Remove Access Model **(b) Upload/download Model**

Fig. 4.29

- For virtually all modern Unix systems, NFS is generally implemented following the layered architecture shown in Fig. 4.30.
- A client accesses the file system using the system calls provided by its local operating system.
- However, the local Unix file system interface is replaced by an interface to the Virtual File System (VFS). The VFS is a de facto standard for interfacing to different (distributed) file systems.

- Virtually all modern operating systems provide VFS, and not doing so more or less forces developers to largely re-implement huge parts of an operating system when adopting a new file-system structure.
- With NFS, operations on the VFS interface are either passed to a local file system or passed to a separate component known as the NFS client, which takes care of handling access to files stored at a remote server.
- In NFS, all client-server communication is done through so-called Remote Procedure Calls (RPCs).
- An RPC is essentially a standardized way to let a client on machine X make an ordinary call to a procedure that is implemented on another machine Y.

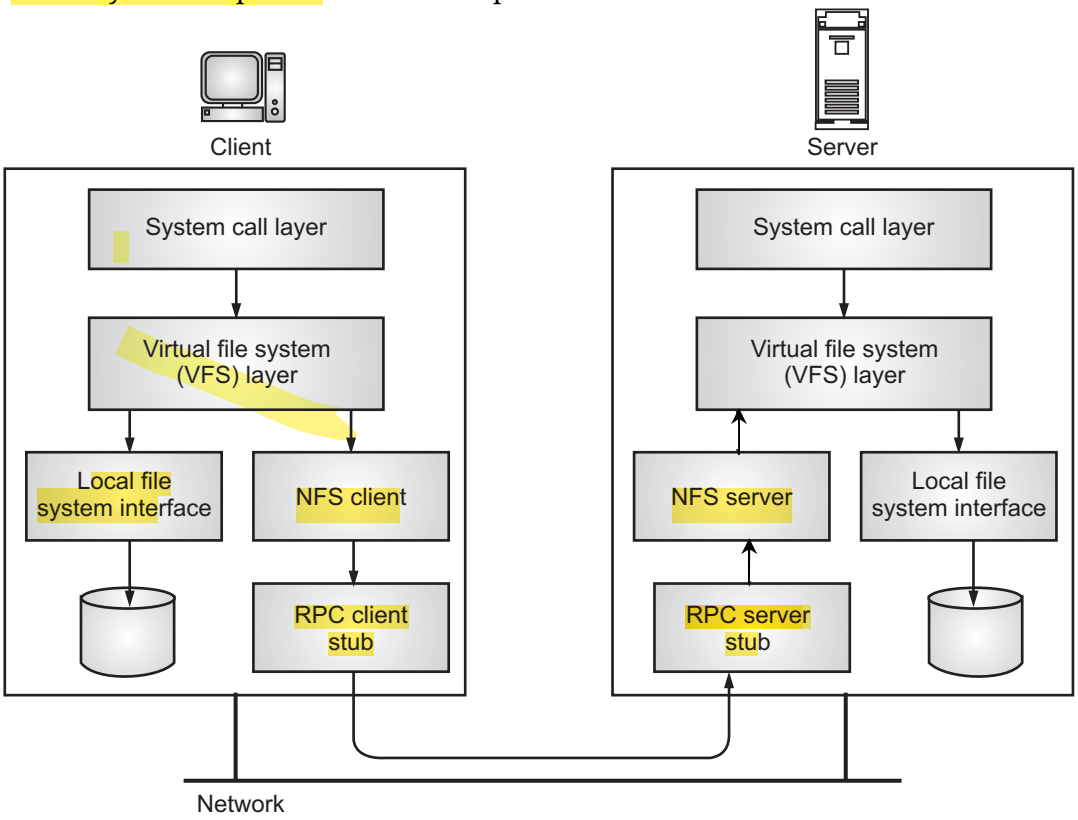


Fig. 4.30: The Basic NFS Architecture for Unix Systems

- The NFS client implements the NFS file system operations as remote procedure calls to the server. Note that the operations offered by the VFS interface can be different from those offered by the NFS client.
- The whole idea of the VFS is to hide the differences between various file systems. The NFS server is responsible for handling incoming client requests.
- The RPC component at the server converts incoming requests to regular VFS file operations that are subsequently passed to the VFS layer.

- Again, the VFS is responsible for implementing a local file system in which the actual files are stored.
- An important advantage of this scheme is that NFS is largely independent of local file systems.
- In principle, it really does not matter whether the operating system at the client or server implements a Unix file system, a Windows file system, or even an old MS-DOS file system.
- The only important issue of this scheme is that these file systems are compliant with the file system model offered by NFS.
- For example, MS-DOS with its short file names cannot be used to implement an NFS server in a fully transparent way.

4.5.2 Web-based Distributed Systems

- The Web-based distributed system has to get used to different hard wares, operating systems, different browsers and browsers of different versions.
- The Web-based distributed system can be the most favorable solution to solve the problem of system integration.
- With the development of cloud computing, the Web-based distribution will enjoy a greater development.
- The architecture of Web-based distributed systems is not fundamentally different from other distributed systems.
- However, it is interesting to see how the initial idea of supporting distributed documents has evolved since its inception in the 1990s.
- Documents turned from being purely static and passive to dynamically generated containing all kinds of active elements.
- Furthermore, in recent years, many organizations have begun supporting services instead of just documents.

Simple/Traditional Web-Based Systems:

- Unlike many of the distributed systems we have been discussing so far, Web-based distributed systems are relatively new.
- Many Web-based systems are still organized as relatively simple client-server architectures.
- The core of a Website is formed by a process that has access to a local file system storing documents.
- The simplest way to refer to a document is by means of a reference called a Uniform Resource Locator (URL).
- The URL specifies where a document is located, often by embedding the DNS name of its associated server along with a file name by which the server can look up the document in its local file system.

- A URL also specifies the application-level protocol for transferring the document across the computer network.
- A client interacts with Web servers through a special application known as a browser. A browser is responsible for properly displaying a document.
- A browser also accepts input from a user mostly by letting the user select a reference to another document, which it then subsequently fetches and displays.
- The communication between a browser and Web server is standardized and they both adhere to the HyperText Transfer Protocol (HTTP).
- The overall organization of a simple/traditional Website is shown in Fig. 4.31.
- Web documents or Web pages is a standard text file. Web documents have been marked up, usually done in the markup language; the most widely-used markup language in the Web is HTML (HyperText Markup Language).
- Another, increasingly important markup language is the Extensible Markup Language (XML) which, as its name suggests, provides much more flexibility in defining what a document should look like.

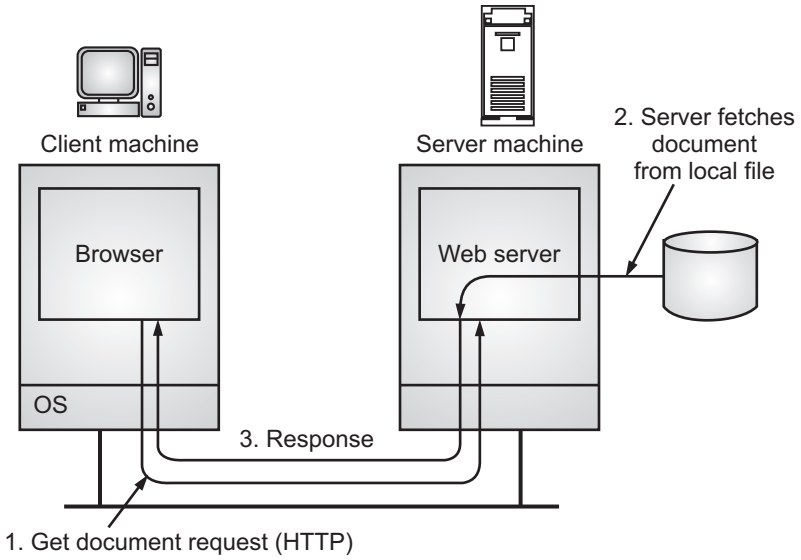


Fig. 4.31: Overall Organization of a Traditional/Simple Website

Multi Tiered Architectures:

- Fig. 4.32 shows a simple two-tiered client-server system of Web. By now, this simple architecture has been extended to support much more sophisticated means of documents.
- Most things that we get to see in our browser have been generated on the spot as the result of sending a request to a Web server.
- Content is stored in a database at the server's side, along with client-side scripts and such, to be composed on-the-fly into a document which is then subsequently sent to the client's browser. Documents have thus become completely dynamic.

- One of the first enhancements to the basic architecture was support for simple user interaction by means of the Common Gateway Interface (CGI).
- CGI defines a standard way by which a Web server can execute a program taking user data as input.
- Usually, user data come from an HTML form; it specifies the program that is to be executed at the server side, along with parameter values that are filled in by the user.
- Once the form has been completed, the program's name and collected parameter values are sent to the server, as shown in Fig. 4.32.
- When the server sees the request, it starts the program named in the request and passes it the parameter values.
- At that point, the program simply does its work and generally returns the results in the form of a document that is sent back to the user's browser to be displayed.

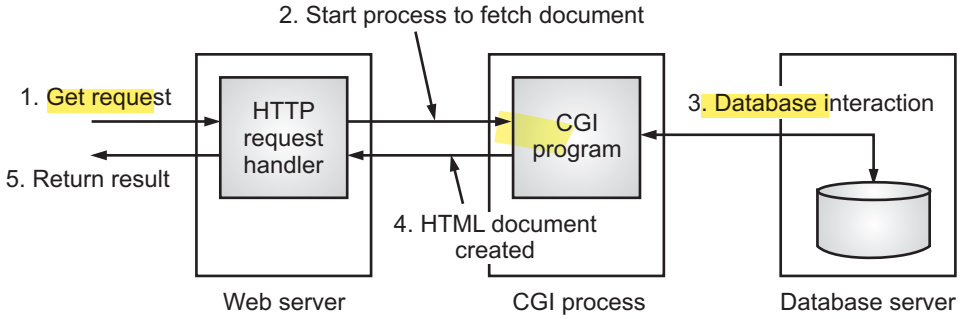


Fig. 4.32: Principle of using Server-Side CGI Programs

- CGI programs can be as sophisticated as a developer wants. For example, as shown in Fig. 4.32, number of programs operates on a database local to the Web server.
- After processing the data, the program generates an HTML document and returns that document to the server. The server will then pass the document to the client.
- An interesting observation is that to the server, it appears as if it is asking the CGI program to fetch a document. In other words, the server does nothing but delegate the fetching of a document to an external program.
- The main task of a server used to be handling client requests by simply fetching documents.
- With CGI programs, fetching a document could be delegated in such a way that the server would remain unaware of whether a document had been generated on the fly, or actually read from the local file system.
- Note that we have just described a two-tiered organization of server-side software. However, servers nowadays do much more than just fetching documents.
- One of the most important enhancements is that servers can also process a document before passing it to the client.
- In particular, a document may contain a server-side script, which is executed by the server when the document has been fetched locally.

- The result of executing a script is sent along with the rest of the document to the client. The script itself is not sent.
- In other words, using a server-side script changes a document by essentially replacing the script with the results of its execution.
- As server-side processing of Web documents increasingly requires more flexibility, it should come as no surprise that many Websites are now organized as a three-tiered architecture consisting of a Web server, an application server, and a database.
- The Web server is the traditional Web server that we had before; the application server runs all kinds of programs that may or may not access the third tier, consisting of a database.
- For example, a server may accept a customer's query, search its database of matching products, and then construct a clickable Web page listing the products found.
- In many cases the server is responsible for running Java programs, called Servlets, that maintain things like shopping carts, implement recommendations, keep lists of favorite items, and so on.

PRACTICE QUESTIONS

Q. I Multiple Choice Questions:

1. Which is a computing environment in which various components are spread across multiple computers (or other computing devices like laptops, smartphones) on a network to appear as a single system to the end-user?
   (a) Centralized system (b) Distributed system
   (c) Computer system (d) None of the mentioned
2. Distributed system architectures are bundled up with,
   (a) Component is a modular unit with well-defined interfaces and reusable.
   (b) Connector is a communication link between modules which mediates coordination or cooperation among components
   (c) Both (a) and (b) (d) None of the mentioned
3. Which is a system software over a collection of independent, networked, communicating and physically separate computational nodes?
   (a) grid operating system (b) cloud operating system
   (c) distributed operating system (d) None of the mentioned
4. Design goals of distributed system include,
   (a) Being scalable (b) Computation speed-up
   (c) Being open (d) All of the mentioned
5. Which is generally refers in two or more computers (nodes) connected together?
   (a) cluster (b) grid
   (c) cloud (d) None of the mentioned

6. The term cloud refers to,
   (a) a network (b) the internet
   (c) Both (a) and (b) (d) None of the mentioned
7. Examples of public cloud in includes,
   (a) Amazon elastic compute cloud (EC2) (b) Windows Azure Services Platform
   (c) Google App Engine (d) All of the mentioned
8. Which is an application simply consists of a server running that application and making it available to remote programs called clients?
   (a) centralized (b) networked
   (c) file system (d) None of the mentioned
9. The operations on a database are carried out in the form of, transactions.
   (a) transactions (b) logs
   (c) locks (d) None of the mentioned
10. The logical levels, essentially following a layered architecture style includes,
    (a) Application interface level (b) Processing level
    (c) Data level (d) All of the mentioned
11. Which consists of multiple detection stations called sensor nodes, each of which is small, lightweight and portable?
    (a) sensor network (b) cloud network
    (c) grid network (d) None of the mentioned
12. Which computing refers to a broad set of computing operations that allow a user to access information from portable devices such as laptop computers, PDAs, cell phones?
    (a) Cloud (b) Grid
    (c) Mobile (d) All of the mentioned
13. Which computing refers to a broad set of computing operations that allow a user to access information from portable devices such as laptop computers, PDAs, cell phones?
    (a) Cloud (b) Grid
    (c) Mobile (d) All of the mentioned
14. The object-based style is based on loosely coupled arrangement of,
    (a) grids (b) objects
    (c) clouds (d) clusters
15. The REST stands for,
    (a) REpresentational State Transfer (b) REpresentation State Transfer
    (c) RepresEntational State Transaction (d) ReprEsentational State Transaction

16. The NFS architecture is a,
    (a) peer-to-peer (b) grids-cluster
    (c) client/server (d) All of the mentioned

Answers

1. (b)	2. (c)	3. (c)	4. (d)	5. (a)	6. (c)	7. (d)	8. (b)	9. (a)	10. (d)
11. (a)	12. (d)	13. (c)	14. (b)	15. (a)	16. (c)				

Q. II Fill in the Blanks:

1. The idea behind _____ systems is to provide a viewpoint of being a single coherent system, in which the set of independent computers or nodes are interconnected through a computer network like LAN and/or WAN.
2. The _____ is a de facto standard for interfacing to different (distributed) file systems.
3. A distributed operating system is generally designed to support system-wide _____ of resources, such as I/O devices, files etc.
4. The peer-to-peer system contains nodes that are _____ participants in data sharing.
5. _____ architectures are attractive because they provide a natural way to encapsulate data (called the state) and the operations that can be performed on that data (called methods or behavior) in a single entity i.e., object.
6. In the _____ architecture the clients can make requests from the server and the server will respond accordingly.
7. The _____ is architecture contains a client program, server program, and a protocol that helps for communication between the client and server.
8. The _____ computing involves mobile communication, mobile hardware and mobile software.
9. The _____ computing is an approach to achieving high performance, high reliability or high throughout computing by using a collection of interconnected computer systems.
10. A distributed system contains _____ nodes that are physically separate but linked together using the network.
11. Grid computing refers to distributed computing, in which a group of computers from multiple locations are connected with each other to achieve a common objective/goal.
12. A _____ is combination of servers, storages, network devices (like switches etc.) networked together from where computing facilities are delivered to users.
13. The _____ cloud is private in nature and allows systems and services to be accessible within an organization. It is more secured because of its.

14. Application _____ is the process of enabling independently designed applications to work together.
15. _____ peer-to-peer systems attempt to maintain a specific, deterministic overlay network.
16. A _____ network generally consists of tens to hundreds or thousands of relatively small nodes, each equipped with one or more sensing devices.
17. The _____ computing is an advanced computing concept where computation occurs anywhere using any computer, everywhere and in any medium.
18. An _____ peer-to-peer system each node maintains an ad hoc list of neighbors.

Answers

1. distributed	2. VFS	3. sharing	4. equal
5. Object-based	6. client/server	7. NFS	8. mobile
9. cluster	10. multiple	11. Grid	12. cloud
13. private	14. integration	15. Structured	16. sensor
17. ubiquitous	18. unstructured		

Q. III State True or False:

1. An operating system that manages many computers in a computer network but presents an interface of single computer to the user is called distributed operating system.
2. A distributed system is a system consisting of a collection of autonomous computers connected by a computer networks.
3. Object-based architectures and Service-Oriented Architectures (SOAs) architectural style is based on an arrangement of loosely coupled objects.
4. Grid computing refers to a network of same or different types of computers whose target is to provide an environment where a task can be performed by multiple computers together on need/requirement basis.
5. BitTorrent is an example of client/server architecture.
6. A distributed transaction is a set of operations on data that is performed across two or more data repositories (especially databases).
7. A sensor node in sensor networks monitors the data collected by the sensor and transmits this to other sensor nodes.
8. Ubiquitous computing is the new trend in computer technology, involving improvement in mobile technology and pervasive computing.
9. In structured peer-to-peer systems, locating relevant data items can become problematic as the network grows this scalability problem.
10. A computer cluster is a set of computers that work together so that they can be viewed as a single system.

11. Cloud computing is the on-demand availability of computer system resources, especially data storage (cloud storage) and computing power, without direct active management by the user.
12. The basic idea behind VFS is that each file server provides a standardized view of its local file system.
13. Hybrid cloud is a combination of the public cloud and the private cloud.
14. The edge-server systems are deployed on the Internet where servers are placed "at the edge" of the network.

Answers

1. (T)	2. (T)	3. (T)	4. (T)	5. (F)	6. (T)	7. (T)	8. (T)	9. (F)	10. (T)
11. (T)	12. (F)	13. (T)	14. (T)						

Q. IV Answer the following Questions:

(A) Short Answer Questions:

1. Define distributed system.
2. Define distributed operating system.
3. What is meant by system architecture?
4. Define grid.
5. List design goals for distributed system.
6. What are the types of distributed systems?
7. Define P2P architecture.
8. What is NFS?
9. Define SOA.
10. Define Web-based distributed systems.
11. Define cluster.
12. Define sensor network.
13. Define object-based architectures.
14. What is meant by resource-centered architecture style?
15. Define cloud.
16. Give example of hybrid cloud.
17. Define mobile computing.

(B) Long Answer Questions:

1. What is distributed system? How it works? Explain diagrammatically.
2. What is distributed operating system? Explain in detail.
3. Describe design goals of distributed system.
4. Write short note on: High performance distributed computing.
5. Define grid computing? How it works? Explain diagrammatically.

6. What is distributed information systems? Explain in detail.
7. Define cloud computing? List types of clouds with examples.
8. What is cluster computing? Define it. Also state its advantages and disadvantages.
9. What is database transaction and nested transaction? Explain with example.
10. What is mobile computing? State its advantages and disadvantages.
11. What is meant by pervasive systems and ubiquitous computing systems? Also compare them.
12. With the help of diagram describe application layering.
13. What is sensor network? Explain in detail.
14. Write short a note on: Architectural styles.
15. With the help of diagram describe layered architecture.
16. What is object-based architecture style? Explain diagrammatically.
17. What is RESTful? Explain with example.
18. With the help of diagram describe P2P architecture. Also state its advantages and disadvantages.
19. Compare grid, cloud and cluster computing.
20. What is client server system? Describe with diagram. Also state its advantages.
21. Differentiate between client/server and peer-to-peer architectures/systems.
22. Write a short note on: Decentralized organizations.
23. With the help of diagram describe hierarchically organized P2P networks.
24. Describe hybrid architectures in detail.
25. Explain NFS with its models.
26. What is web-based system? Explain in detail.
27. Differentiate between mobile computing and sensor network.

UNIVERSITY QUESTIONS AND ANSWERS

April 2018

1. Write a brief note on Distributed Operating System. **[3 M]**

Ans. Refer to Section 4.1.

■■■

CHAPTER 5

Mobile Operating Systems

Objectives...

- To understand Concept of Mobile Operating Systems
- To learn Features, Constraints and Requirements for Mobile Operating Systems
- To study ARM and Intel Architectures
- To learn Commercial Mobile Operating Systems

5.0 INTRODUCTION

- A mobile operating system is an operating system for mobile phones, tablets, smart watches or other mobile devices.
- A mobile operating system, also known as a mobile OS, a mobile platform, or a handheld operating system, is the operating system that controls a mobile device.
- Mobile operating systems combine features of a personal computer operating system with other features useful for mobile or handheld use (PDAs, tablet PCs etc.)
- Examples of mobile device operating systems include Apple's iOS, Google's Android, Research in Motion's BlackBerry OS, Nokia's Symbian and so on.

5.1 OVERVIEW OF MOBILE OPERATING SYSTEMS

- The mobile operating system is an operating system that is specifically designed to run in a small compact environment found in mobile devices such as mobile phones, smartphones, PDAS, tablet computers and other handheld devices.
- The mobile operating system is the software platform on top of which other programs, called application programs can run on mobile devices.
- A mobile operating system is an operating system that helps to run other application software on mobile devices.
- Mobile operating systems combine features of a personal computer operating system with other features useful for mobile or handheld use.

- In a mobile handset, the operating system (OS) performs following two main responsibilities:
  1. **Managing Resources:** An important responsibility of the operating system of the operating system of a mobile device is to facilitate efficient utilization of the resources of the device by performing multiple tasks. The resources that are managed by the operating system include processor, memory, files, and various types of attached devices such as camera, speaker, keyboard, and screen. Typically, a mobile device is expected to run multiple applications at the same time and each application may in turn require running multiple tasks. A task can have multiple threads.
  2. **Providing different Interfaces:** The operating system of a mobile device on the one hand provides a highly interactive interface to the user of the device and on the other interfaces with several devices and networks. An important interface concerns control, data, and voice communications with the base station, possibly requiring use of different types of protocols. Besides, an OS takes care of recognizing inputs from the keyboard, sending outputs to the display screen, and interfacing with peripheral devices such as other mobile devices, computers, printers, etc. Unlike laptops and desktops, there is a considerable variation among the user interfaces of different mobile handsets. Some handsets have touch screen-based user interface, while in some others it is based on a physical keyboard
- An Operating System (OS) is a program that acts as an interface between the system hardware and the user.
- Examples of mobile device operating systems include Apple iOS, Google Android, Research in Motion's BlackBerry OS, Nokia's Symbian OS, Hewlett-Packard's webOS (Palm OS), Harmony OS (used for IoT devices) and Microsoft's Windows Phone OS.
- A mobile operating system needs a new architecture and different features in order to provide adequate services for handheld devices. Several mobile operating systems are already available, and each employs a different architecture and implementation.
- Fig. 5.1 shows a generalized mobile operating system structure, which can be visualized as a six-layer stack.
- Fig. 5.1 shows a generic mobile operating system structure, which can be visualized as following a six-layer stack:
  1. **Layer 1 (Mobile Applications):** This layer refers to customer-level applications such as micro-browsers and mobile retailing.
  2. **Layer 2 (Graphical User Interface (GUI)):** Applications use the API to display information on the GUI, which is more limited in a mobile operating system than the GUI in a desktop OS.

3. **Layer 3 (Application Programming Interface (API) Framework):** This layer provides the framework between the low-level architecture components and the application layer. By using this framework, application developers do not need to know the details of the underlying low-level components in order to take full advantage of their capabilities.
4. **Layer 4:** This layer consists of following three components:
   - **(i) Multimedia:** Widely adopting multimedia applications is one of the reasons for the success of mobile OS. Multimedia involves image/video related functionality.
   - **(ii) Communication Infrastructure:** Includes wide area networking stack including TCP/IP, personal area networking including Bluetooth and so on.
   - **(iii) Security:** Security is most important in mobile OS because, the wireless and mobile networks are inherently more vulnerable and open to attack.
5. **Layer 5:** This layer consists of following three components:
   - **(i) Computer Kernel:** This is the central module of a mobile operating system that provides all the essential services required by the other parts the operating system and applications.
   - **(ii) Power Management:** A problem with all handheld devices is their short battery life. This component manages the power consumption in order to prolong the battery life.
   - **(iii) Real-time Kernel:** mobile handheld devices need real-lime responses for ti me-critical applications such as voice communication.
6. **Hardware Controller Layer:** It is the bottom layer includes hardware such as displays.

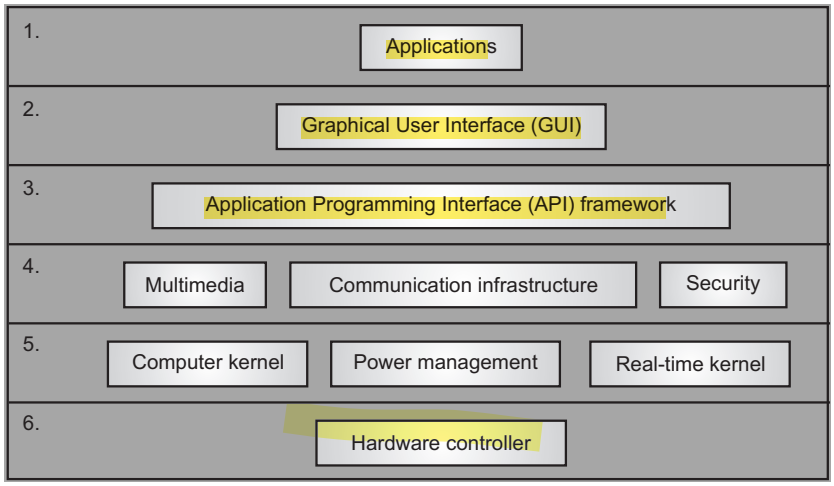


Fig. 5.1: A Generalized Structure of Mobile Operating System

5.2 FEATURES OF MOBILE OPERATING SYSTEMS

- The operating system used on mobile devices is generally referred to as mobile operating systems. Devices such as phones, smart watches, tablets, PDAs or other mobile devices make use of mobile operating systems to run programs and applications.
- Mobile operating systems are historically different from desktop OS as they were originally designed to support mobility of devices.
- An operating system for smartphones, tablets and other mobile devices is called mobile OS.
- A mobile operating system also provides an environment for other programs to run on mobile devices.
- Modern mobile operating systems mix the features of PC operating systems with many other features, such as touch screen, video camera, voice recorder, Bluetooth, Infrared, Wi-Fi, GPS mobile navigation, speech recognition, etc.
- Like a PC operating system controls the desktop or laptop computer, a mobile operating system also provides an environment for other programs to run on mobile devices.
- The following are some features of mobile operating system:

1. Easy to Use:

- The graphics should be attractive. The operating system of a mobile device on the one hand provides a highly interactive interface to the user of the device and on the other interfaces with several devices and networks.
- The buttons and features should be easy to use. Moreover, the functionalities should not be very complicated.
- Features should be powerful and useful.

2. Good App Store:

- An app is one of the basic parts of an OS.
- Good and useful apps form an important part of an OS.
- The apps should be simple and interactive.

3. Good Battery Life:

- Power is one of the main requirements of a smartphone.
- They require power for processors sensors etc. Therefore, the battery holds a very important role.
- Smartphones power usage keeps on increasing therefore, a good battery backup is very essential.
- Mobile OS like Symbian OS is designed to make minimal demands on batteries and to have low memory.

4. **Data Usage and Organization:**

- An operating system should focus on controlling the data and network usage. It should keep the limit and requirement in focus.
- Secondly, the organization of data related to to-do lists, calendars, alarms, reminders etc is very important. A good OS should keep this data in a very organized and safe manner. Moreover, the data should be readily and easily available.
- The use of Mobile OS based technologies based on agreed-upon standards which ensuring that applications are robust, portable, and interoperable.

5. **Multitasking:**

- Multitasking involves being able to rapidly switch between different apps and to combine multiple sources of information.
- Small mobile screens limit users' ability to see content from different apps at the same time, so current operating-system support for multitasking focuses mostly on switching between different apps.
- This increases users' memory load, so mobile designers must help users compare and rapidly retrieve recent items.
- Mobile OS support multitasking means all applications are designed to work seamlessly in parallel.

6. **Mobility:**

- The mobile OS should be able to work normally' as it is carried everywhere with the mobile phones.

7. **Connection:**

- Mobile OS on smartphones enables to connect to the wireless network, local computer and other devices.

8. **Innovation:**

- Mobile OS is derived from computer operating systems at first. Developers of mobile OS are striving for breakthrough innovation to make it local to smartphones.

9. **Open:**

- Mobile OS is working on based on a open platform to inactivate the applications, which is developed by independent technology and software vendors.

10. **Protection and Security:**

- Mobile security, or more specifically mobile device security, is the protection of smartphones, tablets, and laptops from threats associated with wireless computing.
- Mobile OS has security mechanisms for enabling secure communications and safe data storage.

- It should provide security for the following:
  - **Data:** Smartphones are devices for data management, and may contain sensitive data like credit card numbers, authentication information, private information, activity logs (calendar, call logs);
  - **Identity:** Smartphones are highly customizable, so the device or its contents can easily be associated with a specific person.
  - **Availability:** Attacking a smartphone can limit access to it and deprive the owner of its use.

11. The use of Mobile OS based technologies based on agreed-upon standards which ensuring that applications are robust, portable, and interoperable.
12. Mobile OS has object-oriented software architecture.
13. Mobile OS has application support for international environment with built-in Unicode character sets.
14. Mobile OS has a rich and varied API allowing access to reusable components in developer applications.

5.3 SPECIAL CONSTRAINTS AND REQUIREMENTS OF MOBILE OPERATING SYSTEMS

- Design and capabilities of a Mobile OS (Operating System) is very different than a general purpose OS running on desktop machines.
- The operating system for a mobile device needs to serve in the presence of numerous types of constraints which aren't present in a traditional computer.
- As an illustration of such a constraint, consider the fact that a mobile device is powered by oppressively limited energy stored in a bitsy battery.
- Thus, an important constraint for a mobile device lies in avoiding complex calculations and it should make the device to enter into a low power sleep mode as soon as possible. Such a constraint isn't generally assessed on a traditional operating system.
- Further, the number of times that a mobile device is generally turned on per day is significantly advanced than that of a desktop or any other computer.
- Due to this constraint, the mobile OS would need to be loaded (booted) much faster each time after it is switched on, compared to a desktop. Consequently, the kernel of a mobile OS needs to be of a very small size.
- Mobile devices have **constraints and restrictions** on their physical characteristic such as:
  1. **Limited Memory:** A mobile device usually has much less permanent and volatile storage compared to that of a contemporary desktop or laptop. For limited memory, OS must be small as possible and yet provide rich set of functionalities to meet user demands. The size of the kernel is, therefore, considered to be a very important criteria for mobile OS.

2. **Miniature Keyboard:** Typing in the documents and entering long string commands is rather difficult. Hence, mobile handsets are either provided with a small keypad or the small-sized display screen is designed to be used as a keyboard in a touch screen mode.
3. **Limited Screen Size:** The size of a mobile handset needs to be small to make it portable. This limits the size of the display screen. Consequently, new innovative user interfaces need to be supported by the mobile OS to overcome this constraint and minimize user inconveniences.
4. **Limited Processing Power**: A vast majority of modern mobile devices incorporate ARM-based processors. These processors are certainly energy efficient, powerful, and cheaper compared to the desktop or laptop processors, yet these are significantly slower. The sizes of the on-chip and off-chip memory are also restricted. Due to restricted processing power, storage, and battery power, usually the OS is made to provide only a limited number of functionalities.
5. **Limited Battery Power**: Mobile devices need to be as lightweight as possible to increase their portability. Due to the severe restrictions that are placed on their size and weight, a mobile device usually has a small battery and often recharging cannot be done as and when required.
6. **Limited and Fluctuating of the Wireless Medium**: The operating system of a mobile handset needs to run complex protocols due to the inherent problems caused by mobility and the wireless medium.

5.4 SPECIAL SERVICE REQUIREMENTS

- Several facilities and services that are normally not expected to be supported by a traditional operating system are mandated to be supported by a mobile OS.
- **Support for Specific Communication Protocols:**
  - Mobile devices are often required to be connected to the base station and various types of peripheral devices, computers and other mobile devices.
  - This requires enhanced communication support.
  - For communication with other devices and with computers, TCP/IP and wireless LAN protocols also need to be supported.
  - For web browsing as well as communication with other personal devices such as pen drive and headphones, though mobile devices are equipped with USB and other types of ports, mobility constraints often make infrared or Bluetooth connections preferable. This mandates the operating system to support multiple interfacing protocols and hardware interface.
- **Support for a Variety of Input Mechanism:**
  - A miniature keyboard forms the main user input mechanism for an inexpensive mobile device. Sophisticated mobile devices (smartphones) usually support the QWERTY keyboard.

- Along with handwriting detection, several current mobile devices include touch screen or even stylus-based input techniques. The many input mechanisms that must be supported have a significant impact on a device's intended primary function as well as the specific consumer segment for which it is positioned.
- A mobile OS needs to support a variety of input mechanisms to make it generic and usable by different manufacturers of mobile devices.

- **Compliance with Open Standard:**
  - To facilitate the third party software development as well as to reduce the cost of development and time-to-market by the mobile handset manufacturers, the OS should adhere to open standards
  - Smartphones come in many different shapes and sizes and have varying screen sizes and user input capabilities.
  - Therefore, the user interface and networking capabilities of a mobile OS need to be designed keeping these diversities in view.
- **Extensive Library Support:**
  - The third application requires extensive library support by OS. The expected library support includes:
    - primitives for email
    - SMS
    - MMS
    - Bluetooth
    - Multimedia
    - GSM/GPRS functionalities.
- **Support for Integrated Development Environment (IDE):**
  - General purpose IDE can be satisfactory used for software development.
  - But mobile operating systems need have their own IDE for effective software development and good performance of the developed software.

Difference between Mobile OS and Desktop OS:

Sr. No.	Parameters	Mobile Operating System	Desktop Operating System
1.	Definition	It allows smartphones, tablet PCs, and other devices to run applications and programs.	Main control program or environment through which user controls a personal computer and it manages all applications and programs in a computer.
2.	Purpose	Manages cellular and wireless connectivity, and phone access.	Manages hardware and software resources of the system.

contd. ...

3.	Boot time	Boots faster than desktop OS.	It boots much slower.
4.	Storage	Uses a flash drive to store data/information.	Uses hard drives / flash drives to store data/information.
5.	Power requirements	Optimized to work under minimal power requirements and have features to prevent energy loss.	Desktop OS is not readily optimized for energy loss.
6.	Interface	Mobile Os operates with touch screen or touch pad.	The PC operates via many input devices such as mouse, keyboard etc.
7.	Memory usage	Optimized to work on minimum RAM.	Requires a good amount of memory to operate.
8.	Features	• Specialized for a specific set of devices. • Don't offer complete access to system hardware (such as administrator or root). • Limited or no interoperability (Mobile apps are strictly hardware specific).	• Full features. • Designed to take advantage of fast CPUs, large amount of disk space and RAM. • Based on X86 majorly it is more flexible in terms of interoperability.
9.	OS flavors	Apple iOS, Google Android, Bada (Samsung electronics), Blackberry OS, iPhone OS / iOS, Symbian OS, Windows Mobile OS, Harmony OS, Palm OS, WebOS (Palm/HP) etc.	Windows 10, MacOS, Windows Vista etc.

5.5 ARM AND INTEL ARCHITECTURES

- The CPU is the core of a mobile handheld device. When we are choosing a smartphone or tablet, we will notice that some models use Intel processors, while others are based on the competing ARM architecture.
- Both families of chips are designed for low-power operation, to give mobile devices the long battery life they need. ARM is RISC (Reduced Instruction Set Computing) based while Intel (x86) is CISC (Complex Instruction Set Computing).

- In this section we will study ARM and Intel (x86) processor architectures. The core hardware in mobile handheld devices is the mobile processor.
- The performance and functionality of the mobile devices are largely dependent on the capabilities of their processors.
- There used to be several brands available, but recently mobile processors designed by ARM Ltd. have begun to dominate the market. ARM is the industry's leading provider of 32-bit embedded RISC microprocessors, with almost 75% of the market.
- Handheld devices are becoming more sophisticated and efficient every day and mobile users are demanding more functionality from the devices.
- Intel entered the mobile marketplace with mobile specific microprocessor products competitive with ARM Ltd., the market leader in the mobile space.
- Combined with the robustness and flexibility of the Android OS, the power and compatibility of the x86 series of processors is bringing a competitive new device family to the mobile market.
- The x86 architecture is a CISC system, built with more complex instructions facilitating ease of use and simpler implementations.
- ARM is Intel's main competitor for mobile processors and is RISC system, without these features. For example, it might take three to four instructions to load a given value into memory, whereas in a CISC system there is one single instruction specifically written to do this.
- The x86 architecture is also register to memory based, meaning instructions can affect both registers and memory.
- The term "x86" came into being because the names of several successors to Intel's 8086 processor end in "86", including the 80186, 80286, 80386 and 80486 processors.
- Intel's latest home computing processor series based on x86 architectures is nicknamed the Intel Core i-series.
- This i-series supports 64-bit operations and is focused on performance and speed. All of the processors support hyper-threading and have multiple cores, which allow for concurrent processing.
- Running parallel to the personal computing Core i-series of microprocessors is the x86 based Atom series for mobile devices.

Concept of Mobile Processor:

- The core hardware in mobile handheld devices are the mobile processors and the performance and functionality of the devices are largely dependent on the capabilities of the processors.
- Unlike desktop processors, mobile processors power portable devices such as laptops, tablets or smartphones.
- In mobile terms, "processor" more often refers to the System on a Chip (SoC). In other words, the CPU in a smartphone will usually be integrated with a system on a chip process.

- System on a Chip (SoC) processors, as the name suggests, integrate many different functions into a single integrated circuit.
- Normally SoC will include CPU, memory and secondary storage. It may also include a Graphics Processing Unit (GPU), Wi-Fi and other technologies.
- This is in contrast to the personal computer that uses a motherboard with various circuits (including CPU, memory and secondary storage) attached to it.
- The SoC integrates the CPU with the other key components such as memory, GPU, USB controller, wireless radio etc., into a single integrated chip. The SoC has the advantage of its highly compact size, lower cost and less wiring.
- Today's mobile processors must have the following features:
  1. **High Performance:** The clock rate must be higher than the typical 30 MHz for Palm OS PDAs, 50 MHz for cellular phones and 200 MHz for devices that run Microsoft's Pocket PC.
  2. **Low Power Consumption:** This prolongs battery life and prevents heat buildup in handheld devices that lack the space for fans or other cooling mechanisms.
  3. **Multimedia Capability:** Applications like audio/image/video are recurring themes in mobile commerce.
  4. **Real-time Capability:** This feature is particularly important for time-critical applications such as voice communication.

5.5.1 ARM Architecture

- ARM acronym for Advanced RISC Machines. The ARM is a microprocessor architecture initially developed by a British company called Acron.
- The ARM company (a spin-off of Acron), licenses ARM technologies to semiconductor manufacturers and electronics device manufacturers.
- The ARM core is the leading 32-bit embedded processor architecture for a variety of mobile devices.
- The ARMv6, released to licensees in October of 2002. In 2015, ARM introduced the ARMv7 revision of its architecture, known as the "Cortex" architecture. The ARMv7 is used widely in the modern smartphone market.
- The ARM design was initially targeted at ultralow-power embedded devices. As technology evolved so did ARM's processor design and it is estimated that an ARM processor care is used in 95% of mid- to high-end mobile devices today.
- The current ARM-Cortex A9 and A15 platforms used in high-end smartphones and tablets is the result of a bottom-up approach, as it has evolved from earlier platforms for simpler devices.
- The number of typical mobile general purpose processors are based on the ARM architecture, which describes a family of computer processors designed in accordance with a RISC CPU design.

- The ARM architecture includes load/store architecture with restrictions to misaligned memory and a uniform 16×32-bit register file.
- A fixed instruction of 32-bits allows easy decoding and pipelining with a decreased code density.
- However, to improve compiled code-density, most ARM processors include Thumb, a 16-bit instruction subset.
- A VPP (Vector Floating Point) co-processor is included to provide for low-cost floating points computations although later versions of the architecture have abandoned it in favor of more complete SMID units.
- The particularities of ARM processors enable some code optimizations to achieve higher performance.
- General ARM optimization tips include, for example, the use of do-while loops and counter decrement.
- Today, a lot of operating systems support the ARM architecture. Examples are fully embedded operating systems of low-end to mid-range mobile devices to operating systems for smartphones such as Symbian and Windows Phone.
- In addition, ARM processors are also used with operating systems that were initially developed for desktop computers such as Linux and Windows 8.
- Linux is a relatively new operating system for mobile devices as the first mass market device based on Linux was only shipped in 2008 as part of Google's Android operating system.
- The advantage of using Linux as an operating system for mobile devices is that a significant amount of code can be shared between the desktop and the mobile version of the operating system.
- As relatively little code directly deals with the differences of the x86 architecture found in the PC world and the ARM architecture found on mobile devices.
- ARM processor features include:
  1. Load/store architecture.
  2. An orthogonal instruction set.
  3. Mostly single-cycle execution.
  4. Enhanced power-saving design.
  5. ARM has 64 and 32-bit execution states for scalable high performance.
  6. Hardware virtualization support.

Advantages of ARM Architecture:

1. ARM architectures are incredibly small. The ARM processor's smaller size makes them suitable for increasingly miniaturized devices.
2. ARM chips can contain a number of core components like CPUs, GPUs, DSPs and so on.

3. ARM architectures consume a minimal amount of power consumption. The lower power used the longer battery life.
4. The use of more cores and caches in ARM processors have proven to be an effective means to maintain lower power consumption with increased speed.

Disadvantages of ARM Architecture:

1. ARM chips are usually slower than their Intel x86.
2. In heavy processing situations ARM chips down performance.
3. ARM processors are also inherently less scalable, especially in comparison to modern Intel CPUs.
4. Lower speed and performance.

5.5.2 Intel Architecture

- The x86 forms the base architecture of an enormous family of Intel processor, ranging from the earliest Intel 8006 to the Pentium line, the i-series, recent virtualization hypervisor-equipped server processors, low-power Atom and Haswell micro-processors designed for mobile and embedded use and the tiny Quark system-on-chip aimed at wearable computing.
- Intel is at the other end of the spectrum and is keen to play a major role in the mobile space with its x86 processor architecture.
- A few years ago Intel tried to get a foothold in the mobile space by licensing ARM technology and building a product line around that architecture.
- In the meantime, however, Intel has abandoned this approach and has been refining their x86 architecture for low power consumption and size for several years.
- In 2012, the size, processing speed and power consumption of the chipset was for the first time balanced enough for a smartphone-sized device.
- First prototypes were shown running an x86 version of Android on a form factor smartphone and commercial products based on this design appeared shortly afterward on the market.
- This rather late competition to ARM's dominance in the mobile space is the result of Intel's approach that is directly the opposite of ARM's as they had to streamline powerful desktop processor architecture for smaller devices.
- Using an x86 platform for mobile devices has the advantage that even fewer adaptations are required for operating systems such as Linux and Windows to use them on mobile devices compared to the ARM approach described above.
- In the case of Android, most applications are executed in a virtual machine based in Java and only compiled to native code at runtime, so the same executable runs without modification or the need for recompilation on both CPU architectures.
- At the time of publication, Intel and ARM have come quite close in terms of performance and power consumption and the two architectures are now competing for use in high-end mobile devices.

Advantages of Intel Architecture:

1. Intel processors have the highest performance than other processors.
2. The x86 has higher processing speed.
3. The x86 architecture is granted to access to the largest collection of software available.
4. The x86 architecture gives scalability of high-end systems.

Disadvantages of Intel Architecture:

1. The Intel x86 architecture is physically much larger than other brands CPUs.
2. The Intel x86 architecture consumes high power

Difference between ARM Architecture and x86 Architecture:

Sr. No.	ARM Architecture	x86 Architecture
1.	Uses Reduced Instruction Set computing Architecture (RISC).	Uses Complex Instruction Set computing Architecture (CISC).
2.	Executes single instruction per cycle.	Executes complex instruction at a time, and it takes more than a cycle.
3.	Optimization of performance with a software focused approach.	Hardware approach to optimize performance.
4.	Requires less registers, more memory.	It uses more registers and less memory
5.	Pipelining of instructions is a unique feature.	Less pipelined.
6.	Faster execution of instructions reduces time.	Time to execution of instructions is more.
7.	Complex addressing is managed by software.	Inherently designed to handle complex addresses.
8.	Compilers play a key role in managing operations.	The microprogram does the trick.
9.	Multiple instructions are generated from a complex one and executed individually.	Its Architecture is capable of managing complex statement execution at a time.
10.	Managing code expansion is difficult.	Code expansion is managed easily.
11.	Decoding of instruction is handled easily.	Decoding is handled in a complex way.
12.	Uses available memory for calculations.	Needs supplement memory for calculations.
13.	Deployed in mobile devices where size, power consumption speed matters.	Deployed in Servers, Desktops, Laptops where high performance and stability matters.

5.5.3 Power Management

- Portable devices such as mobile phones and tablets require power management techniques to meet the increasingly challenging performance requirements.
- Mobile devices are battery powered devices. In today's world of mobile communications, one of the most precious commodities is power. The mobile host can only operate as long as its battery maintains power.
- Batteries are the most common power source for mobile computers. In mobile communications, power is the most consuming source.
- The mobile can work only until its battery maintains power. But the battery is a limited source of energy.
- Increased portability of mobile devices has been claimed to enable users to access it anytime anywhere, however, battery life limitations remain to be one of the major constraints. Once the battery runs out, the mobile device becomes nothing.
- The mobile host can only operate as long as its battery maintains power. Wireless networks consist of small portable devices such as PDAs, mobile phones, headsets etc., which have limited processing power and battery energy.
- Message transmission consumes energy and transmission energy requirements vary with time depending on the channel conditions. The use of mobile devices is directly affected by batteries.
- Because sometimes, the battery needs to be replaced or recharged. Many physical components are responsible for energy consumption in a mobile device.
- The portable devices have limited battery energy that is why it is required to manage it properly.
- Some people try to reduce the physical dimensions of batteries but it will only reduce the amount of charge retained by the batteries.
- It will reduce the amount of time a user can use the mobile device being forced to recharge the batteries.
- The main need of battery management is interacting and involving the user in managing their resource.
- For mobile phones, users get into habits of charging their mobile at suitable periods, as they know their call patterns and the device has been optimized for low power stand modes.
- Power management broadly refers to the management of power-related activities in a mobile device, which include generation, storing, distribution, conservation, and control of regulated voltages required to operate the host mobile system.
- The power management unit in mobile phones can be divided into two sections namely, the power distribution and switching unit, and the charging unit.

- Power distribution section is used for the distribution of voltage and current to the other components of the smartphone. This section is generally part of an Analog BaseBand (ABB) unit.
- The ABB takes power from the battery and converts it to various voltages such as 4.8 V, 2.8 V, 1.8 V and 1.6 V and distributes it to other components.
- Charging section is responsible for charging the battery of the mobile phone. It is composed of a charging integrated circuit, which takes power from an external source and charges the battery of the smartphone.
- Mobile OS are energy-dependent and its high performance relies heavily on how much energy it can consume.
- There are usually several modes for users to choose such as energy saving model, sleeping mode, normal mode and high performance mode.
- While Mobile OS under its higher performance mode gives significantly faster response to user requests, it also consumes much more energy than its normal mode or energy saving mode.
- Most of the power stored in the battery of a Smartphone is consumed by the mobile phone's operating system.
- Due to the limited battery time, user activities may be affected. In other words, value creation is not fully realized.
- Power management aims to improve battery life of equipment by minimizing power consumption while guaranteeing expected system performance.
- The rise of the ARM architecture in mobile computing has the potential to adjust the balance of power in the computing world as mobile devices become more popular and supplant PCs for many users.
- ARM is the CPU architecture used by all modern Smartphone in both the Android and Apple ecosystems.
- ARM processors are often powered by small batteries when being used extensively in consumer electronics, including PDAs, tablets, mobile phones, digital media and music players, handheld game consoles, calculators and computer peripherals such as Hard drives, printers and routers.
- In recent years, ARM architecture has been embedded in low-power products such as smartphones, media players and tablet PCs.
- ARM is an RISC architecture to reduce the power. Many power saving techniques are selectively implemented in chips.

5.6 MOBILE OS ARCHITECTURES

- Mobile phones are the most popular device for communication today. Every mobile requires some type of mobile operating system as a platform to run the other services and being easy for the users to use the services like voice calling, messaging service, camera functionality, Internet facilities and so on.

- The architecture of a system reflects the structure of the underlying system. The architecture defines the different components of the system, the functions of these components and the overall interactions and relationships between these components.
- This concept is true for general computer systems as well as for software systems. The software architecture of a program or computing system is the structure or structures of the system, which comprises software elements or modules, the externally visible properties of these elements and the relationships between them.
- Software architecture can be thought of as the representation of an engineering system and the process(es) and discipline(s) for effectively implementing the design(s) of such a system.
- In Fig. 5.2, mobile operating system layers are shown which are generic in nature.
  - At the bottom, hardware boards with processors are controlled by the operating system and the user has no control.
  - Middle library API can be accessed by user programs. But the application framework controls permission during installations.
  - Moreover, communication services are also staked in the form of HTTP and similar.

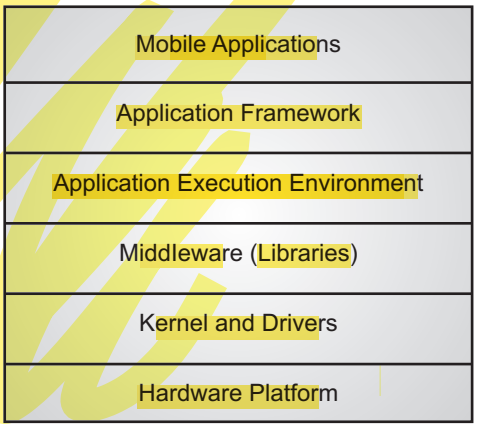


Fig. 5.2: Stack of Mobile Operating System

5.6.1 Underlying OS

- An Operating System (OS) is a collection of software that manages computer hardware and provides services for programs.
- Specifically, it hides hardware complexity, manages computational resources, and provides isolation and protection. Most importantly, it directly has privileged access to the underlying hardware.
- The operating system and hardware where the application will be deployed, there are a number of changes we can make to improve performance.
- The mobile operating system is the underlying technology that controls the mobile device. The capability of a mobile operating system directly impacts the mobile device functionality.

- The most common mobile operating systems are Symbian from the Symbian foundation. Android from Google, iOS from Apple, RIM Blackberry OS and Windows Mobile from Microsoft.
- The major improvement in mobile operating systems came with the introduction of smartphones which were supported by full featured mobile systems that allowed more advanced computing.
- A mobile operating system, also called a mobile OS, is an operating system that is specifically designed to run on mobile devices such as mobile phones, smartphones, PDAs, tablet computers and other such devices.
- The mobile operating system is a software platform consisting of related programs and data. These programs are used to run the applications on the mobile devices.
- The mobile OS is responsible for managing all the hardware, resources, memory, and optimizes the efficiency of the applications in the device.
- A mobile operating system or mobile OS is an operating system that is designed to run on a mobile device, such as a mobile phone, Personal Digital Assistant (PDA) or a tablet corrupter.
- Like traditional operating systems, a mobile OS is the software layer that sits between the device's hardware and its application layers.
- As such, a mobile OS is an abstraction built in software that hides the details of the hardware layer from the applications that sit atop it.

5.6.2 Kernel Architecture

- The kernel is the core of an operating system. It is the software responsible for running programs and providing secure access to the machine's hardware.
- The kernel relies upon software drivers that translate the generic command into instructions specific to that device.
- The kernel provides and manages computer resources, allowing other programs to run and use these resources.
- The core programs and data of an operating system together comprise the kernel. The kernel consists of the code that runs the operating system on a CPU and the data - typically organized in tables - that are used to keep track of how things are running on an operating system.
- The kernel is where the access to hardware is done and the kernel implements the operating system‘s design model.
- Mobile OS typically employ a micro kernel approach for provisioning OS services. The approach does not just mean that the kernel is of small size, rather it provides a set of essential OS services which are used by a different set of other processes that team up together to provide high level OS services to the applications.
- The system level services such as file system, network interface and TCP/IP are implemented as user libraries, and do not need privileged executions.

- Accesses to these services are coordinated by the kernel through a client-server mechanism.
- The kernel is directly responsible for the management and the protection of memory for both the user applications and the applications providing the OS services.
- The kernel also provides few other core system services such as process management, scheduler service, and driver modules.
- Since, high level OS services are implemented as non-privileged processes; it is possible to extend the OS services using APIS in the user written applications.
- The kernel is a computer program at the core of a computer's operating system and has complete control over everything in the system.
- A kernel connects the application software to the hardware of a computer as shown in Fig. 5.3.

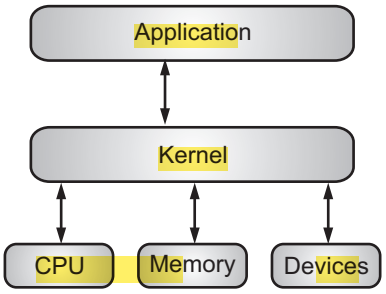


Fig. 5.3: Kernel Architecture

- There are several types of kernel. Monolithic kernels are found on some general-purpose computers; they implement all operating system functions and hardware abstractions within the kernel itself.
- Monolithic kernel (See Fig. 5.4) usually comprises large amounts of code and large amounts of memory for system tables.
- Linux is typically considered a monolithic-kernel operating system.

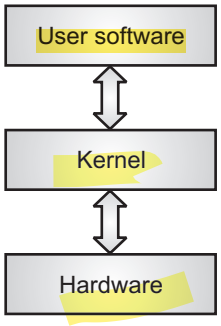


Fig. 5.4: Monolithic Kernel Structure

- Micro kernels provide only a small set of system functions and hardware models.

- Much of the remaining functionality that might be found in a monolithic kernel is provided by server applications that run outside a microkernel, (See Fig. 5.5).
- Servers in Symbian OS provide this type of functionality.

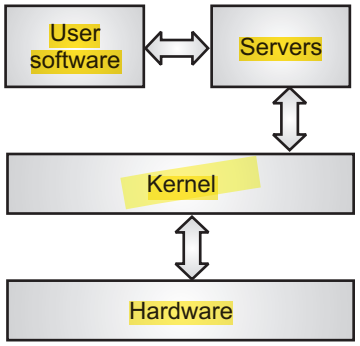


Fig. 5.5: Structure of a Microkernel

- Hybri rnels except that some of the external application function is implemented in the kernel for performance reasons, (See Fig. 5.6).

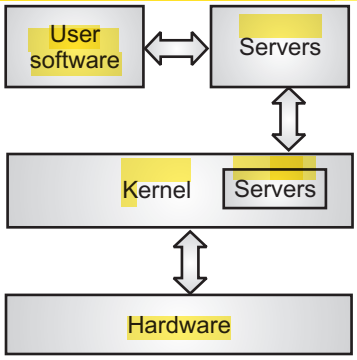


Fig. 5.6: Hybrid Kernel Structure

5.6.3 Native Level Programming

- Native code is computer programming that is compiled to run with a particular processor (such as an Intel x86-class processor) and its set of instructions.
- If the same program is run on a computer with a different processor, software can be provided so that the computer emulates the original processor.
- In programming, native code is code that is written to run on a specific processor. To run the code on a different processor, an emulator must be used to trick the program into thinking it has a different processor.
- If an emulator is used, the code almost always runs more slowly than it would in its native environment.
- Native language is a language that can run on the platform without converting it first, or to do anything with it. Example is C/C++ on Symbian.

- Managed language is a language that must be converted or interpreted before it can be run on the platform. Example is Java on Windows or .NET.
- Dynamic language is a language that allows us to convert some a string to an integer without a lot of things. Example is Python.
- Native mobile application is low-level programming and all native applications that we know are written by native apps such as clock, camera, contacts, location and calendar.
- However, it means that every platform does not work on clock on iOS that is written on Android or uses different native apps.
- The Android application is written in Java programming language while the iOS application is written in the Objective-C programming language.
- Means we can't use the same codebase for each platform and we must write the same logic for each one.
- The advantage of choosing a native app is that it is the fastest and most reliable when it comes to user experience.
- Native apps can also interact with all the device's operating system features such as the microphone, camera, contacts lists, etc.
- A "native" implementation means we are writing the application using the programming language and programmatic interfaces exposed by the mobile operating system of a specific type of device.
- For example, a native implementation for an iPhone will be written using the Objective-C language (or more recently the Swift programming language) and the iOS operating system APIs that Apple supplies and supports.
- Native application implementation has the advantage of offering the highest fidelity with the mobile device. However, native mobile app implementations are completely non-portable to any other mobile operating system - for example, a native Apple iOS app must be totally rewritten if it is to run on an Android device.

5.6.4 Runtime Issues

- A runtime issue is a problem that happens during the execution of a program.
- A program is running or during its runtime, it may encounter problems during execution. When a problem arises that the software cannot resolve, it throws a runtime error.
    - A runtime issue can be caused by poor programming practices.
    - Viruses and other malware can run in the background undetected and can cause runtime issues.
- Because most mobile operating systems are similar in a number of ways to their older versions, the operating systems in the PCs and laptops, which have seen and continue to see growing problems with security such as backdoors, spyware, worms, Trojans and a growing list of others.

- Mobile operating systems are as crucial and central to the running and security of the mobile device as they are in the larger, less mobile devices such as PCs and laptops.
- When it comes to security-related issues, the mobile device is as secure as its operating system. So every mobile device integrates in its operating systems as many security features as it can possibly carry without sacrificing speed, ease of use and functionalities expected by the consumers.
- Because most mobile operating systems are similar in a number of ways to their older brothers, the operating systems in the PCs and laptops, which have seen and continue to see growing problems with security such as backdoors, spyware, worms and a growing list of others.

5.6.5 Approaches to Power Management

- Power demands in mobiles are increasing rapidly because more and more power consuming applications are developed for mobile platforms.
- In this section we will study approaches for power management in mobile devices including Advanced Power Management (APM), Advanced Configuration Power Interface (ACPI) and Operating System Power Management (OSPM).
- To make low power consumption possible for mobile computing systems, hardware and operating systems need to work together.
- Coordinating operating systems and hardware for both power consumption and power management requires a unified set of interface specifications.
- The earliest specification was Advanced Power Management (APM), released by Intel and Microsoft.
- APM is a set of APIs, running on IBM-compatible PC operating systems and BIOS synergy to manage power consumption.
- The current specification is Advanced Configuration and Power Interface (ACPI), which comes from the development of APM.
- The basic working principle of APM is to control the power consumption of the system by monitoring the system activity and adjusting the power states or the power resources accordingly.
- In APM, one or more layers of software support power management in computers with power manageable hardware.
- APM's objective is to control the power usage of a system on the basis of the system's activity.
- Power is reduced in APM gradually as system resources become unused until the system suspends.
- ACPI defines new methods for power control. It enables an operating system to implement system-directed power management.

- The ACPI hardware interface is a standardized way to integrate power management throughout a portable system's hardware and OS and applications software.
- The ACPI gives the operating system direct control over the power management and plug-and-play functions of a computer.
- ACPI is an open industry standard for power-management services. ACPI is compatible with multiple operating systems with the initial goal is to use it with personal computers.
- Operating System Power Management (OSPM) is an operating system technology for managing the power of the underlying platform and switching it between different power states.
- OSPM enables a platform or system to implement the most efficient power mode and is applicable across all devices and components within a platform/system.
- OSPM is also known as Operating System-directed configuration and Power Management.
- OSPM is primarily designed to work on handheld or portable devices such as laptops and tablets.
- For OSPM, the entire system is divided into different categories such as the core system and subsystem, each with its own unique power requirements.
- OSPM works with the System Controller Unit (SCU), a power management unit that directly interfaces with the all hardware components.
- OSPM power management approach defines power policies, monitors the system, and power management decisions.

5.7 COMMERCIAL MOBILE OPERATING SYSTEMS

- Like computer operating systems a mobile operating system is a software platform on top of which other programs run.
- When we purchase a mobile device, the manufacturer will have chosen the operating system for that specific device.
- The operating system is responsible for determining the functions and features available on your device, such as thumb wheel, keyboards, WAP, e-mail, text messaging and more.
- Some of the more common and well-known Mobile operating systems include Windows Mobile OS, iPhone OS (iOS), Android OS and so on.

5.7.1 Windows Mobile OS

- Windows Mobile is a discontinued family of mobile operating systems developed by Microsoft for smartphones and personal digital assistants.
- The Windows Mobile, a variant of the Windows CE (also known officially as Windows Embedded Compact), was developed for the Pocket PCs at the beginning but arose by 2002 to the HTC2 mobile phones.

- Windows Mobile is a compact operating system designed for mobile devices and based on Microsoft Win32.
- It provides ultimate interoperability. Users with various requirements are able to manipulate their data.
- Windows CE (Compact Edition) designed specifically for handheld devices, based on the Win32 API.
- PDA (personal digital assistant), palmtop computer, PocketPC were original intended platform for the Windows Mobile OS.
- This is 32-bit multithreading and multitasking operating system. This OS is used in many handheld devices which are used on the plant floor.

Features:

1. Most versions of Windows Mobile have a standard set of features, such as multitasking and the ability to navigate a file system similar to that of Windows 9x and Windows NT, including support for many of the same file types.
2. Similarly to its desktop counterpart, it comes bundled with a set of applications that perform basic tasks.
3. Internet Explorer Mobile is the default web browser, and Windows Media Player is the default media player used for playing digital media. The mobile version of Microsoft Office is the default office suite.
4. Internet Connection Sharing, supported on compatible devices, allows the phone to share its Internet connection with computers via USB and Bluetooth.
5. Windows Mobile supports virtual private networking over PPTP protocol. Most devices with mobile connectivity also have a Radio Interface Layer.
6. The Radio Interface Layer provides the system interface between the Cell Core layer within the Windows Mobile OS and the radio protocol stack used by the wireless modem hardware. This allows OEMs to integrate a variety of modems into their equipment.
7. The user interface changed dramatically between versions, only retaining similar functionality. The Today Screen later called the Home Screen, shows the current date, owner information, upcoming appointments, e-mails, and tasks.
8. The taskbar displays the current time as well as the volume level. Devices with a cellular radio also show the signal strength on said taskbar.

Architecture of Windows Mobile OS:

- Windows Mobile operating system is a mobile operating system developed by Microsoft, used in smartphones.
- A Windows Mobile platform is actually a combination of the underlying operating system (an optimized variation of Windows CE), APIs, software development tools and a set of standard applications.

- Fig. 5.7 shows the architecture of Windows Mobile. On top of the Original Equipment Manufacturer (OEM) hardware is the operating system, a version of Windows CE that was designed to support a variety of low-capability mobile devices such as PDAs, cell phones, smartphones, bar code readers, handheld computers, and embedded devices in automobiles, among others.
- The .Net Compact Framework provides a facility and classes that allow one to develop managed applications.
- The traditional Win32 application design paradigm utilizing Win32 APIs or Microsoft Foundation Classes (MFC) is still supported.

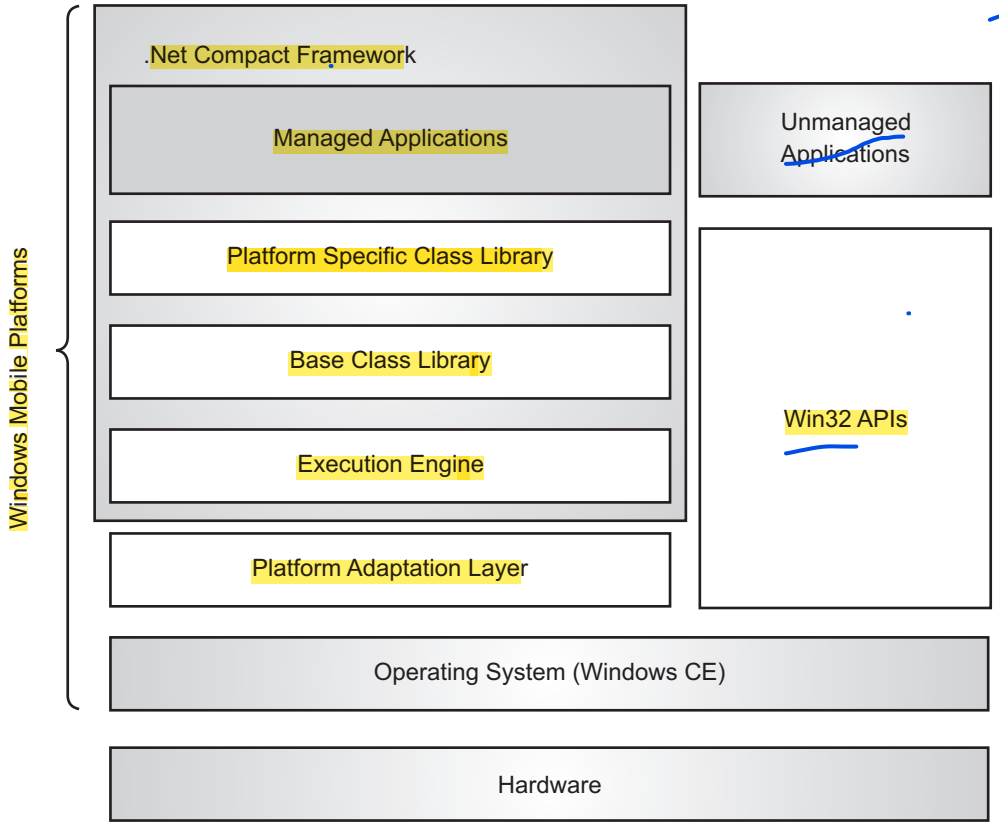


Fig. 5.7: Architecture of Windows Mobile

Advantages of Windows Mobile OS:

1. Built in support for Windows Office documents
2. Multi-tasking
3. Phones available form most service providers
4. Excellent development tools, with free versions available to students
5. Updates available directly from Microsoft

Disadvantages of Windows Mobile OS:

1. Closed architecture
2. Small number of applications available
3. Browser is a mix of IE7 and IE8 (a bit dated)
4. Applications must be approved by Microsoft before being used

5.7.2 iPhone OS (iOS)

- The iOS (formerly iPhone OS) is a mobile operating system created and developed by Apple Inc. exclusively for its hardware.
- The iOS is the operating system that powers many of the company's mobile devices, including the iPhone and iPod Touch; the term also included the versions running on iPads until the name iPadOS was introduced with version 13 in 2019.
- The iOS (iPhone OS) is a mobile operating system which is successfully designed and developed by Apple Inc.
- iOS is the largest used mobile operating system after Android. It is basically designed for iPhone, iPad, and iPod Touch.
- The iOS is Apple's most advanced and feature-rich proprietary mobile OS and was released with the first generation of the iPhone.
- When introduced, it was named iPhone OS and it was later renamed iOS to reflect the unified nature of the OS that powers all Apple iOS devices, such as the iPhone, iPod touch, iPad and Apple TV.
- The iOS is derived from core OSX technologies and is streamlined to be compact and efficient for mobile devices.

Features of iOS:

1. **Multitasking:** iPhone offers multitasking features. It started with iPhone 4, iPhone 3GS. By using the multitasking feature on an iOS device or using a multi-finger gesture on an iPad, you can swiftly go from one app to another at any moment.
2. **Social Media:** Sharing content and displaying an activity stream are just a few of the ways iOS makes it simple to integrate social network interactions into the app.
3. **iCloud:** Apple's iCloud is a service that offers Internet-based data storage. It works on all Apple devices and has some Windows compatibility, and handles most operations in the background. It is highly encrypted. It offers the backup option to help the user not lose any of their data.
4. **In-App Purchase:** In-app purchases, which are available on all Apple platforms, provide users with additional material and services, such as digital items (iOS, iPadOS, MacOS, WatchOS), subscriptions, and premium content, right within the app. We may even use the App Store to promote and sell in-app purchases.

5. **Game Center:** Game Center, Apple's social gaming network, adds even more pleasure and connection to your games. Game Center provides access to features such as leaderboards, achievements, multiplayer capability, a dashboard, and more.
6. **Notification Center:** Notification Center is a feature in iOS that shows you all of your app alerts in one place. Rather than needing immediate resolution, it displays notifications until the user completes an associated action. However, we can control the notification settings.
7. **Accelerometer:** An accelerometer is a device that detects changes in velocity along a single axis. A three-axis accelerometer is built into every iOS device, providing acceleration readings in each of the three axes. LIS302DL 3-axis MEMS-based accelerometer is used in the original iPhone and first-generation iPod touch.
8. **Gyroscope:** The rate at which a gadget rotates around a spatial axis is measured using a gyroscope. A three-axis gyroscope is found in many iOS devices, and it provides rotation data in each of the three axes.
9. **Accessibility:** Every Apple product and service is built with one-tap accessibility capabilities that work the way you do.
10. **Bluetooth:** Apple supplies the Core Bluetooth framework, which includes classes for connecting to Bluetooth-enabled low-energy wireless technology.
11. **Orientations:** The iOS apps can be used in both portrait and landscape modes. Apple, on the other hand, provides size classes in XCode for creating interfaces in landscape and portrait orientations.
12. **Camera Integration:** In iOS, Apple provides the AVFoundation Capture Sub-system, which is a standard high-level architecture for audio, image, and video capture.
13. **Location Services:** The Location Services enable applications and websites to access the user's device location with the user's permission. When location services are operational, the status bar displays a black or white arrow icon.
14. **Maps:** Apple offers an online mapping service that can be utilized as the iOS default map system. It has a variety of functions, such as a flyover mode. Apple's MapKit may be used to create applications that utilize maps.

Architecture of iOS:

- The iOS is the operating system created by Apple Inc. for mobile devices. The iOS is used in many of the mobile devices for Apple such as iPhone, iPod, iPad etc.
- iPhone OS architecture consists of a number of different software layers, each of which provides programming frameworks for the development of applications that run on the top of the operating system.

- The iOS architecture is layered. It contains an intermediate layer between the applications and the hardware so they do not communicate directly.
- The lower layers in iOS provide the basic services and the higher layers provide the user interface and sophisticated graphics.
- Fig. 5.8 shows layered architecture of iOS.

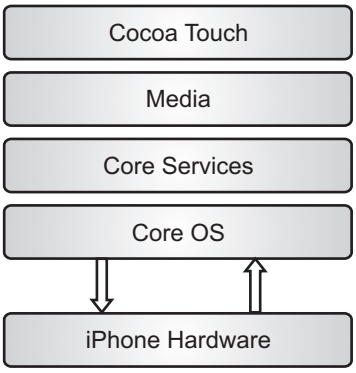


Fig. 5.8: Layered Architecture of iPhone

- The different layers of iOS layered architecture are explained below:

Core OS Layer:

- The Core layer provides a variety of services including low level networking, access to external entities, memory management, authentication and security, file system handling and threads.
- All the iOS technologies are built on the low level features provided by the Core OS layer.
- These technologies include Core Bluetooth Framework, External Accessory Framework, Accelerate Framework, Security Services Framework, Local Authorisation Framework etc.

Core Services Layer:

- The Core Services layer provides much of the foundation on which the above layers are built. There are many frameworks available in the cure services layer.
- Details about some of these services are given as follows:
  - **Cloudkit Framework:** The data can be moved between the app and the iCloud using the Cloudkit Framework.
  - **Core Foundation Framework:** This provides the data management and service features for the iOS apps.
  - **Core Data Framework:** The data model of the model view controller app is handled using the Core Data Framework.
  - **Address Book Framework:** The address book framework provides access to the contacts database of the user.

- **Core Motion Framework:** All the motion based data on the device is accessed using the core motion framework.
- **Healthkit Framework:** The health related information of the user can be handled by this new framework.
- **Core Location Framework:** This framework provides the location and heading information to the various apps.

Media Layer:

- The Media layer provides the iPhone OS with capabilities for playing, recording and editing audio-visual media. It's also responsible for rendering 2D and 3D graphics.
- The media layer enables all the graphics, audio and video technology of the system. The different frameworks are explained below:
  - **UIKit Graphics:** This provides support for designing images and animating the view content.
  - **Core Graphics Framework:** This provides support for 2-D vector and image based rendering and is the native drawing engine for iOS apps.
  - **Core Animation:** The Core Animation technology optimizes the animation experience of the apps.
  - **Media Player Framework:** This framework provides support for playing playlists and enables the user to use their iTunes library.
  - **AV Kit:** This provides various easy to use interfaces for video presentation.

Cocoa Touch Layer:

- The Cocoa Touch layer is the application development environment for building software programs to run on iOS for the iPhone.
- This layer contains key frameworks that define the appearance of the application. Cocoa Touch is primarily written in Objective-C and it is based on the standard Mac OSX Cocoa API.
- The Cocoa Touch layer provides the following frameworks:
  - **EventKit Framework** shows the standard system interfaces using view controllers for viewing and changing calendar related events.
  - **GameKit Framework** provides support for users to share their game related data online using Game center.
  - **MapKit Framework** provides a scrollable map which can be included into the app user interface.
- This layer contains some high-level features such as app extensions, which allow sharing media content to social entities, performing simple tasks with content, photo editing and providing shared storage location (for applications that use document picking).

Advantages of iOS:

1. iOS provides high customer service.
2. iOS provides excellent security which offers its users stay safe from external threats.
3. Easy access to free apps in the Apple Store and Larger number of applications availability.
4. A stable OS for different cell phones and frequent free OS updates.

Disadvantages:

1. It is not open source.
2. It supports only Apple devices.
3. iOS has closed architecture.
4. No multitasking for applications.
5. The cost of iOS apps is very high.
6. The iOS applications are larger in size as compared to other mobile platforms.

5.7.3 Android OS

- Android brought a drastic change in the mobile technology. Android is a mobile operating system which is successfully developed by Google. Android OS is written in Java language.
- The Android OS is based on Linux operating system and open source operating system which is specially developed for touch screen mobile devices like tablet, smartphones, AndroidTv, wear OS, etc.
- Android is a mobile/desktop operating system based on a modified version of the Linux kernel and other open source software, designed primarily for touch screen mobile devices such as smartphones and tablets.
- Android is an open source and Linux-based Operating System for mobile devices such as smartphones and tablet computers. Android was developed by the Open Handset Alliance, led by Google.
- Android OS is written in Java language. Android brought a drastic change in mobile technology.
- Android applications can be packaged easily and sold out either through a store such as Google Play, SlideME, Opera Mobile Store, Mobango, F-droid and the Amazon Appstore.

Features of Android OS:

1. **Open Source:** Android is a freely available operating system. So that all the people can work on android not only the one company developers.
2. **Storage:** Android uses SQLite, a lightweight relational database used for data storage purposes.

3. **Beautiful UI:** Android OS basic screen provides a beautiful and intuitive User Interface (UI).
4. **Connectivity:** Supports GSM/EDGE, Evolution-Data Optimized EV-DO, UniversalMobile Telecommunication System, Bluetooth, WiFi, IDEN, CDMA, LTE, and WiMAX.
5. **Messaging:** It supports both messaging SMS and MMS.
6. **Media Support:** Android includes support for the media like H.263, H.264 (in 3GP or MP4 container), MPEG-4 SP, MP3, MIDI, WAV, JPEG, PNG, GIF and BMP.
7. **Multitasking:** Android supports different multitasking applications like displaying two apps side by side in a single user interface design.
8. **Multi-touch:** Android supports multi-touch facility.
9. **Tethering:** It also shares the Internet with the mobiles and other devices by using a wired/wireless hotspot facility.
10. **Voice-based Features:** Google search through voice is also being developed and available.

Architecture of Android:

- Fig. 5.9 shows layered architecture of Android OS.
- The Linux kernel acts as an abstraction layer between the hardware and the rest of the software stack.
- Android runtime includes core libraries and Dalvik Virtual Machine (DVM). Core libraries have a set of libraries to provide the functionality of Java language. Every application runs on its own Dalvik VM.
- Android has a set of C/C++ libraries used by various components of the operating system. It ships with a set of core applications that offers developers the ability to build various applications with an open development.

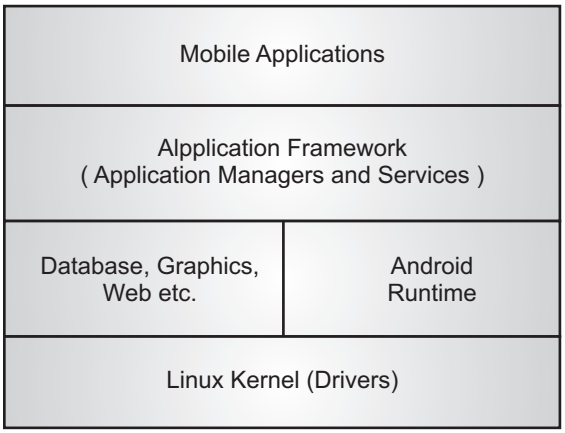


Fig. 5.9: Architecture of Android Operating System

Advantages of Android OS:

1. Android is based on the Linux kernel. This facilitates easy accessibility to a rich development environment and core functionality of the mobile device.
2. It provides rich browser facilities as well. This facilitates the developer to provide enhanced services.
3. Android OS is open source and it is free of cost.
4. It is a multitasking OS and can be virtualized.
5. It is a flexible OS. Widgets (or self-contained programs), add functionality and flexibility to Android devices. The development tools are easy to use.
6. Stability and security is better than other mobiles OS as it is based on Linux Kernel.
7. Android is highly customizable. Users can change almost everything in the UI.

Disadvantages of Android OS:

1. Android OS can also slow down if installing more apps.
2. In android OS the wastage of battery is more due to the background processing.
3. Numbers of the Android applications have low security to steal sensitive information from unknown resources.
4. There is no standard for Android applications.
5. Android OS updates are controlled by device manufacturers and may be slow or non-existent.
6. The complexity involved in Android app development makes the process more time-consuming than that of an iOS app.

5.8 COMPARATIVE STUDY OF MOBILE OPERATING SYSTEMS

- In this section we will discuss comparative study of Mobile Operating Systems like Palm OS, Android, Symbian OS, Blackberry OS and Apple iOS.

5.8.1 Palm OS

- Palm OS was a proprietary mobile operating system. Designed in 1996 for Palm Computing, Inc.'s new Pilot PDA, it has been implemented on a wide array of mobile devices, including smartphones, wrist watches, handheld gaming consoles, barcode readers and GPS devices.
- Palm OS is a fully ARM-native, 32-bit operating system designed for use on Palm handhelds and other third-party devices.
- Its popularity can be attributed to its many advantages, such as its long battery life, support for a wide variety of wireless standards and the abundant software available.
- Palm OS is the underlying operating system for Palm PDAs. Palm also licenses Palm OS to other PDA manufacturers such as Handspring (merged with Palm in 2003), Sony and IBM.

- Officially, the company that develops Palm OS is Palm Source, a spin-off from PalmOne, the new name of Palm, Inc., after it merged with Handspring.
- Palm operating system is especially designed for PDAs and handheld devices. Following are the two major versions of Palm OS are currently under development:
  1. **Palm OS Garnet** is an enhanced version of Palm OS 5 and provides features such as dynamic input area, improved network communication and support for a broad range of screen resolutions.
  2. **Palm OS Cobalt** is Palm OS 6, which focuses on enabling faster and more efficient development of smartphones and integrated wireless (WiFi/Bluetooth) handhelds.
- The key **features of the current Palm OS Garnet** are:
  1. Simple, single-tasking environment to allow launching of full screen applications with a basic, common GUI set
  2. Monochrome or color screens with resolutions up to 480×320 pixel.
  3. It has an elementary memory management system. To keep the operating system small and fast, Palm OS does not isolate the memory areas of applications from each other. Consequently, any misbehaving application can crash the system.
  4. Palm supplies Palm emulator, which emulates the Palm hardware on a PC. This allows Palm programs to be developed and debugged on a PC before being run on the Palm hardware.
  5. Handwriting recognition input system called Graffiti 2.
  6. HotSync technology for data synchronization with desktop computers.
  7. Sound playback and record capabilities.
  8. Simple security model: Device can be locked by password, arbitrary application records can be made private.
  9. TCP/IP network access.
  10. Serial port/USB, infrared, Bluetooth and Wi-Fi connections.
  11. Expansion memory card support.
  12. Defined standard data format for personal information management applications to store calendar, address, and task and note entries, accessible by third-party applications.
- Fig. 5.10 shows the structure of Palm OS5, which consists of five layers namely, applications, PACE, core Palm OS and licensee libraries, DAL and HAL.
- Palm OS allows third-party hardware to be used as part of the system provided the hardware abstraction layer and system services support the third party hardware.
- The kernel of Palm OS is based on AMX, licensed from Kodak, which offers preemptive multitasking and protective memory management.
- System services are a set of modular components that provide communication, input method, GUI event handling and multimedia processing. Palm OS 6 has built-in Bluetooth and Wi-Fi support.

- Core OS libraries supply low-level file system management functionality and TCP/IP. Third party software libraries can also be plugged into the system; for example, J2ME profile implementations can be added as third-party libraries.
- The PACE layer emulates non-ARM processors, thus allowing legacy applications to execute.
- Palm OS allows third-party hardware to be used as part of the system provided the hardware abstraction layer and system services support the third party hardware.
- The kernel of Palm OS is based on AMX, licensed from Kodak, which offers preemptive multitasking and protective memory management.
- System services are a set of modular components that provide communication, input method, GUI event handling, and multimedia processing.
- Palm OS 6 has built-in Bluetooth and Wi-Fi support. Core OS libraries supply low-level file system management functionality and TCP/IP.
- Third party software libraries can also be plugged into the system; for example, J2ME profile implementations can be added as third-party libraries.
- The PACE layer emulates non-ARM processors, thus allowing legacy applications to execute.

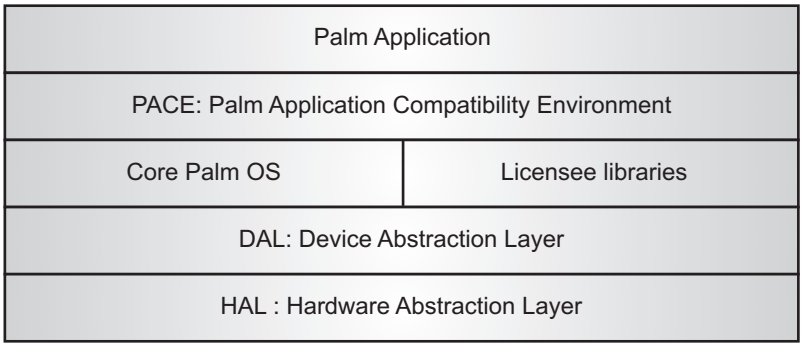


Fig. 5.10: Architecture of Palm OS5

5.8.2 Android OS

- Android is an open source and Linux-based Operating System for mobile devices such as smartphones and tablet computers.
- Android OS explained in detail in Section 5.7.3.

5.8.3 Symbian OS

- Symbian operating system was developed through collaboration among a few prominent mobile device manufacturers including Nokia, Ericsson, Panasonic, and Samsung.
- Symbian mobile OS is an open-source OS designed for smartphones. Symbian OS is a 32-bit, little-endian operating system, running on different flavors of ARM Architecture.

- Symbian mobile OS is a multitasking operating system and very less dependent on peripherals.
- Symbian OS was originally based on the EPOC operating system and was mainly used for PDAs developed by Psion. Interestingly, Symbian OS began with version 6.0, following EPOC version 5.0.
- The latest version of Symbian OS is 9.0. Symbian OS defines a few platforms of User Interface (UI) reference models to accommodate disparate mobile devices.
- For example, the Quartz model targets PDA-like devices, the Crystal model is for common cell phones, the UIQ model is for customizable pen-based touch-screen mobile devices and the Series 60 model is for numerical keyboard high-end cell phones and smartphones.

Features of Symbian OS:

1. **Real-time:** Symbian OS has a real-time, multithreaded kernel.
2. **Optimized Memory Management:** It supports preemptive multitasking scheduling and memory protection.
3. **Data Caging:** Symbian OS allows applications to have their own private data partition. This feature allows for applications to guarantee a secure data store. It can be used for e-commerce applications, location aware applications etc.
4. **Platform Security:** Symbian OS provides a security mechanism against malware. It allows sensitive operations to be accessed by applications which have been certified by a signing authority. In addition, it supports full encryption and certificate management,
5. **Secure Protocols:** It supports a number of communication and networking protocols including TCP, UDP, PPP, DNS, FTP, WAP, etc. For personal area networking, it supports Bluetooth, InfraRed and USB connectivity (HTTPS, TLS and SSL) and WIM framework.
6. **Multimedia:** Symbian OS supports audio, video recording, playback and streaming and Image conversion.
7. **Internationalization Support:** Symbian OS supports Unicode standard.
8. **Client-server Architecture:** It provides simple and high-efficient inter process communication. This feature also eases porting of code written for other platforms to Symbian OS.

Architecture of Symbian OS:

- Symbian OS is a private, independent company that develops and supplies the open standard mobile operating system Symbian OS.
- It is owned by some large cell phone manufacturers including Nokia, Ericsson, Sony Ericsson, Siemens and Samsung. The primary targets of Symbian OS are cell phones and smartphones.

- Initially, Symbian transport layer security (WTLS) and IPSec. Symbian OS supports ARM processors and x86 emulation.
- Fig. 5.11 shows the architecture of Symbian OS version 8.1.
- The Symbian operating system is based on the java language. It combines middleware of wireless communications and personal information management (PIM) functionality.
- Symbian OS is a real-time, multithreaded, preemptive kernel (versions prior to 8.0 do not provide real-time kennels) that performs memory management, process and thread scheduling, inter-process communication, process-relative and thread-relative resource management, hardware abstraction, and error handling. The kernel runs natively in the ARM core.
- The basic services provide a programming framework for Symbian OS components, such as kernel and user API library, device drivers, file systems and standard C++ library.
- On top of the basic services is a set of communications services, multimedia services, PC connectivity services, and generic OS services. Communication services act as the core to mobile telephony and data network access applications.
- Personal area network (PAN) connectivity such as Bluetooth, IrDA, and USB is also enabled by these services. Multimedia services deal with audio and video recording, playback, and streaming.
- Connectivity services are software components that implement PC synchronization. Generic OS services offer typical OS-related components such as memory management and file system access.
- Application services allow user programs to be executed in separate processes. An exception handling mechanism is also provided, as well as internationalization support.
- The UI framework is consists of an array of UI components and an event-handling mechanism that permit easy porting of UI programs between different Symbian OS devices.
- Symbian OS uses EPOC C++, a pure object-oriented language, as the supporting programming language for both system services implementations and application programming interfaces.
- It also allows Java applications for mobile devices (J2ME applications} to run on top of a small Java runtime environment. Symbian OS implements a CLDC/MIDP 2.0 profile of J2ME specifications.

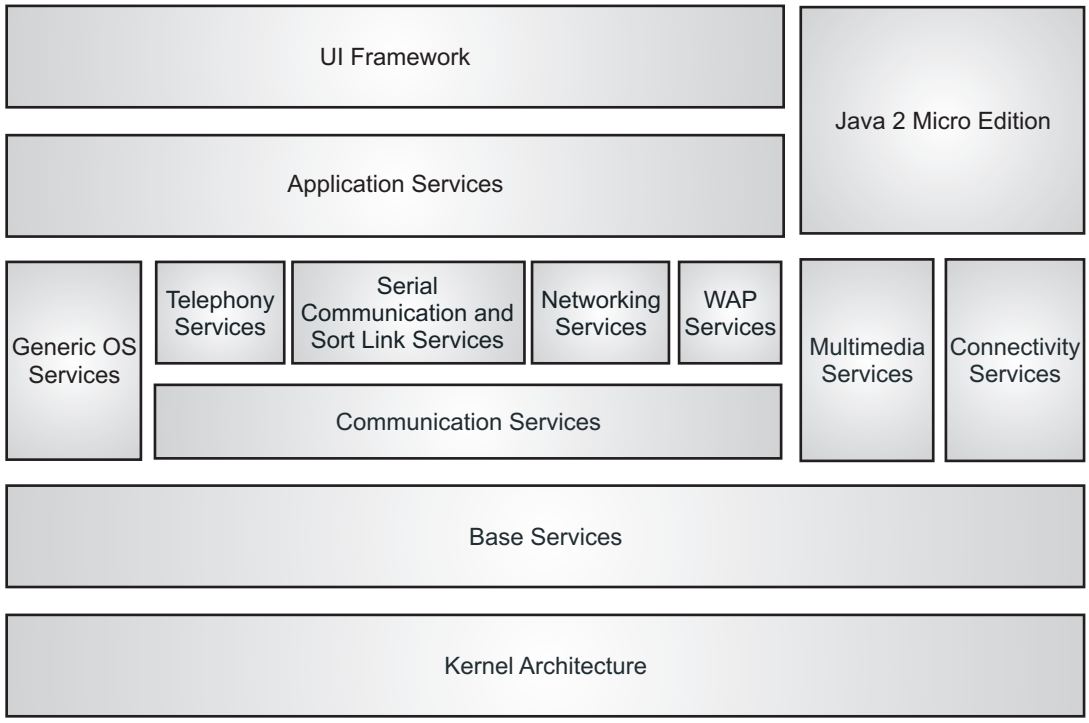


Fig. 5.11: Architecture of Symbian Operating System

- The layers in the Symbian OS are explained below:
  1. **UI (User Interface) Framework:** The UI application-framework subsystem provides an environment for creating differentiated user interfaces and customization of Look And Feel (LAF). It is architected to be even-driven and widget based and also enables sharing screen, keyboard and pointer between applications.
  2. **Application Services:** Symbian OS provides application engines for PIM (Personal Information Management), messaging, content handling, Internet and Web application, data synchronization services and provisioning services.
  3. **Java:** Symbian OS provides a Java application execution environment optimized for smartphones.
  4. **OS Services:** They provide vital operating system infrastructure components. OS services include multimedia and graphics subsystems, networking, telephony, short link protocols, cryptography libraries and PC connectivity infrastructure.
  5. **Communication Services:** Symbian OS communications services subsystem provides frameworks and system services for communications and networking. A communications database manager controls system-wide communications configuration. Socket server and client-side APIs provide a framework for implementing various communications protocols through a socket interface.

6. **Multimedia and Graphics Services:** Symbian OS Multi-Media Framework (MMF) provides a multithread framework for handling multimedia data. It provides commonly used functionality including audio playback, audio recording, audio streaming and audio conversion.
7. **Connectivity Services:** Connection manager and PC-connectivity toolkit comprise the Symbian OS connectivity services subsystem. Symbian OS connection manager manages connections between a PC and a Symbian OS phone using TCP/IP protocols for data transfer.
8. **Base Services:** Symbian OS base subsystem provides the programming framework for all other components of Symbian OS. The main user visible elements are the file system and common user libraries. Symbian OS base subsystem also contains middleware widely used across Symbian OS. It also implements the C standard library and provides relational database access API.
9. **Kernel Services and Hardware Abstraction Interface:** This facilitates the porting of Symbian OS to new hardware. Symbian OS kernel is responsible for scheduling, memory management and power management and offers process and thread creation along with application error handling and cleanup framework. Event-driven multitasking via active objects enables efficient multithreading.

5.8.4 BlackBerry OS

- Blackberry OS is a proprietary mobile operating system developed by Research In Motion (RIM) for its BlackBerry line of smartphones.
- The BlackBerry operating system is a mobile operating system and was designed specifically for BlackBerry handheld devices.
- Research In Motion® (RIM) is a Canadian designer, manufacturer and marketer of wireless solutions for the worldwide mobile communications market.
- Products include the BlackBerry wireless email solution, wireless handhelds and wireless modems.
- RIM is the driving force behind BlackBerry smartphones and the BlackBerry solution. RIM provides proprietary multitasking.
- OS for the BlackBerry, which makes heavy use of specialized input devices, particularly the scroll wheel or more recently the trackball.
- The BlackBerry platform is perhaps best known for its native support for the corporate communication environment, which allows complete wireless activation and synchronization of email, calendar, tasks, notes, and contacts.
- The operating system provides multitasking and supports specialized input devices that have been adopted by BlackBerry Ltd. for use in its handhelds, particularly the track wheel, trackball, and most recently, the trackpad and touch screen.

- The BlackBerry platform is perhaps best known for its native support for corporate email, through MIDP 1.0 and more recently, a subset of MIDP 2.0, which allows complete wireless activation and synchronization with Microsoft Exchange, Lotus Domino, or Novell.
- GroupWise email, calendar, tasks, notes, and contacts, when used with BlackBerry Enterprise Server. The operating system also supports WAP 1.2.
- Updates to the operating system may be automatically available from wireless carriers that support the BlackBerry over the air software loading (OTASL) service.
- Third-party developers can write software using the available BlackBerry APIclasses, although applications that make use of certain functionality must be digitally signed.
- Research from June 2011 indicated that approximately 45% of mobile developers were using the platform at the time of publication.
- BlackBerry OS was discontinued after the release of BlackBerry 10, but BlackBerry will continue support for the BlackBerry OS.
- BlackBerry mobile OS offers the best combination of mobile phone, server software, push email and security from a single vendor.
- BlackBerry mobile OS integrates well with other platforms, it works with several carriers and it can be deployed globally for the sales force which is on the move.
- It is easy to manage, has a longer than usual battery life, and has a small form-factor with an easy-to-use keyboard.
- BlackBerry is good for access to some of the simpler applications, such as contact list, time management and field force applications.
- Fig. 5.12 shows architecture for Blackberry OS. The BlackBerry OS is a modern OS that includes features such as multitasking, interprocess communication and threads.
- The BlackBerry OS is developed by RIM and it is designed in such a way to take inputs from input devices such as trackpad, trackball and track wheel.
- The BlackBerry OS supports the Java Mobile Information Device Profile (MIDP) 1.0 and Wireless Application Protocol (WAP) 1.2.

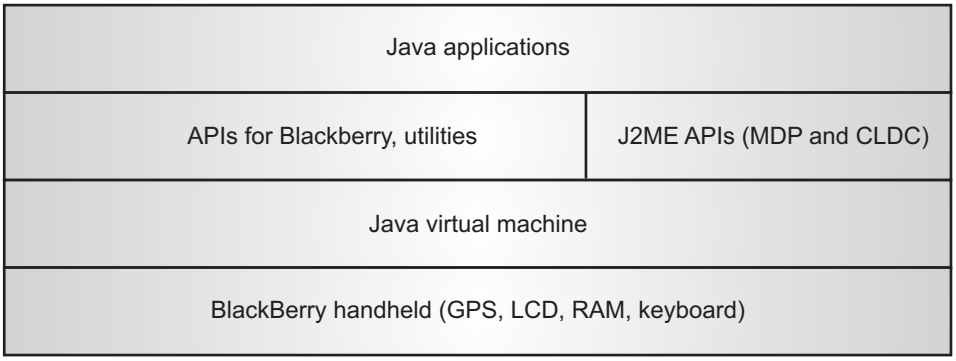


Fig. 5.12: Architecture of BlackBerry Operating System

- **Advantages:** Faster web browsing, Handy for reading mails, and Multi tasking.
- **Disadvantages:** Memory Manager doesn't release memory even after the applications are closed which leads to slow down of the device. Application Distribution is more difficult.

5.8.5 Apple iOS

- The iOS (iPhone OS) was developed by Apple Inc., for the use on its device. The iOS operating system is the most popular operating system today.
- It is a very secure operating system. The iOS operating system is not available for any other mobiles.
- The iOS OS explained in detail in Section 5.7.2.

A comparative summary of the important features of different mobile OSs has been presented below in table:

Parameters	Android	iOS	Symbian	Windows Mobile	BlackBerry	PalmOS
OS family	Linux	Darwin	RTOS	Windows CE-7, Windows NT8	QNX	Garnet OS/Linux
Vendor	Open Handset Alliance, Google	Apple	Symbian Ltd. and Symbian foundation	Microsoft	Blackberry Ltd.	Palm incorporation
Environment IDE	Eclipse(Google)	XCode(Apple), Appcode	QT, Carbide C++, Vistamax, Eclipse	Visual Studio	Eclipse, Blackberry JDE	CodeWarrior
CPU Architecture	ARM, x86, MIPS	ARM 64, ARM	ARM, x86	ARM	ARM	ARM
Source Model	Open source	Closed source	closed source	Closed source	Closed source	Closed source
License	Free, Open source	Proprietary	Proprietary	Proprietary	Proprietary	Proprietary
Written In	C, C++, Java	C, C++, Objective –C, Swift	C, C++, ME, Python, Ruby, Flash Lite	C#, VB.net, c++, F#, JScript	C, C++, HTML 5, JS , CSS, Java, Action Script	C, C++
Memory utilization	Paging, memory map, no swapping	Automatic reference counting, no garbage collection	Memory management unit , cache resides on SOC	RAM/ROM flash memory is used virtual memory management, program run from main memory	Contain slot for external memory, supports for 32GB micro SD card	The Memory, RAM and ROM, for each Palm resides on a memory module known as card
Cross platform	Supports	Not supporting	Supports	Supports	Not supporting	Supports
Application store	Google play	App store	Nokia Ovi store	Windows Phone	Blackberry world	Software store

PRACTICE QUESTIONS

Q. I Multiple Choice Questions:

1. Which operating system allows the user to run other different application software on the mobile, PDAs, tablets, etc.?
   (a) mobile (b) grid
   (c) cloud (d) cluster
2. Which of the following is/are mobile operating system?
   (a) Android (b) Symbian
   (c) iOS (d) All of the mentioned
3. A mobile operating system can be found on all these devices except,
   (a) smartphones (b) PDAs
   (c) computer (d) tablets
4. In which language Android mobile OS applications are usually developed.
   (a) Objective-C (b) Java
   (c) Python (d) Ruby
5. Which mobile OS brought a drastic change in the mobile technology?
   (a) Android (b) Symbian
   (c) iOS (d) Blackberry
6. ARM architecture is based on,
   (a) CISC (b) RISC
   (c) Both (a) and (b) (d) None of the mentioned
7. Which mobile operating system designed for the iPhone and iPad?
   (a) Android (b) BlackBerry
   (c) iOS (d) Windows Mobile
8. Mobile devices are,
   (a) current powered devices (b) battery powered devices
   (c) solar powered devices (d) None of the mentioned
9. Which mobile operating system was developed through a collaboration among a few prominent mobile device manufacturers including Nokia, Ericsson, Panasonic and Samsung?
   (a) Android (b) BlackBerry
   (c) iOS (d) Symbian
10. Which is the core of an operating system?
    (a) kernel (b) application
    (c) hardware (d) None of the mentioned

11. Which is computer programming that is compiled to run with a particular processor (such as an Intel x86-class processor) and its set of instructions?
    (a) Execution code (b) Compiler code
    (c) Native code (d) None of the mentioned
12. The issue is a problem that happens during the execution of a program called as,
    (a) compile time issue (b) runtime time issue
    (c) Both (a) and (b) (d) None of the mentioned
13. Which is a proprietary mobile operating system developed by Research In Motion (RIM)?
    (a) Android (b) Symbian
    (c) iOS (d) BlackBerry
14. Which mobile OS is based on the Windows CE kernel and first appeared as the Pocket PC 2000 operating system?
    (a) Windows Mobile (b) BlackBerry
    (c) iOS (d) Android
15. Which power management approach is used for managing the power of the underlying platform and switching it between different power states?
    (a) APM (b) OSPM
    (c) ACPI (d) None of the mentioned
16. The iOS is written in which language,
    (a) HTML, CSS, Angular JS (b) C#, Node JS, Objective-C, Swift
    (c) Redux, Swift (d) C, C++, Objective-C, Swift
17. Which underlying operating system for Palm PDAs?
    (a) Android (b) Plam OS
    (c) Windows Mobile (d) iOS
18. Which one of the following architecture has large number instructions?
    (a) CISC (b) RISC
    (c) Both (a) and (b) (d) None of the mentioned

Answers

1. (a)	2. (d)	3. (c)	4. (b)	5. (a)	6. (b)	7. (c)	8. (b)	9. (d)	10. (a)
11. (c)	12. (b)	13. (d)	14. (a)	15. (b)	16. (d)	17. (b)	18. (a)		

Q. II Fill in the Blanks:

1. A _____ operating system is an operating system that helps to run other application software on mobile devices such as tablets, smartphones, laptops etc.
2. Intel plays a major role in the mobile space with its _____ processor architecture.
3. The Android mobile OS, developed by Google Inc., is based on _____ kernel.

4. _____ processors are 32-bit RISC processors which aha very low power consumption.
5. _____ OS is the underling OS for Plam PDAs.
6. Linux is typically considered a _____ -kernel operating system.
7. Mobile devices have constraints and restrictions on their physical characteristic such as _____ Memory, Screen Size, Processing Power and Battery Power.
8. Windows Phone mobile OS developed by _____.
9. The _____ is a mobile operating system created and developed by Apple Inc.
10. _____ mobile OS is a proprietary mobile operating system developed by Research In Motion (RIM) for its BlackBerry smartphones.

Answers

1. mobile	2. x86	3. Linux	4. ARM
5. Plam	6. monolithic	7. limited	8. Microsoft
9. iOS	10. Blackberry		

Q. III State True or False:

1. A mobile operating system is an system software that is specifically designed to run on mobile devices such as mobile phones, PDAs, tablet computers.
2. The Android OS is based on Linux is open source mobile operating system for smartphones and tablets.
3. Plam mobile OS developed by Apple Inc.
4. ARM chips are usually slower than their Intel chips because they are designed to compute with low power consumption.
5. A mobile device usually has large permanent and volatile storage compared to that of a contemporary desktop or laptop.
6. Mac OS is a mobile operating system for iPhone.
7. The Symbian operating system is based on the Java language.
8. The hardware connects the application software to the hardware of a computer.
9. Intel's x86 processor architecture uses CISC.
10. Windows 7 is a mobile operating system.
11. System on a Chip (SoC) processors, as the name suggests, integrate many different functions into a single integrated circuit.
12. Power management broadly refers to the management of power-related activates in a mobile device, which include consumption of power in mobile device.
13. Mobile OS typically employ micro kernel approach for provisioning OS services.

Answers

1. (F)	2. (T)	3. (F)	4. (T)	5. (F)	6. (F)	7. (T)	8. (F)	9. (T)	10. (F)
11. (T)	12. (T)	13. (T)							

Q. IV Answer the following Questions:

(A) Short Answer Questions:

1. Define mobile OS?
2. List special constraints of mobile operating system.
3. Mention the requirements of mobile operating system.
4. List the important features of the Android mobile OS.
5. What is ARM stands for?
6. List approaches for power management.
7. What is main difference RISC and CISC?
8. Which CPU architecture is mostly used in mobile OS?
9. What are the main components of android architecture?
10. List the features of the iOS mobile OS.
11. Define power management.
12. Define native level programming.
13. List the important features of the Palm mobile OS.

(B) Long Answer Questions:

1. What is mobile operating system? What are its responsibilities? Also list popular mobile OSs.
2. What are special constraints of mobile operating system?
3. What are the requirements of mobile operating system? Describe in detail.
4. Why power management is important in mobile OS? What are its different approaches? Explain two of them in detail.
5. Compare desktop OS and mobile OS.
6. Compare ARM and Intel architecture for power management.
7. Explain the Android mobile architecture with diagram.
8. What Palm OS? List its architecture diagrammatically.
9. What are the important features of the Windows Phonemobile OS?
10. What is iOs mobile OS? Give its architecture. Also state its advantages and disadvantages.
11. What is Symbian operating system? Give its architecture with diagram.
12. With the help of diagram describe BlackBerry mobile OS.
13. What is native level programming? Describe in detail.
14. Write a short note on: Runtime issues.
15. What are the special service requirements for mobile OS?
16. Differentiate between ARM and x86 processors architectures.

■■■