KEC'S

A Complete TU Solution

& Practice SETS

BCA (IV Semester)

All Subjects

with Model Questions-Answers

FOR 2078

Sailent Features of this book

- **New Syllabus**
- **Model Questions-Answers**
- **TU Questions-Answers**
- **Model Questions Sets**
- **Answer to the Numerical Problems**

CONTENTS

Operating System

Course Title: Operating System (3 Cr.)

Corse Code: CACS251

Year/Semester: II/IV

Class Load: 6Hrs. / Week (Theory: 3 Hrs, Tutorial: 1, Practical: 2 Hrs.)

Course Description

This course includes the topics that help students understand operating system and it's functionality along with its types.

Course Objectives

The general objectives of this subject are to provide the basic feature, function and interface with the hardware and application software to run the computer smoothly.

Course Contents

Unit 1 Introduction to Operating system **2 Hrs**

History, introduction and generation of Operating system, Objectives (Resource Manager and Extended Machine), types of Operating system, Function of Operating system.

Unit 2 Operating System Structure **2 Hrs**

Introduction, Layered System, Kernel, Types of Kernel (Monolithic / Macro Kernel and Micro / Exo-kernel), Client-server Model, Virtual Machines, Shell

Unit 3 Process Management **15 Hrs**

Process concepts (3 Hrs.): Definitions of Process, The process Model, Process States, Process State Transition, The process Control Block, Operations on Process (Creation, Termination, Hierarchies, Implementation), Cooperating Processes, System Calls (Process Management, File Management, Directory Management).

Threads (1 Hrs.): Definitions of Threads, types of Thread Process (Single and Multithreaded process), Benefits of Multithread, Multithreading Models (Many-to-One Model, One-to-One Model, Many- to- Many Model).

Inter Process Communication and Synchronizations (6 Hrs.): Introduction, Race Condition, Critical Regions, Avoiding Critical Region: Mutual Exclusion and serializability; Mutual Exclusion Conditions, proposals for Achieving Mutual Exclusion: Disabling Interrupts, Lock Variable, Strict Alteration (Peterson's Solution), The TSL Instruction, Sleep and Wakeup, Types of Mutual Exclusion (Semaphore, Monitors, Mutexes, Message Passing, Bounded Buffer), Serializability: Locking Protocols and Time Stamp Protocols: Classical IPC Problems (Dining Philosophers Problems, The Readers and Writers Problem, The Sleeping Barber's Problem)

Process Scheduling (5 Hrs.): Basic Concept, Types of Scheduling (Preemptive Scheduling, Non-Preemptive Scheduling, Batch, Interactive, Real Time Scheduling), Scheduling Criteria or Performance Analysis, Scheduling Algorithm (Round-Robin, First Come First Served, Shortest job first, Shortest Process Next, Shortest Remaining time Next, Real Time, Priority Fair share, Guaranteed, Lottery Scheduling, HRN, Multiple Queue, Multilevel Feedback Queue); Some Numerical Examples on Scheduling.

Unit 4 Deadlocks **(4 Hrs)**

System Model, System Resources: Preemptable and Non-Preemptable; Conditions for Resource Deadlocks. Deadlock Modeling, The OSTRICH Algorithm, Method of Handling Deadlocks, Deadlock Prevention, Deadlock Avoidance: Banker's Algorithm, Deadlock Detection: Resource Allocation Graph, Recovery from Deadlock.

Unit 5 Memory Management **(7 Hrs)**

Basic Memory Management (3 Hrs.): Introduction, Memory hierarchy, Logical versus Physical Address space, Memory management with Swapping: Memory Management with Bitmaps and with Linked List; memory Management without Swapping, Contiguous-Memory Allocation: Memory protection, Memory Allocation, Fragmentation (Inter Fragmentation, External Fragmentation); Non-Contiguous Memory Allocation, Fixed Partitioning Vs. Variable Partitioning, Relocation and Protection, Coalescing and Compaction.

Virtual Memory (4 Hrs.): Background, Paging, Structure of Page Table: Hierarchical Page Table, Inverted Page Table, Shared page table; Block mapping Vs. Direct Mapping, Demand Paging, Page Replacement and Page Faults, Page Replacement Algorithms: FIFO, OPR, LRU, SCP; Some Numerical Examples on page Replacement, Trashing, Segmentation, Segmentation with paging.

Unit 6 Input / Output Device Management **(4 Hrs)**

Principle of I/O Hardware: I/O devices, Device Controllers, Memory Mapped I/O, Direct Memory Access; Principle of I/O Software: Goals of I/O Software, Program I/O, Interrupt Driven I/O, I/O using DMA; I/O Software Layers: Interrupt Handler, Device Drivers, Device Independent I/O Software, User space I/O Software; Disk: Disk Hardware, Disk Scheduling: Seek Time, Rational Delay, Transfer Time; Disk Scheduling Algorithms: FCFS Scheduling, SSTF scheduling, SCAN Scheduling, C-SCAN Scheduling, Lock Scheduling.

Unit 7 File System Interface Management **(2 Hrs)**

File Concept: File naming, File Structure, File Type, File Access, File Attributes, File Operation and File Descriptors; Directories: Single-Level Directory system, Hierarchical Directory Systems, Path names, Directory Operations; Access Methods: Sequential, Direct; Protection: Types of Access, Access Control List, Access Control Matrix.

Unit 8 Security Management **(3 Hrs)**

Introduction, Security Problems, User Authentication: Passwords, Password Vulnerabilities, Encrypted Password, One Time Password and Biometrics Password; User Authorizations, Program Threats: Trojan Horse, Trap Door, Stack and Buffer Overflow; System Threats: Worms Viruses, Denial of Services.

Unit 9 Distributed Operating System **(4 Hrs)**

Introduction, Advantages of Distributed System over Centralized System, Advantages of Distributed System Over independent PCs. Disadvantages of Distributed System, Hardware and Software Concepts, Communication in Distributed Systems, Message Passing, Remote Procedure Call, Process in Distribution System, Clock Synchronization

Unit 10 Case Study **(2 Hrs)**

DOS and Windows Operating System, UNIX Operating System, Linux Operating System

Laboratory Work

Lab works should be done covering all the topics listed above and a small project work should be carried out using the concept learnt in this course. Project should be assigned on individual basis.

TRIBHUVAN UNIVERSITY

Faculty of Humanities & Social Sciences

OFFICE OF THE DEAN

TU MODEL QUESTIONS-ANSWERS/ TU QUESTIONS-ANSWERS 2019

Course Title: Operating System — Full Marks: 60
Course No: CACS251 — Pass Marks: 24
Nature of the Course: Theory + Lab — Time: 3
Semester: IV

Candidates are required to answer the questions in their own words as far as possible.

Group A

Attempt all the questions. **[10×1 = 10]**

1. Circle (O) the correct answer in the following questions

i) In UNIX, which system call creates the new process?
a) Fork b) create c) new d) none of the mentioned

ii) Which of the following does not contain process control block (PCB)?
a) Code b) Bootstrap program c) Stack d) Data

iii) The Test and Set instruction in executed:
a) After a particular process b) atomically
c) periodically d) none of these

iv) How circular wait condition can be prevented?
a) By defining linear ordering of resource type
b) By resource grant on all or none basis
c) By using pipes d) By using threads

v) Which process can be affected by other processes executing in the system?
a) Cooperating process b) Init Process
c) Parent process d) Child Process

vi) External fragmentation will not occur when?
a) First fit is used b) Worst fit is used
c) Best fit is used
d) No matter which algorithm is used, it will always occur

vii) What is compaction?
a) a technique for overcoming internal fragmentation
b) a technique for overcoming external fragmentation
c) a paging technique
d) a technique for over coming fatal error

viii) Why is one-time password safe?

a) It is easy to generated
b) It is different for every access
c) It cannot be shared
d) It is complex encrypted password

ix) In distributed system each processor has its own:

a) Local memory
b) Clock
c) Both local memory and clock
d) none of these

x) How process on the remote system are identified:

a) Host ID
b) Host name and identifier
c) Identifier
d) Process ID

Answer-Key

i. (a)	ii. (b)	iii. (b)	iv. (a)	v. (a)	vi. (d)	vii. (b)	viii. (b)	ix. (a)	x. (b)

Group B

Attempt any SIX questions. **[6x5 = 30]**

2. What is an operating system? Explain the functions of operating system.

Ans: An operating system is an important part of almost every computer system. This can be divided roughly into four components: the hardware, the operating system, the application programs and the users.

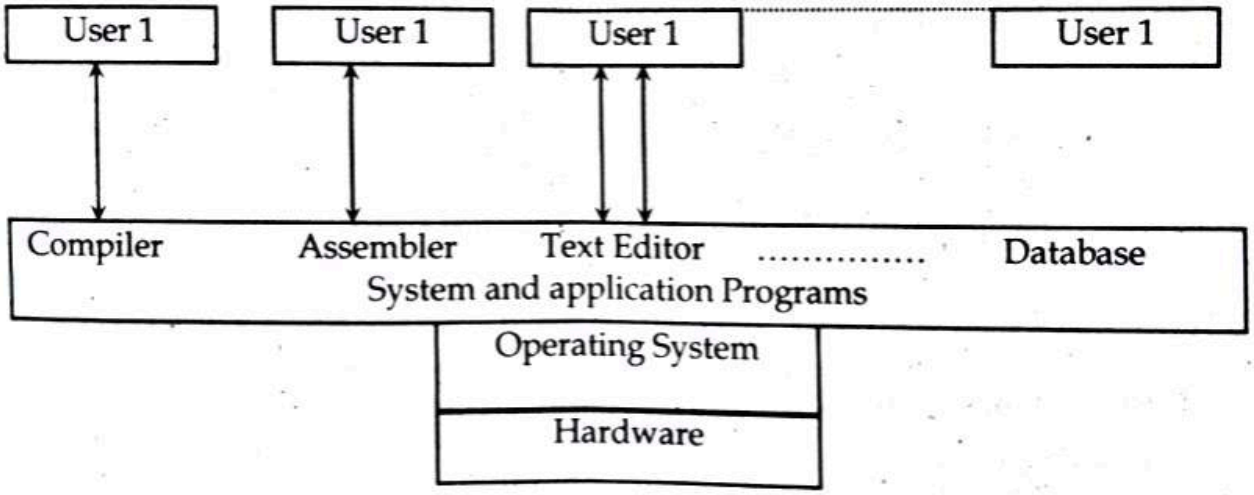


Fig: An overview of Complex System

Hardware, provides basic computing resources (CPU, Memory, I/O devices etc.) whereas Operating System controls and co-ordinates the use of the hardware among the various users. It is usually that portion of software that runs in kernel mode or supervisor mode. The application Programs define the way in which the system resources are used to solve the computing problems and users are the People, machine, other computers or processes. Some common operating systems are desktop operating system; network operating system; mobile operating system; embedded system operating system etc. Examples of operating system are Windows, Android, iOS, Mac OS, Linux, Chrome OS, and Windows Phone OS etc.

The advantage of using Operating System

- Allows you to hide details of hardware by creating an abstraction
- Easy to use with a GUI

- Offers an environment in which a user may execute programs/applications
- The operating system must make sure that the computer system convenient to use
- Operating System acts as an intermediary among applications and the hardware components
- It provides the computer system resources with easy to use format
- Acts as an intermediator between all hardware's and software's of the system

Function of Operating system

An Operating System acts as a communication bridge (interface) between the user and computer hardware. The purpose of an operating system is to provide a platform on which a user can execute programs in a convenient and efficient manner.

The main task an operating system carries out is the allocation of resources and services, such as allocation of: memory, devices, processors and information. The operating system also includes programs to manage these resources, such as a traffic controller, a scheduler, memory management module, I/O programs, and a file system. Important functions of an operating System:

- **Security**

  The operating system uses password protection to protect user data and similar other techniques. It also prevents unauthorized access to programs and user data.

- **Control over system performance**

  Monitors overall system health to help improve performance. Records the response time between service requests and system response to have a complete view of the system health. This can help improve performance by providing important information needed to troubleshoot problems.

- **Job accounting**

  Operating system keeps track of time and resources used by various tasks and users, this information can be used to track resource usage for a particular user or group of user.

- **Error detecting aids**

  Operating system constantly monitors the system to detect errors and avoid the malfunctioning of computer system.

- **Coordination between other software and users**

  Operating systems also coordinate and assign interpreters, compilers, assemblers and other software to the various users of the computer systems.

- **Memory Management**

  The operating system manages the primary memory or main memory. Main memory is made up of a large array of bytes or words where each byte or word is assigned a certain address. Main memory is a fast storage and it can be accessed directly by the CPU. For a program to be executed, it should be first loaded in the main memory. An Operating System performs the following activities for memory management:

  It keeps tracks of primary memory, i.e., which bytes of memory are used by which user program. The memory addresses that have already been allocated and the memory addresses of the memory that has not yet been used. In multi programming, the OS decides the order in which process are granted access to memory, and for how long. It allocates the memory to a process when the process requests it and de-allocates the memory when the process has terminated or is performing an I/O operation.

- **Processor Management**

  In a multi programming environment, the OS decides the order in which processes have access to the processor, and how much processing time each process has. This function of OS is called process scheduling. An Operating System performs the following activities for processor management.

  - Keeps tracks of the status of processes
  - Allocates the CPU that is processor to a process
  - De-allocates processor when a process is no more required.

- **Device Management**

  An OS manages device communication via their respective drivers. It performs the following activities for device management.

  - Keeps tracks of all devices connected to system
  - Designates a program responsible for every device known as the Input/output controller.
  - Decides which process gets access to a certain device and for how long.
  - Allocates devices in an effective and efficient way
  - De-allocates devices when they are no longer required.

- **File Management**

  A file system is organized into directories for efficient or easy navigation and usage. These directories may contain other directories and other files. An Operating System carries out the following file management activities.

  It keeps track of where information is stored, user access settings and status of every file and more. These facilities are collectively known as the file system.

3. Define the term semaphore. How does semaphore help in dining philosopher problem?

Ans: A semaphore is a variable or abstract data type used to control access to a common resource by multiple processes in a concurrent system such as a multitasking operating system. A semaphore is simply a variable. This variable is used to solve critical section problems and to achieve process

synchronization in the multi-processing environment. A semaphore S is an integer variable that can be accessed only through two standard operations: wait() and signal().

The wait() operation reduces the value of semaphore by 1 and the signal() operation increases its value by 1.

```
wait(S)
{
while(S<=0);  // busy waiting
S--;
}
signal(S)
{
S++;
}
```

Dining philosopher problem

In 1965, Dijkstra posed and solved a synchronization problem he called the dining philosophers problem. Since that time, everyone inventing yet another synchronization primitive has felt obligated to demonstrate how wonderful the new primitive is by showing how elegantly it solves the dining philosophers' problem.

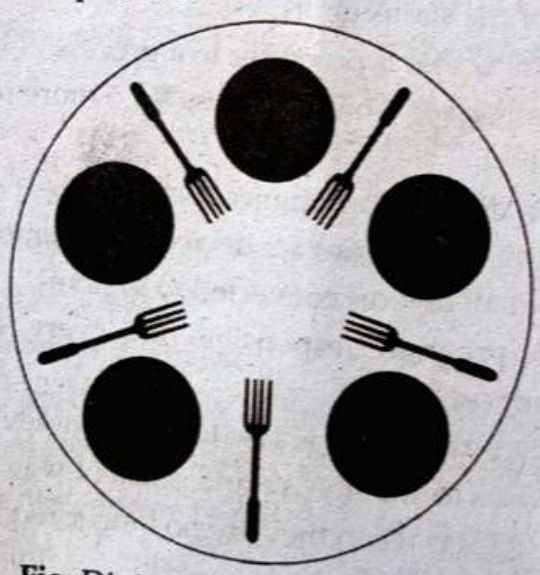


Fig: Dining philosopher problem

There are N philosophers sitting around a circular table eating spaghetti and discussing philosophy. The problem is that each philosopher needs 2 forks to eat, and there are only N forks, one between each 2 philosophers. Design an algorithm that the philosophers can follow that insures that none starves as long as each philosopher eventually stops eating, and such that the maximum numbers of philosopher scan eat at once.

- Philosophers eat/think
- Eating needs 2 forks
- Pick one fork at â time
- How to prevent deadlock

The problem was designed to illustrate the problem of avoiding deadlock, a system state in which no progress is possible. One idea is to instruct each philosopher to behave as follows:

- think until the left fork is available; when it is, pick it up
- think until the right fork is available; when it is, pick it up
- eat
- put the left fork down
- put the right fork down
- repeat from the start

This solution is incorrect: it allows the system to reach deadlock. Suppose that all five philosophers take their left forks simultaneously. None will be able to take their right forks, and there will be a deadlock. We could modify the program so that after taking the left fork, the program checks to see if the right fork is available. If it is not, the philosopher puts down the left one, waits for some time, and then repeats the whole process. This proposal too, fails, although for a different reason. With a little bit of bad luck, all the philosophers could start the algorithm simultaneously, picking up their left forks, seeing that their right forks were not available, putting down their left forks, waiting, and picking up their left forks again simultaneously, and so on, forever. A situation like this, in which all the programs continue to run indefinitely but fail to make any progress is called starvation.

The solution presented below is deadlock-free and allows the maximum parallelism for an arbitrary number of philosophers. It uses an array, state, to keep track of whether a philosopher is eating, thinking, or hungry (trying to acquire forks). A philosopher may move into eating state only if neither neighbor is eating. Philosopher is neighbors are defined by the macros LEFT and RIGHT. In other words, if i is 2, LEFT is 1 and RIGHT is 3.

```
#define N 5                     /* number of philosophers */
#define LEFT (i+N-1)%N          /* number of i's left neighbor */
#define RIGHT (i+1)%N           /* number of i's right neighbor */
#define THINKING 0              /* philosopher is thinking */
#define HUNGRY 1                /* philosopher is trying to get forks */
#define EATING 2                /* philosopher is eating */
Typedefint semaphore;           /* semaphores are a special kind of int */
int state[N];                   /* array to keep track of everyone's state */
semaphoremutex = 1;             /* mutual exclusion for critical regions */
semaphore s[N];                 /* one semaphore per philosopher */
void philosopher (inti)         /* i: philosopher number, from 0 to N-1 */
{
while (TRUE)
{                               /* repeat forever */
think();                        /* philosopher is thinking */
```

```
take_forks(i);                    /* acquire two forks or block */
eat();                            /* yum-yum, spaghetti */
put_forks(i);                     /* put both forks back on table */
}
}
voidtake_forks(inti)              /* i: philosopher number, from 0 to N-1 */
{
down(&mutex);                     /* enter critical region */
state[i] = HUNGRY;                /* record fact that philosopher i is hungry */
test(i);                          /* try to acquire 2 forks */
up(&mutex);                       /* exit critical region */
down(&s[i]);                      /* block if forks were not acquired */
}
voidput_forks(i)                  /* i: philosopher number, from 0 to N-1 */
{
down(&mutex);                     /* enter critical region */
state[i] = THINKING;              /* philosopher has finished eating */
test(LEFT);                       /* see if left neighbor can now eat */
test(RIGHT);                      /* see if right neighbor can now eat */
up(&mutex);                       /* exit critical region */
}
void test(i)                      /* i: philosopher number, from 0 to N-1 */
{
if (state[i] == HUNGRY && state[LEFT] != EATING && state[RIGHT] !=
EATING)
{
state[i] = EATING;
up(&s[i]);
}
}
```

4. A system has two process and three resources. Each process needs a maximum of two resources. Is deadlock possible? Explain with answer.

Ans: Deadlock is not possible. With 2 processes and 3 resources, one process could always get its maximum of 2 resources since the resources are identical. When one process finishes, the resources it held would be released so that the other process could finish its job.

5. Suppose a new process in a system arrives at an average of six processes per minute, and each such process requires an average of 8 seconds of service time. Estimate the fraction of time the CPU is busy in a system with a single processor.

Ans: Given that there are on an average 6 processes per minute.
So the arrival rate = 6 process/min.
i.e. every 10 seconds a new process arrives on an average.
Or we can say that every process stays for 10 seconds with the CPU
Service time = 8 sec.
Hence the fraction of time CPU is busy = service time / staying time
= 8 / 10
=0.8
So the CPU is busy for 80% of the time

6. **Given references to the following pages by a program,**
0, 9, 0, 1, 8, 1, 8, 7, 8, 7, 1, 2, 8, 2, 7, 8, 2, 3, 8, 3
How many page faults will occur if the program has three page frames for each of the following algorithms?
a) FIFO **b)LRU**

Ans:

a) For FIFO page replacement algorithm

Reference string

0	**9**	**0**	**1**	**8**	**1**	**8**	**7**	**8**	**7**	**1**	**2**	**8**	**2**	**7**	**8**	**2**	**3**	**8**	**3**
0	0	0	0	8	8	8	8	8	8	8	8	8	8	8	8	8	3	3	3
	9	9	9	9	9	9	7	7	7	7	7	7	7	7	7	7	7	8	8
			1	1	1	1	1	1	1	1	2	2	2	2	2	2	2	2	2
Pf	Pf	X	Pf	Pf	X	X	Pf	X	X	X	Pf	X	X	X	X	X	Pf	Pf	X

Page fault =09

For LRU page replacement algorithm

0	**9**	**0**	**1**	**8**	**1**	**8**	**7**	**8**	**7**	**1**	**2**	**8**	**2**	**7**	**8**	**2**	**3**	**8**	**3**
0	0	0	0	0	0	0	7	7	7	7	7	8	8	8	8	8	8	8	8
	9	9	9	8	8	8	8	8	8	8	2	2	2	2	2	2	2	2	2
			1	1	1	1	1	1	1	1	1	1	1	7	7	7	3	3	3
Pf	Pf	X	Pf	Pf	X	X	Pf	X	X	X	Pf	Pf	X	Pf	X	X	Pf	X	X

Page fault =09

7. **What is file ? Explain how access control matrix provides resource protection that may access process.**

Ans: A file is a named collection of related information that is recorded on secondary storage such as magnetic disks, magnetic tapes and optical disks. In general, a file is a sequence of bits, bytes, lines or records whose meaning is defined by the files creator and user.

In the Microsoft Windows family of operating systems, users are presented with several different choices of file systems when formatting such media. These choices depend on the type of media involved and the situations in which the media is being formatted. The two most common file systems in Windows are as follows:

- NTFS
- FAT
- exFAT Extended (platform cross) (Mac os)
- HFS Plus Hierarchical File System plus
- EXT Extended File System (linux, unix)

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

represents set of access rights which are given to the processes of domain means each entry(i, j) defines the set of operations that a process executing in domain Di can invoke on object Oj.

	F1	F2	F3	Printer
D1	read		Read	
D2				Print
D3		read	execute	
D4	read write		read write	

According to the above matrix: there are four domains and four objects- three files (F1, F2, F3) and one printer. A process executing in D1 can read files F1 and F3. A process executing in domain D4 has same rights as D1 but it can also write on files. Printer can be accessed by only one process executing in domain D2. The mechanism of access matrix consists of many policies and semantic properties. Specifically, we must ensure that a process executing in domain Di can access only those objects that are specified in row i. Policies of access matrix concerning protection involve which rights should be included in the (i, j)th entry. We must also decide the domain in which each process executes. This policy is usually decided by the operating system. The Users decide the contents of the access-matrix entries.

8. What do you mean by one-time password din authentication ? How worms are differing from virus.

Ans: One-time passwords provide additional security along with normal authentication. In One-Time Password system, a unique password is required every time user tries to login into the system. Once a one-time password is used, then it cannot be used again. One-time passwords are implemented in various ways.

- **Random numbers** – Users are provided cards having numbers printed along with corresponding alphabets. System asks for numbers corresponding to few alphabets randomly chosen.
- **Secret key** – User are provided a hardware device which can create a secret id mapped with user id. System asks for such secret id which is to be generated every time prior to login.
- **Network password** – Some commercial applications send one-time passwords to user on registered mobile/ email which is required to be entered prior to login.

Worms

A computer worm is a malicious, self-replicating software program (popularly termed as malware) which affects the functions of software and hardware programs. Worms don't need a host program in order for them to

run, self-replicate and propagate. Once a worm has made its way onto our system, usually via a network connection or as a downloaded file, it can then make multiple copies of itself and spread via the network or internet connection infecting any inadequately-protected computers and servers on the network. Because each subsequent copy of a network worm can also self-replicate, infections can spread very rapidly via the internet and computer networks.

Viruses

A computer virus is a type of malicious code or program written to alter the way a computer operates and is designed to spread from one computer to another. A virus operates by inserting or attaching itself to a legitimate program or document that supports macros in order to execute its code. In the process, a virus has the potential to cause unexpected or damaging effects, such as harming the system software by corrupting or destroying data.

Difference between Worms and Virus:

S. No.	WORMS	VIRUS
1.	A Worm is a form of malware that replicates itself and can spread to different computers via Network.	A Virus is a malicious executable code attached to another executable file which can be harmless or can modify or delete data.
2.	The main objective of worms to eat the system resources.	The main objective of virus is to modify the information.
3.	It doesn't need a host to replicate from one computer to another.	It require host is needed for spreading.
4.	It is less harmful as compared.	It is more harmful.
5.	Worms can be detected and removed by the Antivirus and firewall.	Antivirus software are used for protection against viruses.
6.	Worms can be controlled by remote.	Virus can't be controlled by remote.
7.	Worms are executed via weaknesses in system.	Viruses are executed via executable files.
8.	Morris Worm, Storm Worm and SQL Slammer are some of the examples of worms.	Resident and Non -resident viruses are two types of Virus.
9.	It does not needs human action to replicate.	It needs human action to replicate.
10.	Its spreading speed is faster.	Its spreading speed is slower as compared.

Group C

Attempt any TWO questions. [2×10 = 20]

9. Write CPU scheduling criteria. For the processes listed in following table, draw Gantt chart illustrating their execution and calculate average waiting time and turnaround time using:

a) First Come First Serve
b) Shortest Remaining TimeNext
c) Priority
d) Round Robin (quantum=1 sec.)

Processes	Arrival Time	Burst Time (sec.)	Priority
A	0.00	7	3
B	2.01	7	1
C	3.01	2	4
D	3.02	2	2

Ans: CPU scheduling is a process which allows one process to use the CPU while the execution of another process is on hold (in waiting state) due to unavailability of any resource like I/O etc., thereby making full use of CPU. The aim of CPU scheduling is to make the system efficient, fast and fair.

Different CPU scheduling algorithms have different properties and the choice of a particular algorithm depends on the various factors. Many criteria have been suggested for comparing CPU scheduling algorithms. The criteria include the following:

a) CPU utilization

The main objective of any CPU scheduling algorithm is to keep the CPU as busy as possible. Theoretically, CPU utilization can range from 0 to 100 but in a real-time system, it varies from 40 to 90 percent depending on the load upon the system.

b) Throughput

A measure of the work done by CPU is the number of processes being executed and completed per unit time. This is called throughput. The throughput may vary depending upon the length or duration of processes.

c) Turnaround time

For a particular process, an important criteria is how long it takes to execute that process. The time elapsed from the time of submission of a process to the time of completion is known as the turnaround time. Turn-around time is the sum of times spent waiting to get into memory, waiting in ready queue, executing in CPU, and waiting for I/O.

d) Waiting time

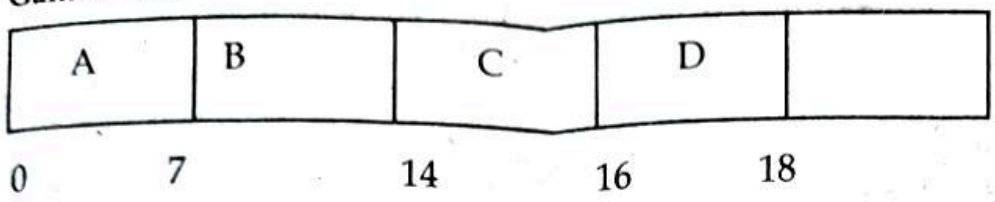
A scheduling algorithm does not affect the time required to complete the process once it starts execution. It only affects the waiting time of a process i.e. time spent by a process waiting in the ready queue.

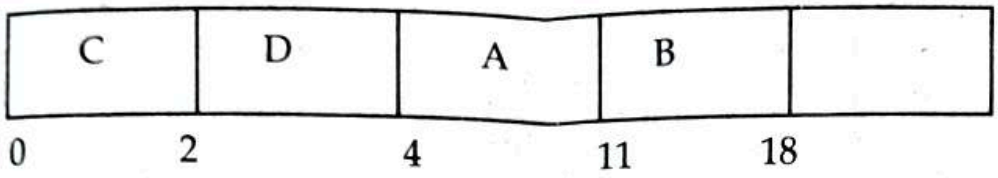
e) Response time

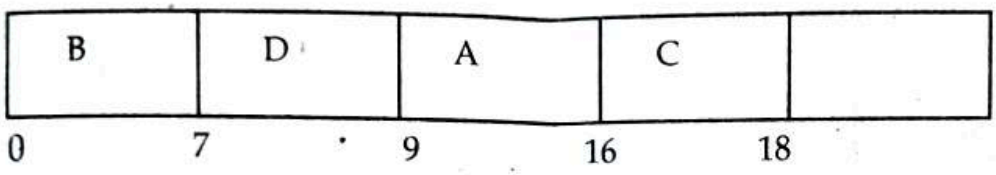
In an interactive system, turn-around time is not the best criteria. A process may produce some output fairly early and continue computing new results while previous results are being output to the user. Thus another criteria is the time taken from submission of the process of request until the first response is produced. This measure is called response time.

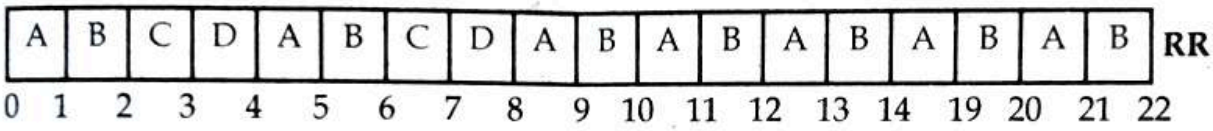
Numerical Part,

Gantt charts









b. **Turnaround time** = Completion time – arrival time

Process	FCFS	RR	SJF	Priority
A	7-0=7	21	11	16
B	14-2.01=11.99	22-2.01=19.99	18-2.01=15.99	7-2.01=4.99
C	16-3.01=12.99	7-3.01=3.99	2-3.01=-1.01	18-3.01=14.99
D	18-3.02=14.98	8-3.02=4.98	4-3.02=0.98	9-3.02=5.98

c. Waiting time (Turnaround time minus burst time)

Process	FCFS	RR	SJF	Priority
A	7-7=0	21-7=14	11-7=4	16-7= 9
B	11.99-7=4.99	19.99-7=12.99	15.99-7=8.99	4.99-7= -2.01
C	12.99-2=10.99	3.99-2=1.99	-1.01-2= -2.01	14.99-2= 12.99
D	14.98-2=12.98	4.98-2=2.98	0.98-2= -1.02	5.98-2= 3.98
Total waiting time	28.96	31.96	9.96	23.96
Average waiting time	7.24	7.99	2.49	5.99

10. Define the term seek time and rotational delay in disk scheduling. Suppose that a disk has 100 cylinders, numbered 0 to 99. The drive is currently serving a request at cylinder 43 and previous request was at cylinder 25. The queue of pending request, in FIFO order is: 86, 70, 13, 74, 48, 9, 22, 50, 30

Starting from the current head position, what is total distance (in cylinders) that the disk arm moves to satisfy all pending request for each of following disk scheduling algorithms?

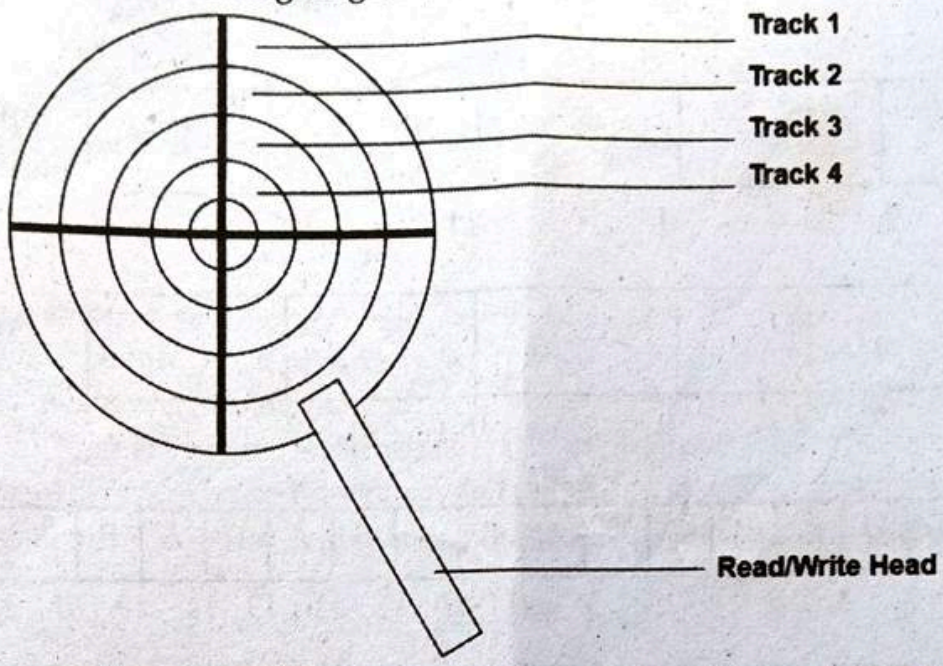
a) FCFS b) SSTF c) SCAN d) LOOK

Ans:

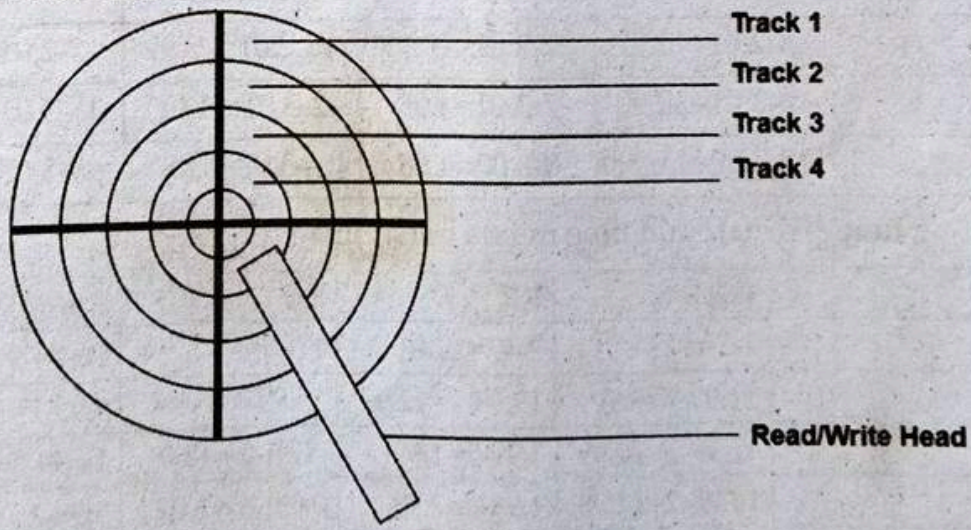
Seek Time: A disk is divided into many circular tracks. Seek Time is defined as the time required by the read/write head to move from one track to another.

Example,

Consider the following diagram, the read/write head is currently on track 1.



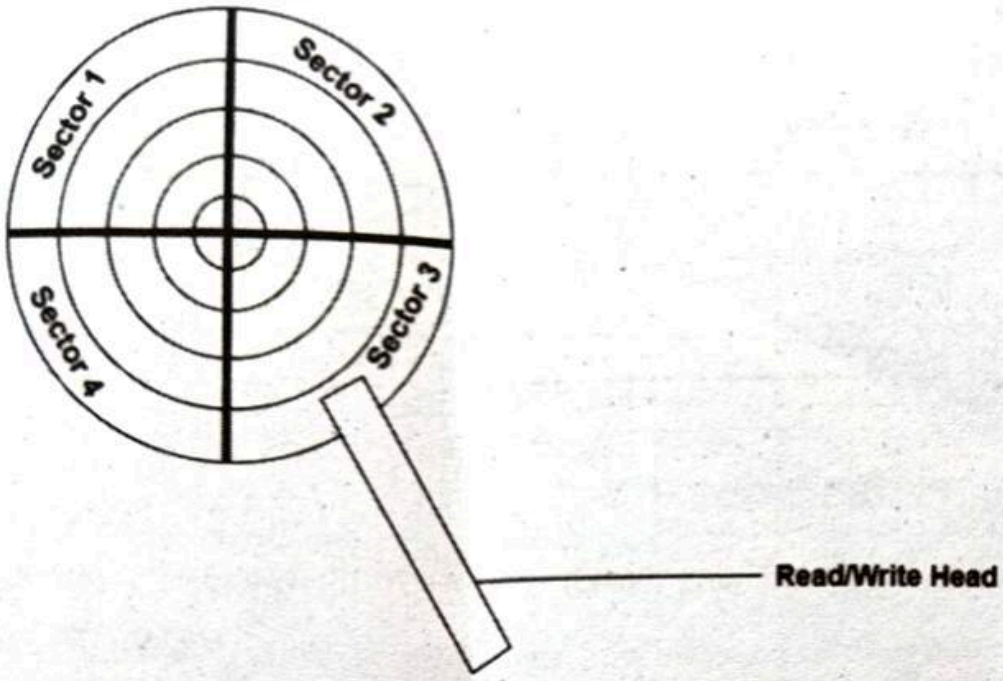
Now, on the next read/write request, we may want to read data from Track 4, in this case, our read/write head will move to track 4. The time it will take to reach track 4 is the seek time.



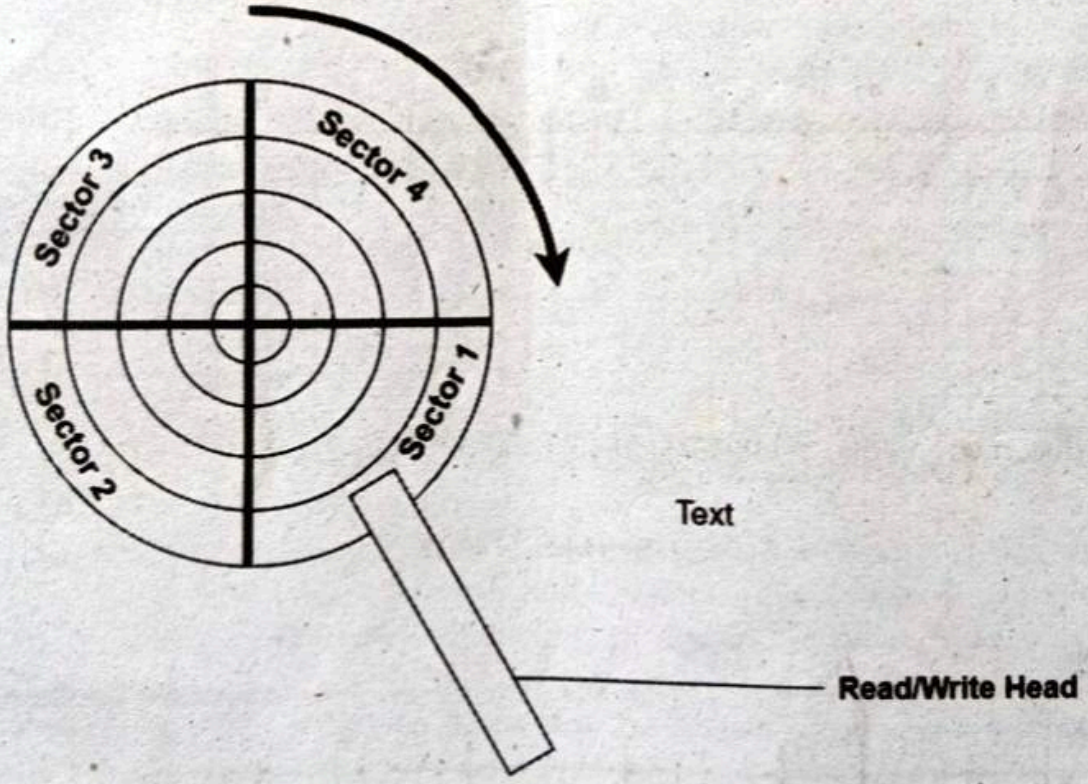
Rotational Latency:

The disk is divided into many circular tracks, and these tracks are further divided into blocks knows as sectors. The time required by the read/write head to rotate to the requested sector from the current position is called Rotational Latency.

Example: Consider the following diagram, we have divided each track into 4 sectors. The systems get a request to read a sector from track 1, thus the read/write head will move to track 1 and this time will be seek time. The read/write head is currently in sector 3.



But the data may not be in sector 3. The data block may be present in sector 1. The time required by read/write head to move from sector 3 to sector 1 is the rotational latency. Below is the final configuration.



Text

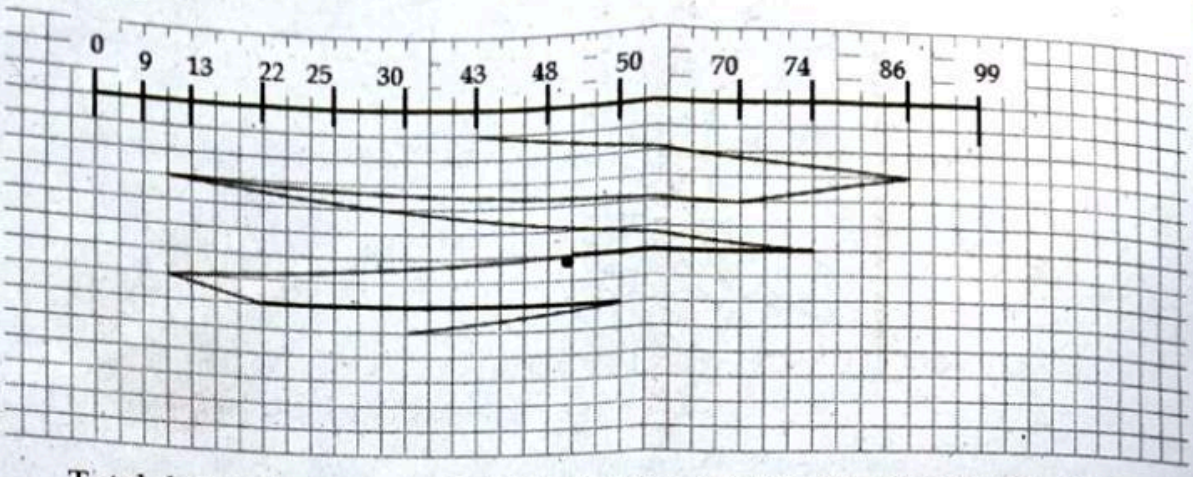
Numerical part,

Suppose that a disk has 100 cylinders, numbered 0 to 99. The drive is currently serving are quest at cylinder 43 and previous request was at cylinder 25. The queue of pending request, in FIFO order is: 86, 70, 13, 74, 48, 9, 22, 50, 30

Starting from the current head position, what is total distance (in cylinders) that the disk arm moves to satisfy all pending request foreach of following disk scheduling algorithms?

a) FCFS b)SSTF c)SCAN d)LOOK

FCFS

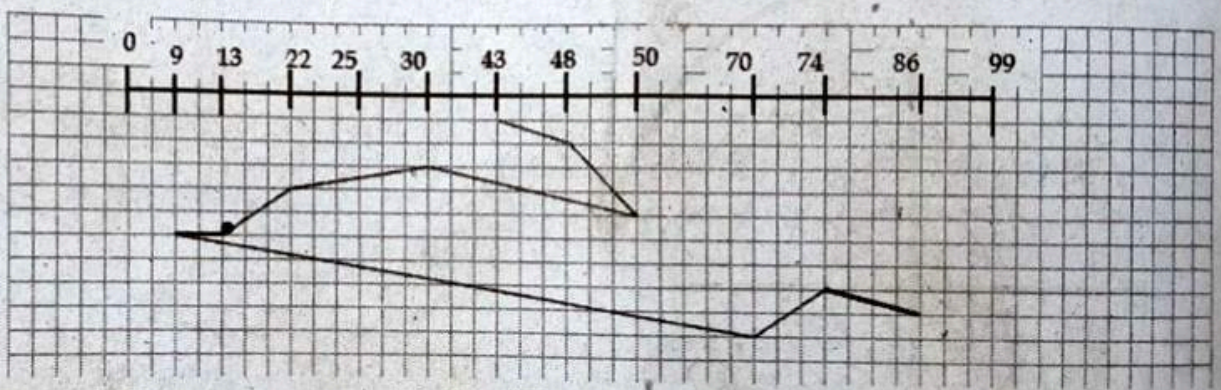


Total disk movements = (86-43)+(86-70)+(70-13)+(74-13)+(74-48)+(84-9)
+(22-9)+(50-22)+(50-30)

= 43+16+57+61+26+75+13+28+20

= 339

SSTF

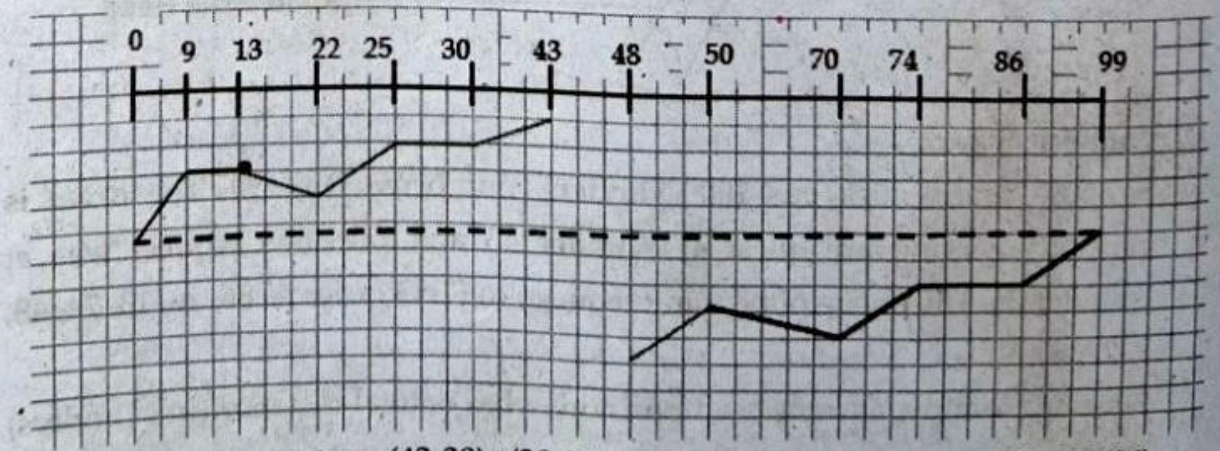


Total disk movements = (48-43)+(50-43)+ (50-30)+ (30-22)+(22-13)+ (13-9)+ (70-9)
+ (74-70)+ (86-43)

= 5+7+20+8+9+4+61+4+43

=161

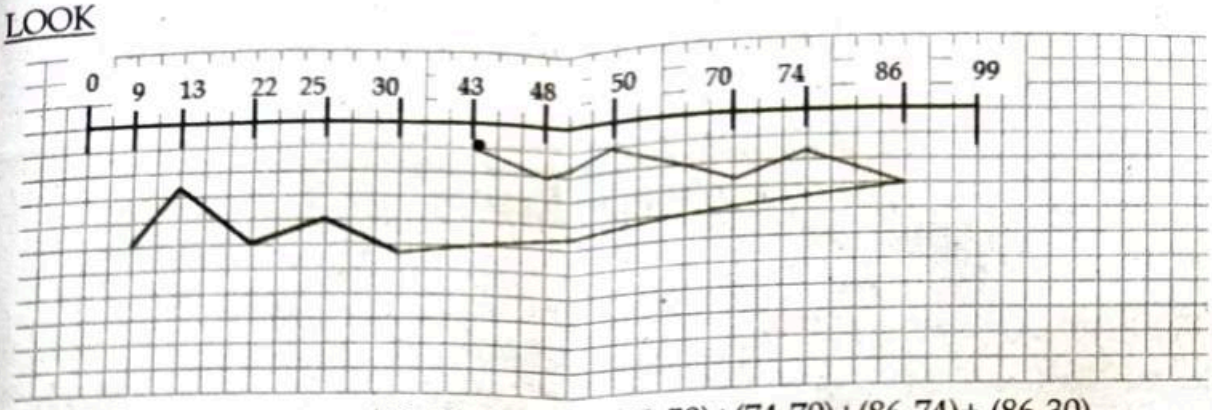
SCAN



Total disk movements = (43-30)+(30-25)+(25-22)+(22-13)+(13-9)+(9-0)+(99-86)
+(86-74)+(74-70)+(70-50)+(50-48)

= 13+5+3+9+4+9+13+12+4+20+2

LOOK



Total disk movements = (48-43)+(50-48)+(70-50)+(74-70)+(86-74)+ (86-30)
+(30-25)+(25-22)+(22-13)+(13-9)

= 5+2+20+4+12+56+5+3+9+4

=120

11. What is clock synchronization? Explain how physical clock synchronize by Berkeley algorithm and logical clock synchronize by lamport's algorithm with suitable example.

Ans: Distributed System is a collection of computers connected via the high speed communication network. In the distributed system, the hardware and software components communicate and coordinate their actions by message passing. Each node in distributed systems can share their resources with other nodes. So, there is need of proper allocation of resources to preserve the state of resources and help coordinate between the several processes. To resolve such conflicts, synchronization is used. Synchronization in distributed systems is achieved via clocks. The clock synchronization can be achieved by 2 ways:

- External and
- Internal Clock Synchronization

External clock synchronization is the one in which an external reference clock is present. It is used as a reference and the nodes in the system can set and adjust their time accordingly.

Internal clock synchronization is the one in which each node shares its time with other nodes and all the nodes set and adjust their times accordingly.

There are 2 types of clock synchronization algorithms: Centralized and Distributed.

- **Centralized** is the one in which a time server is used as a reference. The single time server propagates it's time to the nodes and all the nodes adjust the time accordingly. It is dependent on single time server so if that node fails, the whole system will lose synchronization. Examples of centralized are- Berkeley Algorithm, Passive Time Server, Active Time Server etc.
- **Distributed** is the one in which there is no centralized time server present. Instead the nodes adjust their time by using their local time and then, taking the average of the differences of time with other nodes. Distributed algorithms overcome the issue of centralized algorithms like the scalability and single point failure. Examples of Distributed algorithms are – Global Averaging Algorithm, Localized Averaging Algorithm, NTP (Network time protocol) etc.