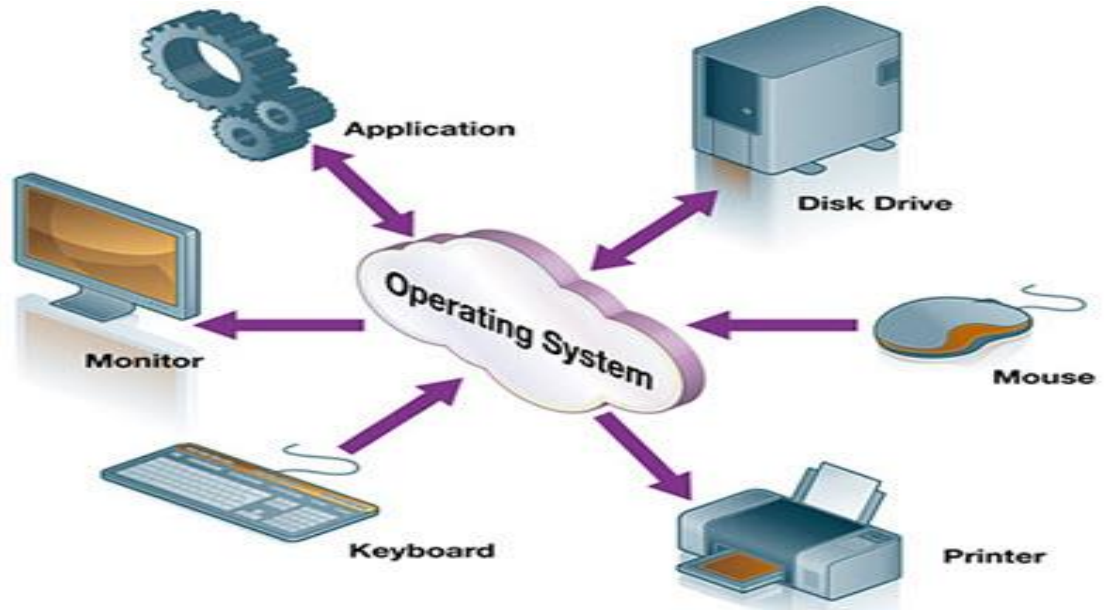
BHADRAK ENGINEERING SCHOOL & TECHNOLOGY
(BEST), ASURALI, BHADRAK

OPERATING SYSTEM

(TH-01)

(As per the 2020-21 syllabus of the SCTE&VT, Bhubaneswar, Odisha)



Fourth Semester

COMPUTER SCIENCE & ENGG.

Prepared By: Er. S.Palit

TH-01- Operating System

Contents

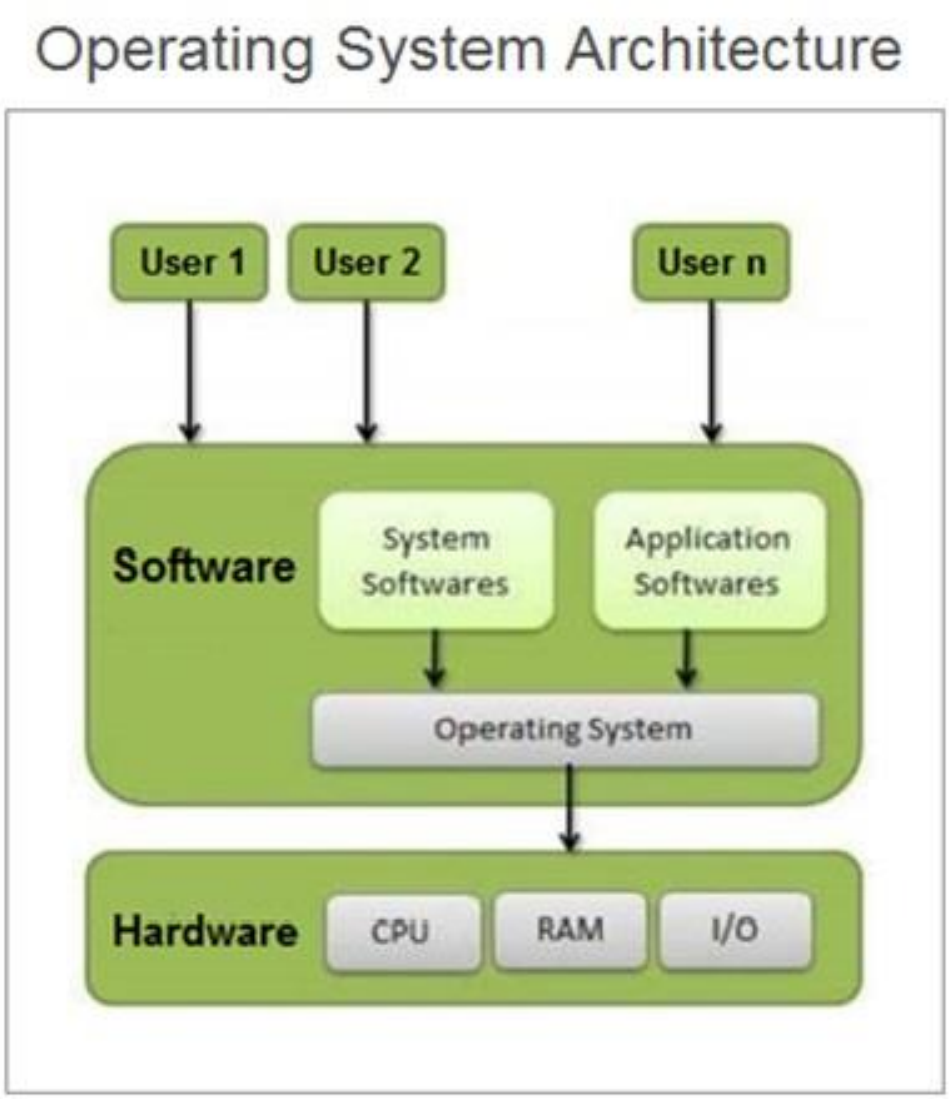
Sl. No.	Topic	Expected Marks
1.	INTRODUCTION	**10**
2.	PROCESS MANAGEMENT	**20**
3.	MEMORY MANAGEMENT	**20**
4.	DEVICE MANAGEMENT	**10**
5.	DEAD LOCKS	**20**
6.	FILE MANAGEMENT	**15**
7.	SYSTEM PROGRAMMING	**15**
TOTAL		**110**

CHAPTER-01.
INTRODUCTION

1.1 Objectives and Explain functions of the operating system.

Operating System

1) OS is a system software.
2) OS acts as an interface between hardware of a computer and user.



Functions of operating system.

1. **Process Management**
2. **memory management**
3. **device management**
4. **file management**

5. **Security**
6. **error detection**

1 . Process management

Operating system manages any kind of activities

A. User programs
B. System programs(printer, spooler, nameserver, file server etc.)

- ➢ Operating system most ,delete, suspend, resume and schedule the processes.
- ➢ According system supports inter process communication, synchronisation, handle deadlock
- ➢ OS allocate the processor to execute a chosen process.

2. Memory management

- ➢ In memory management
  $operating\ systems\ finds\ free\ space\ in\ memory\ and\ allocate\ to\ different\ processes.$
- ➢ OS decides when to load each process into memory and also decides when a process should remove from memory.
- ➢ OS keep track of memory in huge on used memory protect memory space allocate and locate spaces for process and swap processes.

3. **Device management**

- ➢ In device management function locate the device or resource to a process.

4. File management

In file management function os keeps track and information about files that how they are open and closed.

5. Security

OS provides security facilities to the system.

.

6. **Error detection**

OS also provides error detection facilities.

1.2 Evolution of Operating system

1. Batch processing operating system

- ➢ In this OS user does not have the direct Access to the machines rather users submit the job on the punch card to a computer operator fruit that is the jobs together sequentially and place the entire batch on an input device for the use of a monitor.

- ➢ In this type of OS the CPU is often idle because the speed of I/O devices is slower.

Multiprogramming OS

- Single program cannot keep the CPU or IO device all time busy.
- In multiprogramming OS the job are organised in such a way that CPU has always one job to execute.
- it increases the utilisation of CPU.
- In multiprogramming OS more than one task aur program can be executed buy one CPU at the same time.

Multitasking or time sharing OS

- ➢ It is a logical extension of of multiprogramming OS.
- ➢ in multiprogramming OS we can only e efficiently use the CPU but there is no facility for user interaction with computer.
- ➢ In time sharing aur multitasking OS the CPU is utilised effectively and there are more than one user can interact with the system at the same time.
- ➢ The switching of CPU between two users is so fast that it gives the impression to the user that he is only working on the system but actually it is shared among multiple users.

Network operating system

- ➢ In this type of OS softwares we can communicate with other computers through a network.
- ➢ This OS allows resources(file, application program, printer) to be shared among computers.
- ➢ Example- Windows NT ,Unix ,MS LAN manager etc.

Distributed OS

- ➢ It has more than one CPU.
- ➢ It controls and manages the hardware and software resources of a distributed system.
- ➢ When a program is executed in a distributed system user is not aware of where the program is executed or or the location of resources is accessed.

Multiprocessor OS

- ➢ It is a set of processors that share a set of physical memory blocks over an interconnection network.

Real time operating system

- ➢ It provides weak response time and it needs a scheduling deadline.
- ➢ real time system has many events that Must be expanded and processed in a short time.
- ➢ It is two types
    1. Soft real time system
    2. hard real time system

Soft real time system

- If certain deadlines are missed then the system continues working with no failure.
- but it's performance degrade
- it is less restrictive.
- Example - telephone switching equipment

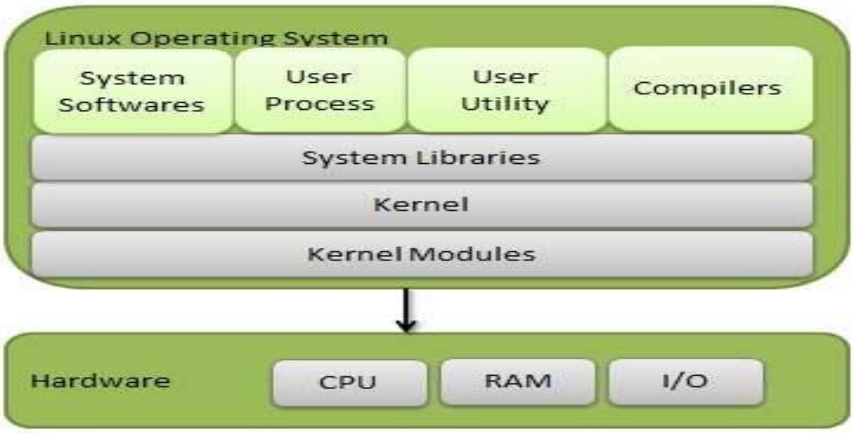
Hard real time system

- If any deadlock is missed then the system will fail to work.
- it is more restrictive.
- Example- Missile launching , flight control.

1.3 Structure of operating system.

Kernel

- It is the protective part of the OS .
- It is the control module of OS.
- It is an essential part of the OS which is loaded first at the time of booting and remains in the memory until the computer system is shut down.



Short Questions with answer

1.What is OS?

- ➢ OS is a system software.
- ➢ OS acts as an interface between hardware of a computer and user.

2. Define multiprogramming OS .

- In multiprogramming OS the job are organised in such a way that CPU has always one job to execute.
- it increases the utilisation of CPU.
- In multiprogramming OS more than one task aur program can be executed buy one CPU at the same time.

3.Define kernel.

- It is the protective part of the OS .
- It is the control module of OS.
- It is an essential part of the OS which is loaded first at the time of booting and remains in the memory until the computer system is shut down.

Long Questions

1) **Define OS. Explain various functions of OS.**
2) **Differentiate multiprogramming and multiprocessing.**
3) **Distinguish between network operating system and distributed OS.**

CHAPTER-02.
PROCESS MANAGEMENT

2.1 Process concept

- A program during execution is known as a process.
- The execution of a process must progress in a sequential fashion.

process control

- A process control block (PCB) is a data structure used by computer operating systems to store all the information about a process.
- Information in a process control block is updated during the transition of process states.

Interacting processes

- Processes interact by making system calls into the operating system proper (i.e. the kernel).
- Though we will see that, for stability, such calls are not direct calls to kernel functions.
- Programming interface to the services provided by the OS (e.g. open file, read file, etc.)

Inter process messages.

- **Inter**-**process** communication (IPC) is a mechanism that allows **processes** to communicate with each other and synchronize their actions.
- The communication between these **processes** can be seen as a **method** of co-operation between them.

2.2 Implementation issues of Processes.

2.3 Process scheduling

The process scheduling is the activity of the process manager that handles the removal of the running process from the CPU and the selection of another process on the basis of a particular strategy.

FCFS Scheduling-

In FCFS Scheduling,

- **The process which arrives first in the ready queue is firstly assigned the CPU.**
- **In case of a tie, process with smaller process id is executed first.**
- **It is always non-preemptive in nature.**

Advantages-

- **It is simple and easy to understand.**
- **It can be easily implemented using queue data structure.**
- **It does not lead to starvation.**

Disadvantages-

- **It does not consider the priority or burst time of the processes.**
- **It suffers from convoy effect.**

Convoy Effect

In convoy effect,

- **Consider processes with higher burst time arrived before the processes with smaller burst time.**
- **Then, smaller processes have to wait for a long time for longer processes to release the CPU.**

PRACTICE PROBLEMS BASED ON FCFS SCHEDULING-

Problem-01:

Consider the set of 5 processes whose arrival time and burst time are given below-

Process Id	Arrival time	Burst time
P1	3	4
P2	5	3
P3	0	2
P4	5	1
P5	4	3

If the CPU scheduling policy is FCFS, calculate the average waiting time and average turn around time.

Solution-

Gantt Chart-

Gantt Chart

Here, black box represents the idle time of CPU.

Now, we know-

- **Turn Around time = Exit time – Arrival time**
- **Waiting time = Turn Around time – Burst time**

Also read-Various Times of Process

Process Id	Exit time	Turn Around time	Waiting time
P1	7	7 – 3 = 4	4 – 4 = 0
P2	13	13 – 5 = 8	8 – 3 = 5
P3	2	2 – 0 = 2	2 – 2 = 0
P4	14	14 – 5 = 9	9 – 1 = 8
P5	10	10 – 4 = 6	6 – 3 = 3

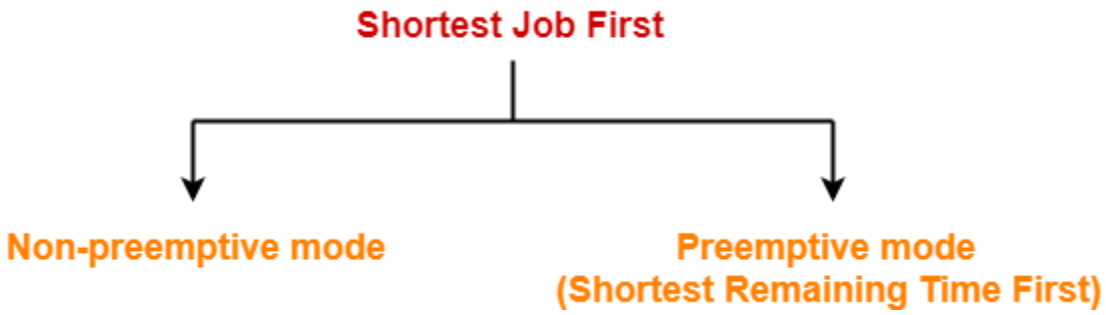
Now,

- **Average Turn Around time = (4 + 8 + 2 + 9 + 6) / 5 = 29 / 5 = 5.8 unit**
- **Average waiting time = (0 + 5 + 0 + 8 + 3) / 5 = 16 / 5 = 3.2 unit**

SJF Scheduling-

In SJF Scheduling,

- **Out of all the available processes, CPU is assigned to the process having smallest burst time.**
- **In case of a tie, it is broken by FCFS Scheduling.**



- **SJF Scheduling can be used in both preemptive and non-preemptive mode.**
- **Preemptive mode of Shortest Job First is called as Shortest Remaining Time First (SRTF).**

Advantages-

- **SRTF is optimal and guarantees the minimum average waiting time.**
- **It provides a standard for other algorithms since no other algorithm performs better than it.**

Disadvantages-

- **It can not be implemented practically since burst time of the processes can not be known in advance.**
- **It leads to starvation for processes with larger burst time.**
- **Priorities can not be set for the processes.**
- **Processes with larger burst time have poor response time.**

PRACTICE PROBLEMS BASED ON SJF SCHEDULING-

Problem-01:

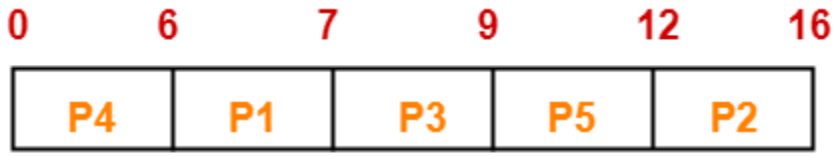
Consider the set of 5 processes whose arrival time and burst time are given below-

Process Id	Arrival time	Burst time
P1	3	1
P2	1	4
P3	4	2
P4	0	6
P5	2	3

If the CPU scheduling policy is SJF non-preemptive, calculate the average waiting time and average turn around time.

Solution-

Gantt Chart-



Gantt Chart

Now, we know-

- **Turn Around time = Exit time – Arrival time**
- **Waiting time = Turn Around time – Burst time**

Also read-Various Times of Process

Process Id	Exit time	Turn Around time	Waiting time
P1	7	7 – 3 = 4	4 – 1 = 3
P2	16	16 – 1 = 15	15 – 4 = 11
P3	9	9 – 4 = 5	5 – 2 = 3
P4	6	6 – 0 = 6	6 – 6 = 0
P5	12	12 – 2 = 10	10 – 3 = 7

Now,

- **Average Turn Around time = (4 + 15 + 5 + 6 + 10) / 5 = 40 / 5 = 8 unit**
- **Average waiting time = (3 + 11 + 3 + 0 + 7) / 5 = 24 / 5 = 4.8 unit**

Problem-02:

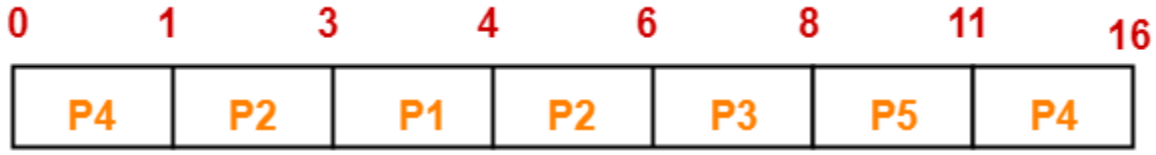
Consider the set of 5 processes whose arrival time and burst time are given below-

Process Id	Arrival time	Burst time
P1	3	1
P2	1	4
P3	4	2
P4	0	6
P5	2	3

If the CPU scheduling policy is SJF preemptive, calculate the average waiting time and average turnaround time.

Solution-

Gantt Chart-



Gantt Chart

Now, we know-

- **Turn Around time = Exit time – Arrival time**
- **Waiting time = Turn Around time – Burst time**

Process Id	Exit time	Turn Around time	Waiting time
P1	4	4 – 3 = 1	1 – 1 = 0
P2	6	6 – 1 = 5	5 – 4 = 1
P3	8	8 – 4 = 4	4 – 2 = 2
P4	16	16 – 0 = 16	16 – 6 = 10
P5	11	11 – 2 = 9	9 – 3 = 6

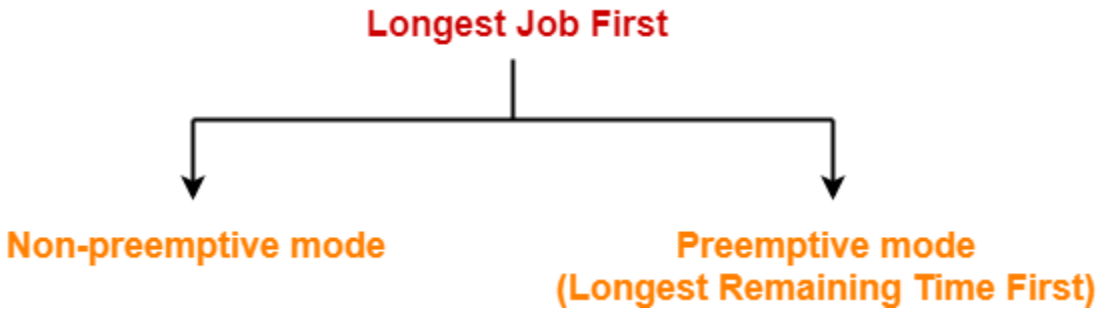
Now,

- **Average Turn Around time = (1 + 5 + 4 + 16 + 9) / 5 = 35 / 5 = 7 unit**
- **Average waiting time = (0 + 1 + 2 + 10 + 6) / 5 = 19 / 5 = 3.8 unit**

Longest Job First Algorithm-

In LJF Scheduling,

- **Out of all the available processes, CPU is assigned to the process having largest burst time.**
- **In case of a tie, it is broken by FCFS Scheduling.**



- **LJF Scheduling can be used in both preemptive and non-preemptive mode.**
- **Preemptive mode of Longest Job First is called as Longest Remaining Time First (LRTF).**

Advantages-

- **No process can complete until the longest job also reaches its completion.**
- **All the processes approximately finishes at the same time.**

Disadvantages-

- **The waiting time is high.**
- **Processes with smaller burst time may starve for CPU.**

PRACTICE PROBLEMS BASED ON LJF SCHEDULING-

Problem-01:

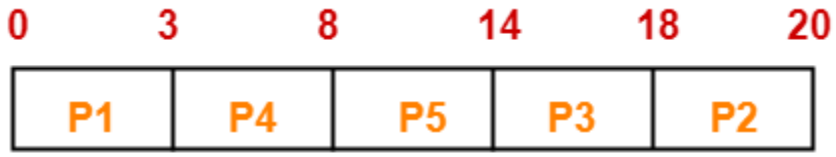
Consider the set of 5 processes whose arrival time and burst time are given below-

Process Id	Arrival time	Burst time
P1	0	3
P2	1	2
P3	2	4
P4	3	5
P5	4	6

If the CPU scheduling policy is LJF non-preemptive, calculate the average waiting time and average turn around time.

Solution-

Gantt Chart-



Gantt Chart

Now, we know-

- **Turn Around time = Exit time – Arrival time**
- **Waiting time = Turn Around time – Burst time**

Also read-Various Times of Process

Process Id	Exit time	Turn Around time	Waiting time
P1	3	3 – 0 = 3	3 – 3 = 0
P2	20	20 – 1 = 19	19 – 2 = 17
P3	18	18 – 2 = 16	16 – 4 = 12
P4	8	8 – 3 = 5	5 – 5 = 0
P5	14	14 – 4 = 10	10 – 6 = 4

Now,

- **Average Turn Around time = (3 + 19 + 16 + 5 + 10) / 5 = 53 / 5 = 10.6 unit**
- **Average waiting time = (0 + 17 + 12 + 0 + 4) / 5 = 33 / 5 = 6.6 unit**

Problem-02:

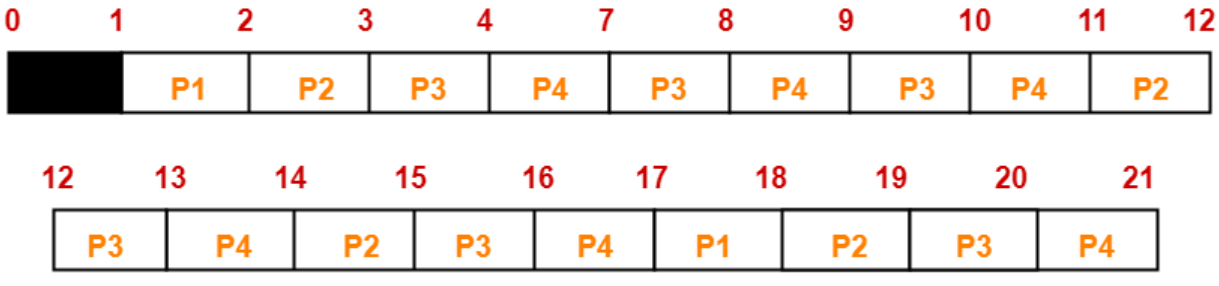
Consider the set of 4 processes whose arrival time and burst time are given below-

Process Id	Arrival time	Burst time
P1	1	2
P2	2	4
P3	3	6
P4	4	8

If the CPU scheduling policy is LJF preemptive, calculate the average waiting time and average turn around time.

Solution-

Gantt Chart-



Gantt Chart

Now, we know-

- **Turn Around time = Exit time – Arrival time**
- **Waiting time = Turn Around time – Burst time**

Process Id	Exit time	Turn Around time	Waiting time
P1	18	18 – 1 = 17	17 – 2 = 15
P2	19	19 – 2 = 17	17 – 4 = 13
P3	20	20 – 3 = 17	17 – 6 = 11
P4	21	21 – 4 = 17	17 – 8 = 9

Now,

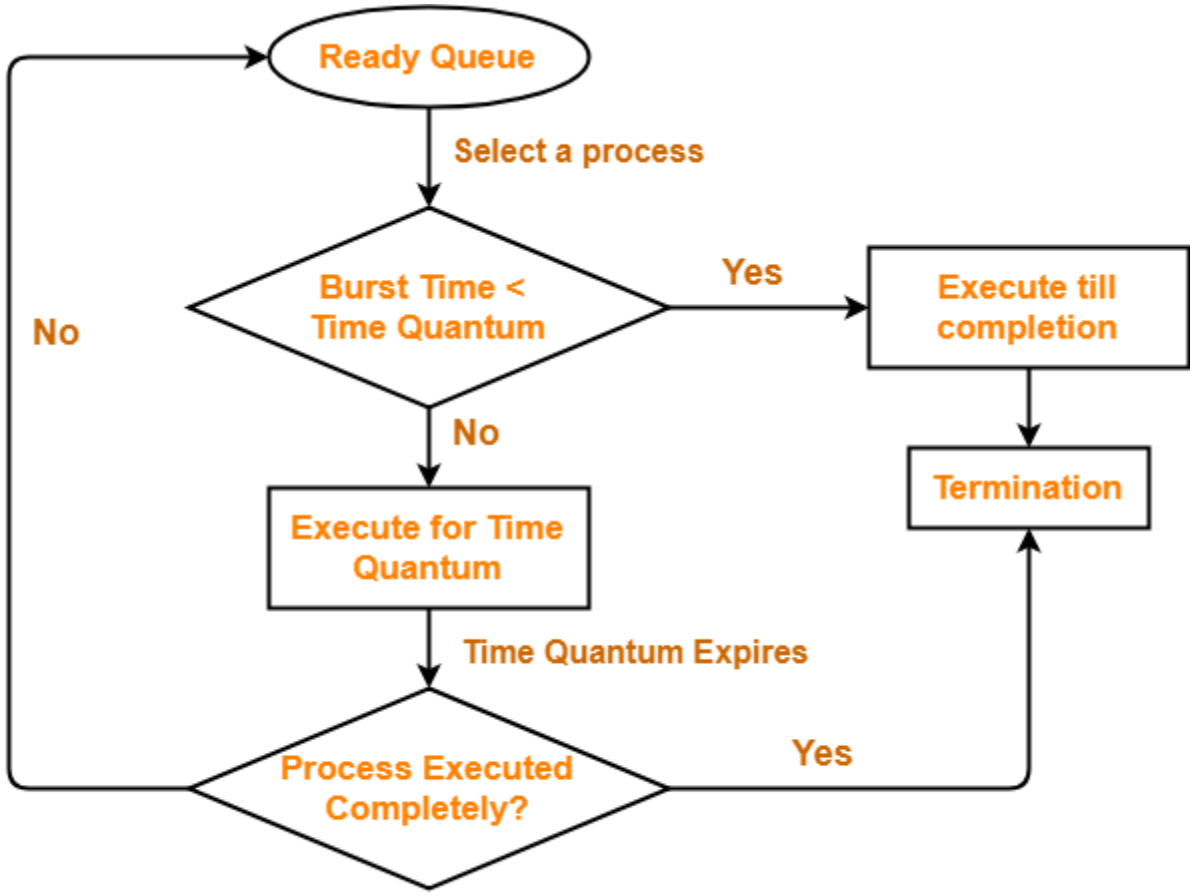
- **Average Turn Around time = (17 + 17 + 17 + 17) / 4 = 68 / 4 = 17 unit**
- **Average waiting time = (15 + 13 + 11 + 9) / 4 = 48 / 4 = 12 unit**

Round Robin Scheduling-

In Round Robin Scheduling,

- **CPU is assigned to the process on the basis of FCFS for a fixed amount of time.**
- **This fixed amount of time is called as time quantum or time slice.**
- **After the time quantum expires, the running process is preempted and sent to the ready queue.**
- **Then, the processor is assigned to the next arrived process.**
- **It is always preemptive in nature.**

Round Robin Scheduling is FCFS Scheduling with preemptive mode.



Round Robin Scheduling

Advantages-

- **It gives the best performance in terms of average response time.**
- **It is best suited for time sharing system, client server architecture and interactive system.**

Disadvantages-

- **It leads to starvation for processes with larger burst time as they have to repeat the cycle many times.**
- **Its performance heavily depends on time quantum.**
- **Priorities can not be set for the processes.**

Important Notes-

Note-01:

With decreasing value of time quantum,

- **Number of context switch increases**
- **Response time decreases**
- **Chances of starvation decreases**

Thus, smaller value of time quantum is better in terms of response time.

Note-02:

With increasing value of time quantum,

- **Number of context switch decreases**
- **Response time increases**
- **Chances of starvation increases**

Thus, higher value of time quantum is better in terms of number of context switch.

Note-03:

- **With increasing value of time quantum, Round Robin Scheduling tends to become FCFS Scheduling.**
- **When time quantum tends to infinity, Round Robin Scheduling becomes FCFS Scheduling.**

Also read-FCFS Scheduling

Note-04:

- **The performance of Round Robin scheduling heavily depends on the value of time quantum.**
- **The value of time quantum should be such that it is neither too big nor too small.**

PRACTICE PROBLEMS BASED ON ROUND ROBIN SCHEDULING-

Problem-01:

Consider the set of 5 processes whose arrival time and burst time are given below-

Process Id	Arrival time	Burst time
P1	0	5
P2	1	3

P3	2	1
P4	3	2
P5	4	3

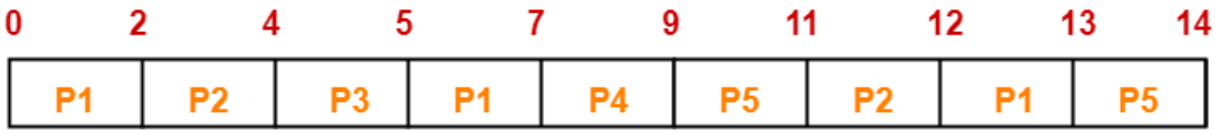
If the CPU scheduling policy is Round Robin with time quantum = 2 unit, calculate the average waiting time and average turn around time.

Solution-

Gantt Chart-

Ready Queue-

P5, P1, P2, P5, P4, P1, P3, P2, P1



Gantt Chart

Now, we know-

- **Turn Around time = Exit time – Arrival time**
- **Waiting time = Turn Around time – Burst time**

Also read-Various Times of Process

Process Id	Exit time	Turn Around time	Waiting time
P1	13	13 – 0 = 13	13 – 5 = 8
P2	12	12 – 1 = 11	11 – 3 = 8
P3	5	5 – 2 = 3	3 – 1 = 2
P4	9	9 – 3 = 6	6 – 2 = 4
P5	14	14 – 4 = 10	10 – 3 = 7

Now,

- **Average Turn Around time = (13 + 11 + 3 + 6 + 10) / 5 = 43 / 5 = 8.6 unit**
- **Average waiting time = (8 + 8 + 2 + 4 + 7) / 5 = 29 / 5 = 5.8 unit**

Problem-02:

Consider the set of 6 processes whose arrival time and burst time are given below-

Process Id	Arrival time	Burst time
P1	0	4
P2	1	5
P3	2	2

P4	3	1
P5	4	6
P6	6	3

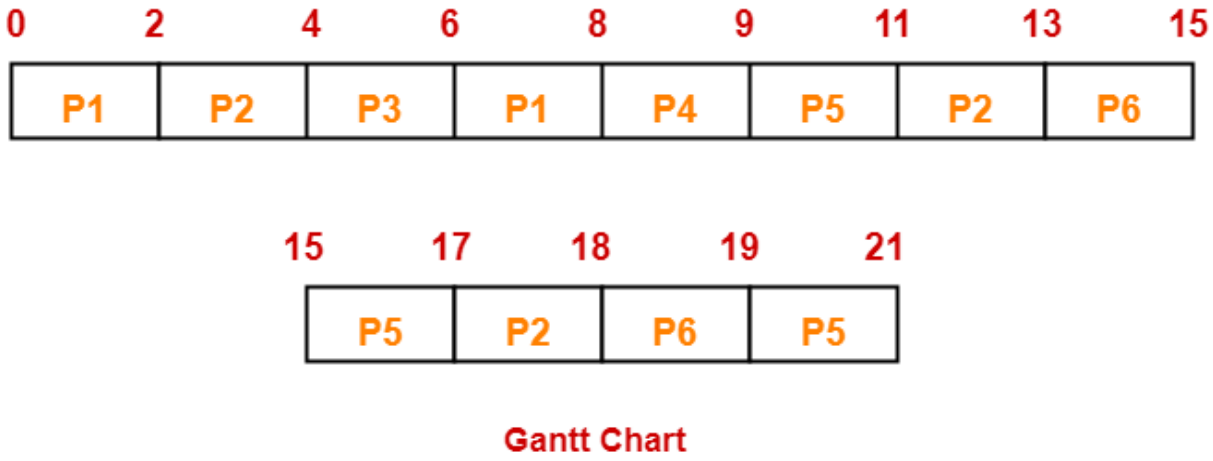
If the CPU scheduling policy is Round Robin with time quantum = 2, calculate the average waiting time and average turn around time.

Solution-

Gantt chart-

Ready Queue-

P5, P6, P2, P5, P6, P2, P5, P4, P1, P3, P2, P1



Gantt Chart

Now, we know-

- **Turn Around time = Exit time – Arrival time**

- **Waiting time = Turn Around time – Burst time**

Process Id	Exit time	Turn Around time	Waiting time
P1	8	8 – 0 = 8	8 – 4 = 4
P2	18	18 – 1 = 17	17 – 5 = 12
P3	6	6 – 2 = 4	4 – 2 = 2
P4	9	9 – 3 = 6	6 – 1 = 5
P5	21	21 – 4 = 17	17 – 6 = 11
P6	19	19 – 6 = 13	13 – 3 = 10

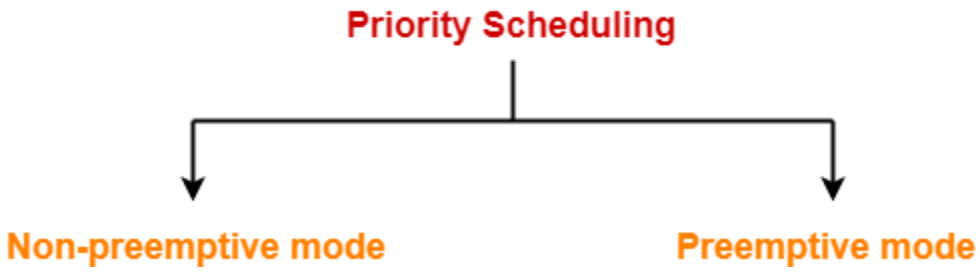
Now,

- **Average Turn Around time = (8 + 17 + 4 + 6 + 17 + 13) / 6 = 65 / 6 = 10.84 unit**
- **Average waiting time = (4 + 12 + 2 + 5 + 11 + 10) / 6 = 44 / 6 = 7.33 unit**

Priority Scheduling-

In Priority Scheduling,

- **Out of all the available processes, CPU is assigned to the process having the highest priority.**
- **In case of a tie, it is broken by FCFS Scheduling.**



- **Priority Scheduling can be used in both preemptive and non-preemptive mode.**

Advantages-

- **It considers the priority of the processes and allows the important processes to run first.**
- **Priority scheduling in preemptive mode is best suited for real time operating system.**

Disadvantages-

- **Processes with lesser priority may starve for CPU.**
- **There is no idea of response time and waiting time.**

Important Notes-

Note-01:

- **The waiting time for the process having the highest priority will always be zero in preemptive mode.**

- **The waiting time for the process having the highest priority may not be zero in non-preemptive mode.**

Note-02:

Priority scheduling in preemptive and non-preemptive mode behaves exactly same under following conditions-

- **The arrival time of all the processes is same**
- **All the processes become available**

PRACTICE PROBLEMS BASED ON PRIORITY SCHEDULING-

Problem-01:

Consider the set of 5 processes whose arrival time and burst time are given below-

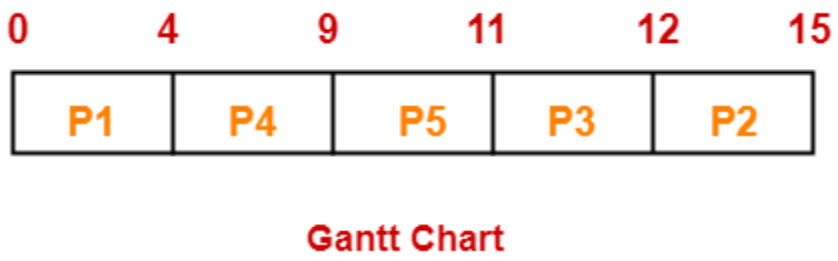
Process Id	Arrival time	Burst time	Priority
P1	0	4	2
P2	1	3	3
P3	2	1	4
P4	3	5	5

P5	4	2	5

If the CPU scheduling policy is priority non-preemptive, calculate the average waiting time and average turn around time. (Higher number represents higher priority)

Solution-

Gantt Chart-



Gantt Chart

Now, we know-

- **Turn Around time = Exit time – Arrival time**
- **Waiting time = Turn Around time – Burst time**

Also read-Various Times of Process

Process Id	Exit time	Turn Around time	Waiting time
P1	4	4 – 0 = 4	4 – 4 = 0
P2	15	15 – 1 = 14	14 – 3 = 11

P3	**12**	**12 – 2 = 10**	**10 – 1 = 9**
P4	**9**	**9 – 3 = 6**	**6 – 5 = 1**
P5	**11**	**11 – 4 = 7**	**7 – 2 = 5**

Now,

- **Average Turnaround time = (4 + 14 + 10 + 6 + 7) / 5 = 41 / 5 = 8.2 unit**
- **Average waiting time = (0 + 11 + 9 + 1 + 5) / 5 = 26 / 5 = 5.2 unit**

Job scheduling.

- ➢ Job scheduling is the process where different tasks get executed at a predetermined time or when the right event happens.
- ➢ A job scheduler is a system that can be integrated with other software systems for the purpose of executing or notifying other software components when a predetermined, scheduled time arrives.

2.4 Process synchronization

Process Synchronization-

When multiple processes execute concurrently sharing system resources, then inconsistent results might be produced.

- **Process Synchronization is a mechanism that deals with the synchronization of processes.**
- **It controls the execution of processes running concurrently to ensure that consistent results are produced.**

Need of Synchronization-

Process synchronization is needed-

- **When multiple processes execute concurrently sharing some system resources.**
- **To avoid the inconsistent results.**

Critical Section-

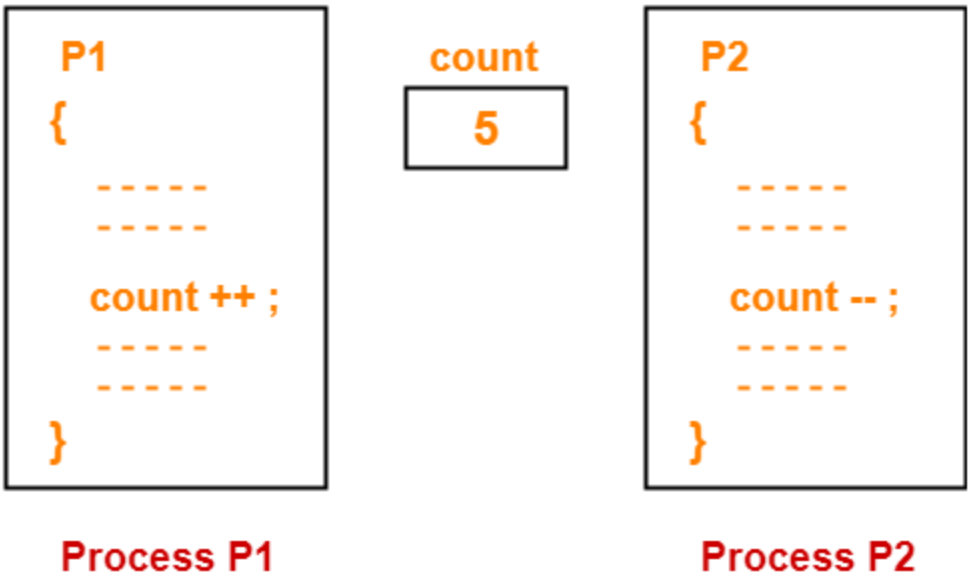
Critical section is a section of the program where a process access the shared resources during its execution.

Example-

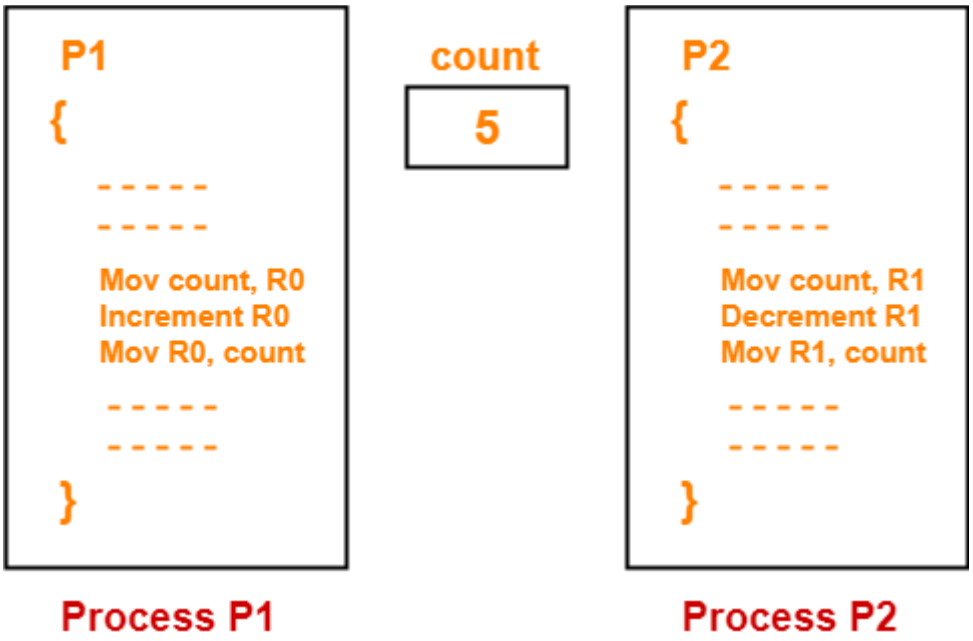
The following illustration shows how inconsistent results may be produced if multiple processes execute concurrently without any synchronization.

Consider-

- **Two processes P_1 and P_2 are executing concurrently.**
- **Both the processes share a common variable named “count” having initial value = 5.**
- **Process P1 tries to increment the value of count.**
- **Process P2 tries to decrement the value of count.**



In assembly language, the instructions of the processes may be written as-



Now, when these processes execute concurrently without synchronization, different results may be produced.

Case-01:

The execution order of the instructions may be-

$$P_1(1), P_1(2), P_1(3), P_2(1), P_2(2), P_2(3)$$

In this case,

Final value of count = 5

Case-02:

The execution order of the instructions may be-

$$P_2(1), P_2(2), P_2(3), P_1(1), P_1(2), P_1(3)$$

In this case,

Final value of count = 5

Case-03:

The execution order of the instructions may be-

$$P_1(1), P_2(1), P_2(2), P_2(3), P_1(2), P_1(3)$$

In this case,

Final value of count = 6

Case-04:

The execution order of the instructions may be-

$$P_2(1), P_1(1), P_1(2), P_1(3), P_2(2), P_2(3)$$

In this case,

Final value of count = 4

Case-05:

The execution order of the instructions may be-

$$P_1(1), P_1(2), P_2(1), P_2(2), P_1(3), P_2(3)$$

In this case,

Final value of count = 4

It is clear from here that inconsistent results may be produced if multiple processes execute concurrently without any synchronization.

Race Condition-

Race condition is a situation where-

- **The final output produced depends on the execution order of instructions of different processes.**
- **Several processes compete with each other.**

Semaphore

Semaphore is simply a variable that is non-negative and shared between threads. This variable is used to solve the critical section problem and to achieve process synchronization in the multiprocessing environment.

Semaphores are of two types:

1. **Binary Semaphore –**

   This is also known as mutex lock. It can have only two values – 0 and 1. Its value is initialized to 1. It is used to implement the solution of critical section problems with multiple processes.

2. **Counting Semaphore –**

   Its value can range over an unrestricted domain. It is used to control access to a resource that has multiple instances.

2.5 Principle of concurrency

Concurrency is the execution of the multiple instruction sequences at the same time. It happens in the operating system when there are several process threads running in parallel. The running process threads always communicate with each other through shared memory or message passing.

Types of scheduling.

Various types of scheduling are

- First-Come, First-Served (FCFS) Scheduling
- Shortest-Job-first (SJF) Scheduling
- Priority Scheduling
- Shortest Remaining Time first
- Round Robin(RR) Scheduling

Short Questions with answer

1. Define process.

➢ A program during execution is known as a process.

➢ The execution of a process must progress in a sequential fashion.

2. Define semaphore.

➢ Semaphore is simply a variable that is non-negative and shared between threads.

➢ This variable is used to solve the critical section problem and to achieve process synchronization in the multiprocessing environment.

3 . What is race condition?

Race condition is a situation where-

- The final output produced depends on the execution order of instructions of different processes.
- Several processes compete with each other.

Long Questions

1) Define FCFS scheduling with example.
2) What is process scheduling? Explain Round robin algorithm with example.
3) Explain priority scheduling with an example.

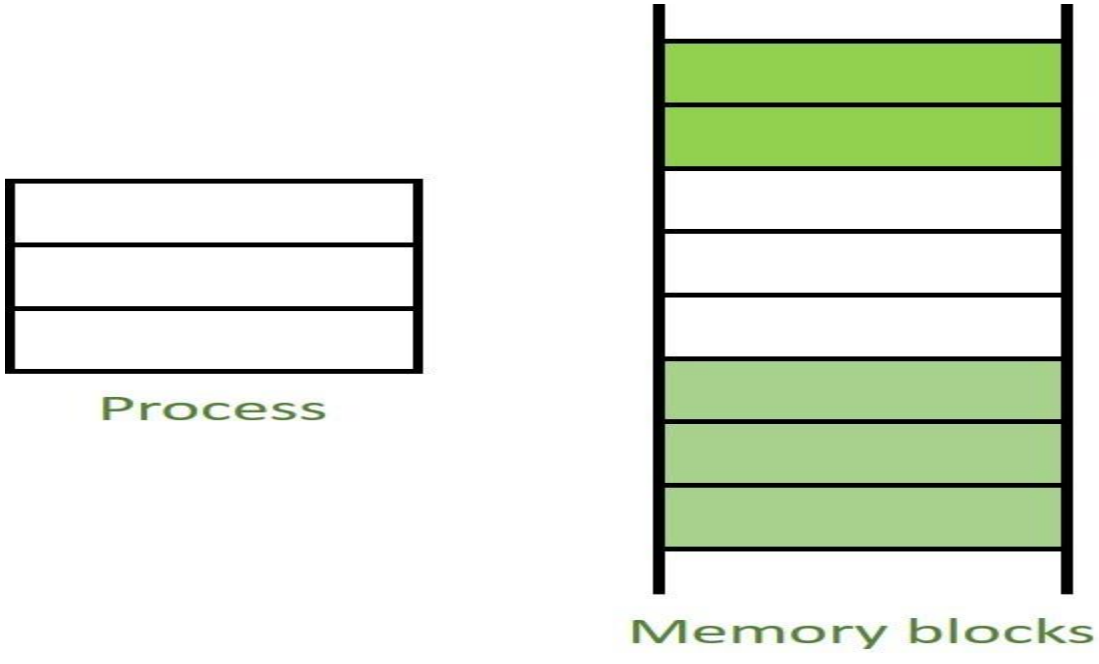
CHAPTER-03.
MEMORY MANAGEMENT

3.1 Memory allocation Techniques

- The two fundamental **methods** of **memory allocation** are static and dynamic **memory allocation**.
- Static **memory allocation method** assigns the **memory** to a process, before its execution.
- On the other hand, the dynamic **memory allocation method** assigns the **memory** to a process, during its execution.

Contiguous memory allocation

- **Contiguous memory allocation** is a **memory allocation** method that allocates a single **contiguous** section of **memory** to a process or a file.
- Because of this all the available memory space resides at the same place together, which means that the freely/unused available memory partitions are not distributed in a random fashion here and there across the whole memory space.

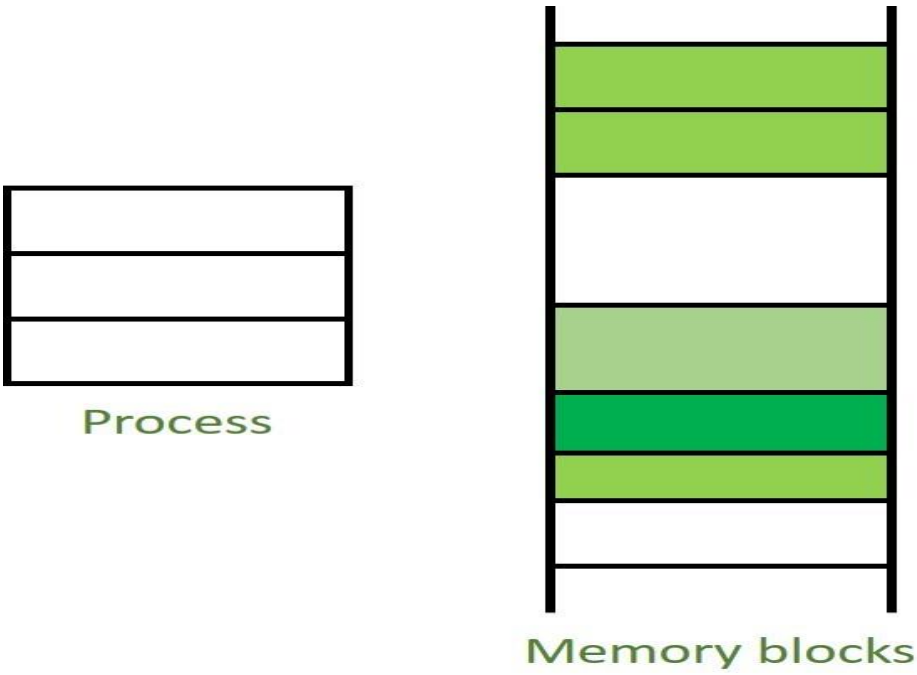


Contiguous Memory Allocation

non contiguous memory allocation

Non-Contiguous Memory Allocation :

- Non-Contiguous memory allocation is basically a method on the contrary to a contiguous allocation method, allocating the memory space present in different locations to the process as per it’s requirements.
- As all the available memory space is in a distributed pattern so the freely available memory space is also scattered here and there.
- This technique of memory allocation helps to reduce the wastage of memory, which eventually gives rise to Internal and external fragmentation.



Noncontiguous Memory Allocation

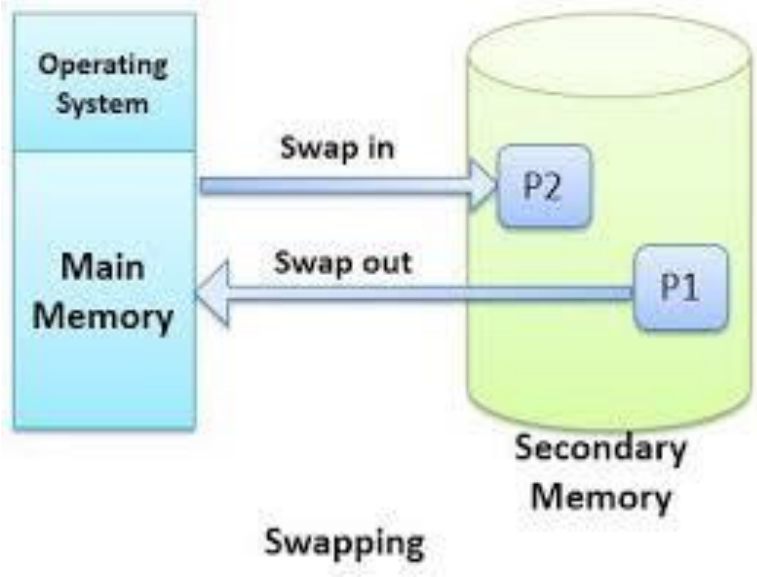
Difference between Contiguous and Non-contiguous Memory Allocation :

S.NO.	Contiguous Memory Allocation	Non-Contiguous Memory Allocation
1.	Contiguous memory allocation allocates consecutive blocks of memory to a file/process.	Non-Contiguous memory allocation allocates separate blocks of memory to a file/process.
2.	Faster in Execution.	Slower in Execution.
3.	It is easier for the OS to control.	It is difficult for the OS to control.
4.	Overhead is minimum as not much address translations are there while executing a process.	More Overheads are there as there are more address translations.

5.	Internal fragmentation occurs in Contiguous memory allocation method.	External fragmentation occurs in Non-Contiguous memory allocation methods.
6.	It includes single partition allocation and multi-partition allocation.	It includes paging and segmentation.
7.	Wastage of memory is there.	No memory wastage is there.
8.	In contiguous memory allocation, swapped-in processes are arranged in the originally allocated space.	In non-contiguous memory allocation, swapped-in processes can be arranged in any place in the memory.

3.2 Swapping

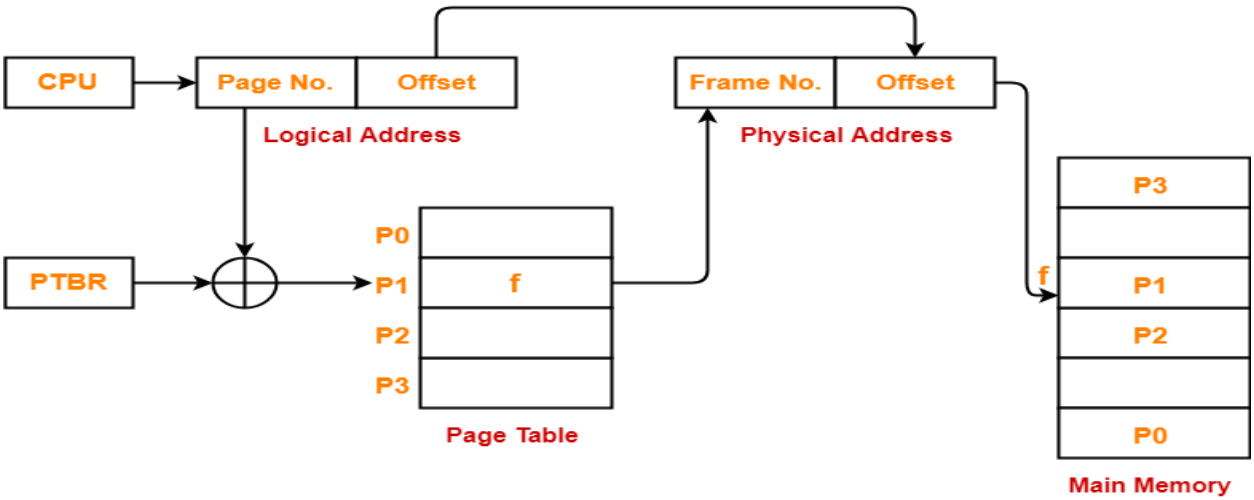
- Swapping is a mechanism in which a process can be swapped temporarily out of main memory (or move) to secondary storage (disk) and make that memory available to other processes.
- At some later time, the system swaps back the process from the secondary storage to main memory.



Swapping

3.3 Paging

- Paging is a memory management scheme that eliminates the need for contiguous allocation of physical memory.
- This scheme permits the physical address space of a process to be non – contiguous.



Translating Logical Address into Physical Address

3.4Segmentation

- In Operating Systems, Segmentation is a memory management technique in which the memory is divided into the variable size parts.
- Each part is known as a segment which can be allocated to a process.
- The details about each segment are stored in a table called a segment table.

- Segmentation gives the user's view of the process which paging does not give. Here the user's view is mapped to physical memory.

There are types of segmentation:

1. **Virtual memory segmentation –**

   Each process is divided into a number of segments, not all of which are resident at any one point in time.

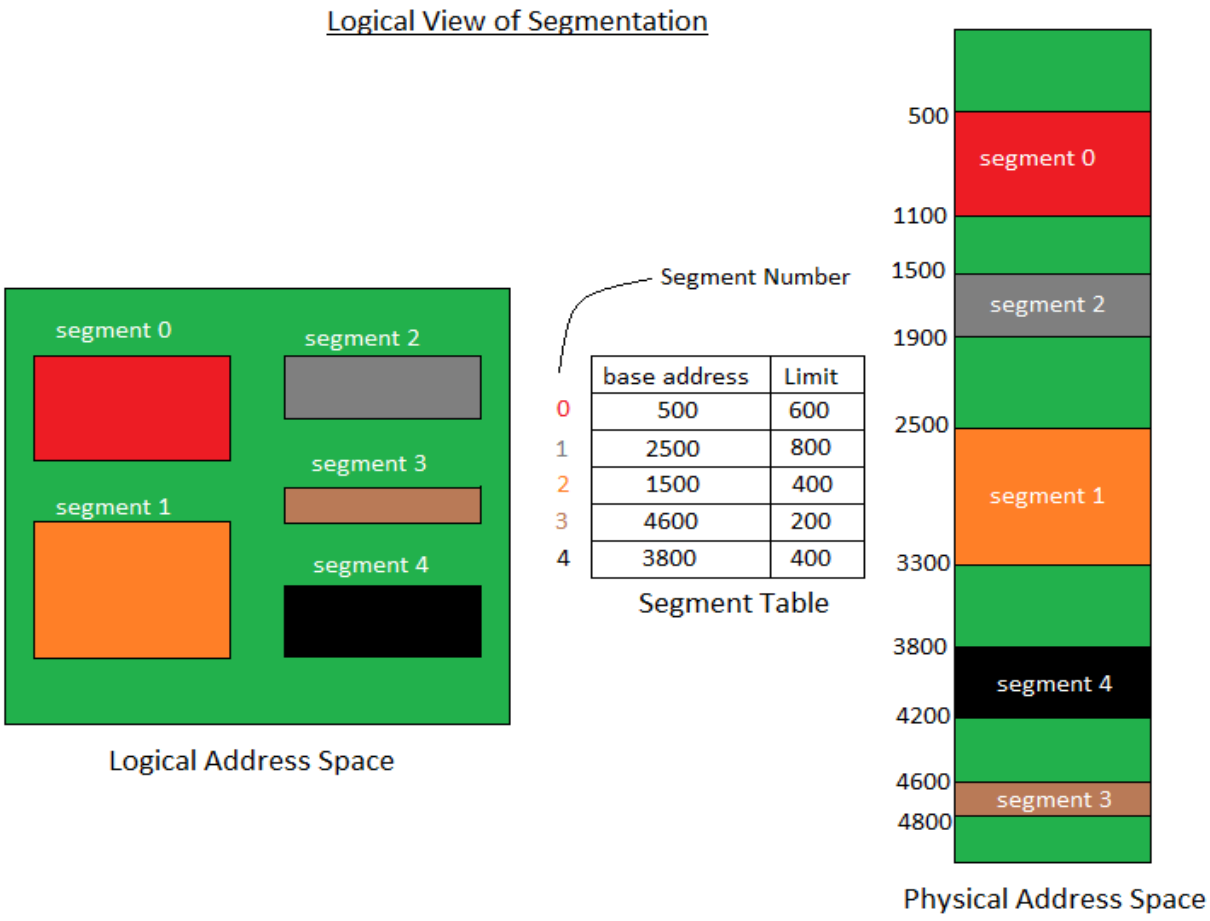
2. **Simple segmentation –**

   Each process is divided into a number of segments, all of which are loaded into memory at run time, though not necessarily contiguously.

- There is no simple relationship between logical addresses and physical addresses in segmentation. A table stores the information about all such segments and is called Segment Table.

Segment Table – It maps a two-dimensional Logical address into one-dimensional Physical address. It's each table entry has:

- Base Address: It contains the starting physical address where the segments reside in memory.
- Limit: It specifies the length of the segment.

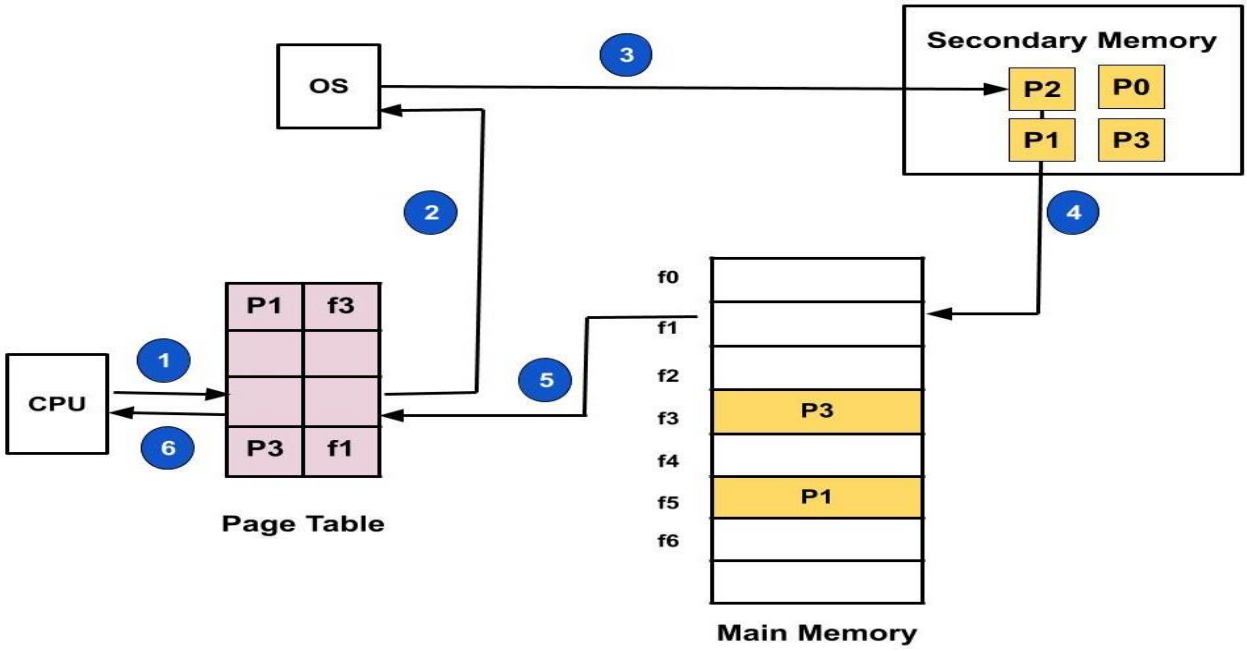


Virtual memory using paging

- Paging and segmentation are processes by which data is stored to, then retrieved from, a computer's storage disk.
- Paging is a computer memory management function that presents storage locations to the computer's CPU as additional memory, called **virtual memory**.
- Each piece of data needs a storage address.

Demand paging

- In a system that uses demand paging, the operating system copies a disk page into physical memory only if an attempt is made to access it and that page is not already in memory (i.e., if a page fault occurs).



Short questions with answer

1. What is swapping?

- Swapping is a mechanism in which a process can be swapped temporarily out of main memory (or move) to secondary storage (disk) and make that memory available to other processes.
- At some later time, the system swaps back the process from the secondary storage to main memory.

2. What is segmentation?

- In Operating Systems, Segmentation is a memory management technique in which the memory is divided into the variable size parts.
- Each part is known as a segment which can be allocated to a process.
- The details about each segment are stored in a table called a segment table.

3. Define paging ?

- Paging is a memory management scheme that eliminates the need for contiguous allocation of physical memory.
- This scheme permits the physical address space of a process to be non – contiguous.

Long Questions

1) **Differentiate contiguous and non-contiguous memory location.**
2) **Explain various types of segmentation.**
3) **Explain Swapping.**
4) **Describe paging technique.**

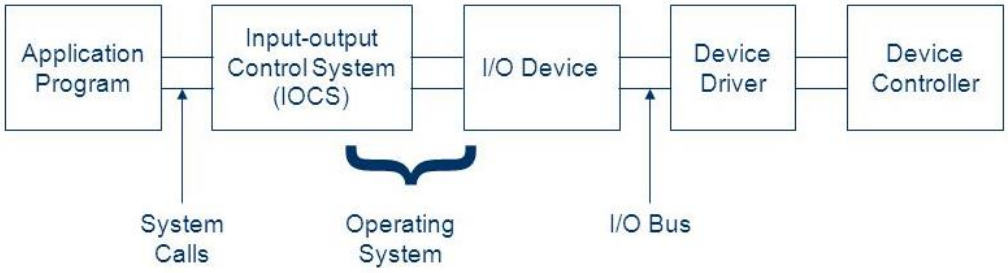
CHAPTER-04.
DEVICE MANAGEMENT

4.1 Techniques for Device Management

There are basically three techniques for managing and allocating devices.

1. **Dedicated**
   - ➢ A dedicated device is allocated to a process for the processes and entire duration.
   - ➢ This technique is difficult.
   - ➢ dedicated devices may be efficient if the process Does not fully or continuously utilise the device.
2. **Shared**
   - ➢ It can be shared concurrently by several processes.
   - ➢ The management of shared devices may become complicated.
3. **Virtual.**
   - ➢ Devices that would normally had to be dedicated may be converted into shared devices. Such devices are known as virtual devices.
   - ➢ This virtual device technique can be applicable to a large number of peripheral devices.

4.2 Device allocation considerations



(Structure of I/O)

Device Driver- It is a program software that links at Peripheral device to an Operating System.

I/O CS- By the help of control system various input output Communications are Performed. I/O CS manages the input output interrupt which takes place. I/O CS has 2 main parts

1) I/O Traffic controller
2) I/O Scheduler

1. I/O Traffic controller- :
   - ➢ In OS the primary role of IO traffic controller is to coordinate and control the action of hardware & software in CPU including the I/O and storage devices.
   - ➢ CPU processes the data via the traffic controller and allows the simultaneous working processing of IO devices.
   - ➢ IO traffic controller is a software.

2. I/O Scheduler-
   - ➢ IO scheduler is a software.
   - ➢ IO scheduler is used to describe the method of and OS decides the order that block IO operation which needs to be submitted to storage spaces.
   - ➢ IOS scheduler Is also called as disk scheduler.

Function of I/O scheduler

- ➢ **It minimises the time wastage by hard disk.**
- ➢ **It prioritise a certain process I/O request.**

Device Handler

- ➢ Device Handlers are software that interface between OS and I/O devices.
- ➢ When I/O instruction is received, Device Handler converts generic instruction and internal character,character code into format required by specific device.

4.3 SPOOLING.

- ➢ Spooling is an acronym for **simultaneous peripheral operations on line**.
- ➢ Spooling refers to putting data of various I/O jobs in a buffer. This buffer is a special area in memory or hard disk which is accessible to I/O devices.

Advantages of Spooling

- Since there is no interaction of I/O devices with CPU, so the CPU need not wait for the I/O operation to take place. The I/O operations take a large amount of time.
- The CPU is kept busy most of the time and hence it is not in the idle state which is good to have a situation.
- More than one I/O device can work simultaneously.

Short Questions with answer

Q.What 1is I/O traffic controller ?

1. In OS the primary role of IO traffic controller is to coordinate and control the action of hardware & software in CPU including the I/O and storage devices.
2. CPU processes the data via the traffic controller and allows the simultaneous working processing of IO devices.
3. IO traffic controller is a software.

Q2. What is spooling?

➢ Spooling is an acronym for **simultaneous peripheral operations on line**.

➢ Spooling refers to putting data of various I/O jobs in a buffer. This buffer is a special area in memory or hard disk which is accessible to I/O devices.

Q3. What is device handler?

➢ Device Handlers are software that interface between OS and I/O devices.

➢ When I/O instruction is received, Device Handler converts generic instruction and internal character,character code into format required by specific device.

Long Questions

1) What are the different types of device management techniques?

2) What Is device allocation concept? explain the function of IO traffic controller and IO scheduler.
3) Describe spooling. state its advantages.

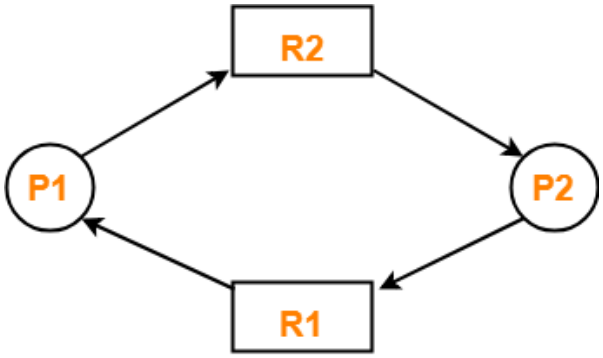
Chapter5

DEAD LOCKS

5.1 Concept of Deadlock -

Deadlock is a situation where-

- The execution of two or more processes is blocked because each process holds some resource and waits for another resource held by some other process.



Example of a deadlock

Here

- Process P1 holds resource R1 and waits for resource R2 which is held by process P2.
- Process P2 holds resource R2 and waits for resource R1 which is held by process P1.
- None of the two processes can complete and release their resource.
- Thus, both the processes keep waiting infinitely.

5.2 System Model

Conditions For Deadlock-

There are following 4 necessary conditions for the occurrence of deadlock-

1. Mutual Exclusion
2. Hold and Wait
3. No preemption
4. Circular wait

1. Mutual Exclusion-

By this condition,

- There must exist at least one resource in the system which can be used by only one process at a time.
- If there exists no such resource, then deadlock will never occur.
- Printer is an example of a resource that can be used by only one process at a time.

2. Hold and Wait-

By this condition,

- There must exist a process which holds some resource and waits for another resource held by some other process.

3. No Preemption-

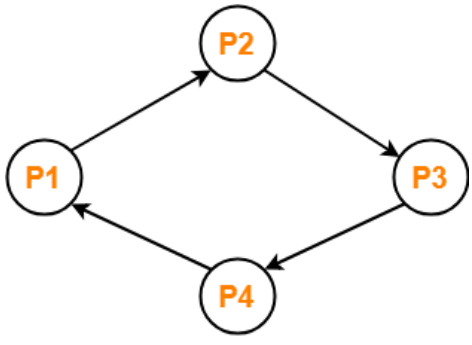
By this condition,

- Once the resource has been allocated to the process, it can not be preempted.
- It means resource can not be snatched forcefully from one process and given to the other process.
- The process must release the resource voluntarily by itself.

4. Circular Wait-

By this condition,

- All the processes must wait for the resource in a cyclic manner where the last process waits for the resource held by the first process.



Circular Wait

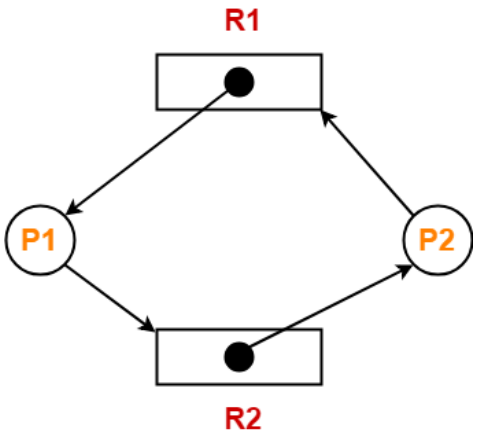
5.3 Dead Lock Detection

- This strategy involves waiting until a deadlock occurs.
- After deadlock occurs, the system state is recovered.
- The main challenge with this approach is detecting the deadlock.

Using Resource Allocation Graph, it can be easily detected whether system is in a Deadlock state or not.

Method-01:

- The given resource allocation graph is single instance with a cycle contained in it.
- Thus, the system is definitely in a deadlock state.



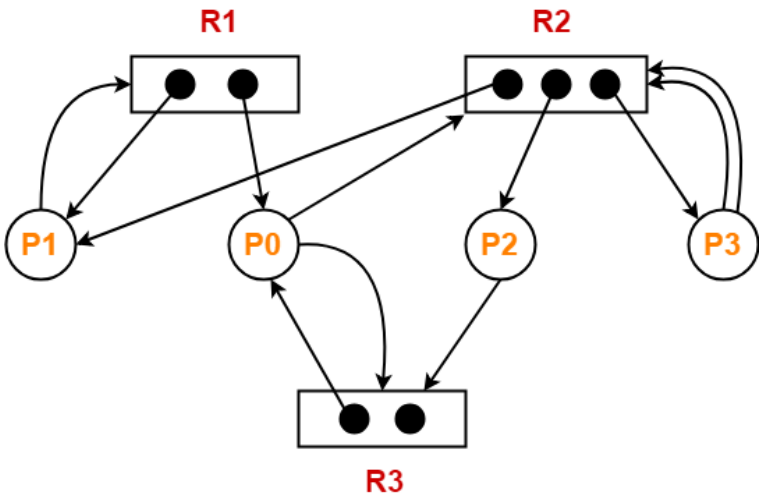
Find if the system is in a deadlock state otherwise find a safe sequence.

- The given resource allocation graph is single instance with a cycle contained in it.
- Thus, the system is definitely in a deadlock state.

Method 2

In a Resource Allocation Graph where all the resources are NOT single instance,

- If a cycle is being formed, then system may be in a deadlock state.
- Banker's Algorithm is applied to confirm whether system is in a deadlock state or not.
- If no cycle is being formed, then system is not in a deadlock state.
- Presence of a cycle is a necessary but not a sufficient condition for the occurrence of deadlock.



Find if the system is in a deadlock state otherwise find a safe sequence.

- The given resource allocation graph is multi instance with a cycle contained in it.
- So, the system may or may not be in a deadlock state.

Using the given resource allocation graph, we have-

	Allocation			Need		
	R1	**R2**	**R3**	**R1**	**R2**	**R3**
Process P0	1	0	1	0	1	1
Process P1	1	1	0	1	0	0
Process P2	0	1	0	0	0	1
Process P3	0	1	0	0	2	0

Available = [R1 R2 R3] = [0 0 1]

Step-01:

- With the instances available currently, only the requirement of the process P2 can be satisfied.
- So, process P2 is allocated the requested resources.
- It completes its execution and then free up the instances of resources held by it.

Then-

Available

= [0 0 1] + [0 1 0]

= [0 1 1]

Step-02:

- With the instances available currently, only the requirement of the process P0 can be satisfied.
- So, process P0 is allocated the requested resources.
- It completes its execution and then free up the instances of resources held by it.

Then-

Available

= [0 1 1] + [1 0 1]

= [1 1 2]

Step-03:

- With the instances available currently, only the requirement of the process P1 can be satisfied.
- So, process P1 is allocated the requested resources.
- It completes its execution and then free up the instances of resources held by it.

Then-

Available

= [1 1 2] + [1 1 0]

= [2 2 2]

Step-04:

- With the instances available currently, the requirement of the process P3 can be satisfied.
- So, process P3 is allocated the requested resources.
- It completes its execution and then free up the instances of resources held by it.

Then-

Available

= [2 2 2] + [0 1 0]

= [2 3 2]

Thus,

- There exists a safe sequence P2, P0, P1, P3 in which all the processes can be executed.
- So, the system is in a safe state.

5.4 Resources allocation Graph

- Resource Allocation Graph (RAG) is a graph that represents the state of a system pictorially.

- It gives complete information about the state of a system such as-

1. How many processes exist in the system?
2. How many instances of each resource type exist?
3. How many instances of each resource type are allocated?
4. How many instances of each resource type are still available?
5. How many instances of each resource type are held by each process?
6. How many instances of each resource type does each process need for execution?

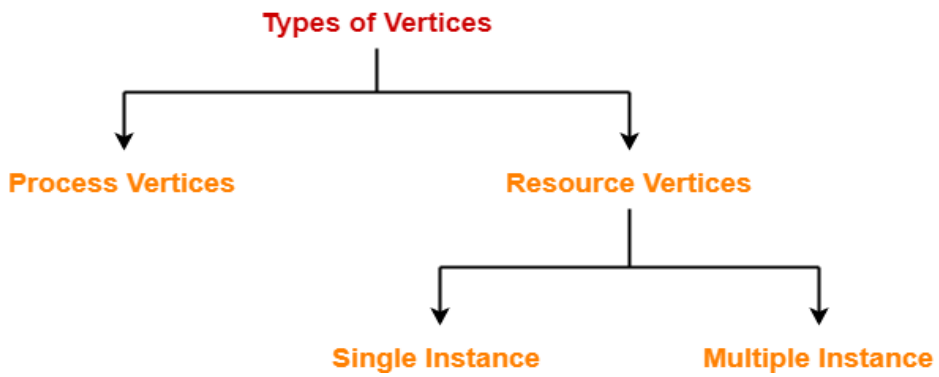
Components Of RAG-

There are two major components of a Resource Allocation Graph-

1. Vertices
2. Edges

Vertices-

There are following types of vertices in a Resource Allocation Graph-



1. Process Vertices
2. Resource Vertices

Process Vertices-

- Process vertices represent the processes.
- They are drawn as a circle by mentioning the name of process inside the circle.

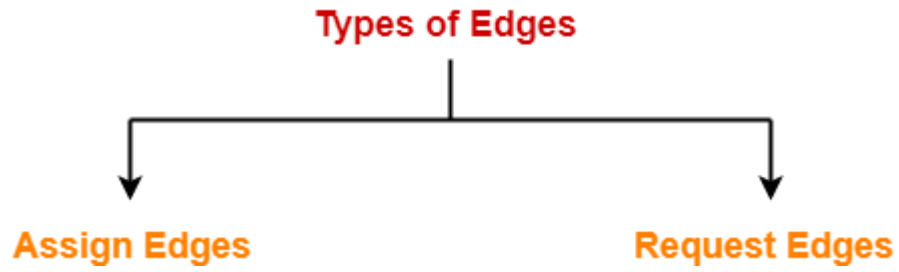
Resource Vertices-

- Resource vertices represent the resources.
- Depending on the number of instances that exists in the system, resource vertices may be single instance or multiple instance.

- They are drawn as a rectangle by mentioning the dots inside the rectangle.
- The number of dots inside the rectangle indicates the number of instances of that resource existing in the system.

Edges-

There are two types of edges in a Resource Allocation Graph-



1. Assign Edges
2. Request Edges

Assign Edges-

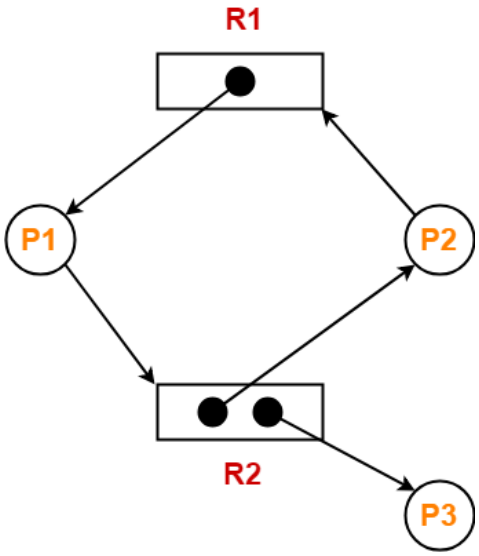
- Assign edges represent the assignment of resources to the processes.
- They are drawn as an arrow where the head of the arrow points to the process and tail of the process points to the instance of the resource.

Request Edges-

- Request edges represent the waiting state of processes for the resources.
- They are drawn as an arrow where the head of the arrow points to the instance of the resource and tail of the process points to the process.
- If a process requires ‘n’ instances of a resource type, then ‘n’ assign edges will be drawn.

Example Of RAG-

The following diagram represents a Resource Allocation Graph-

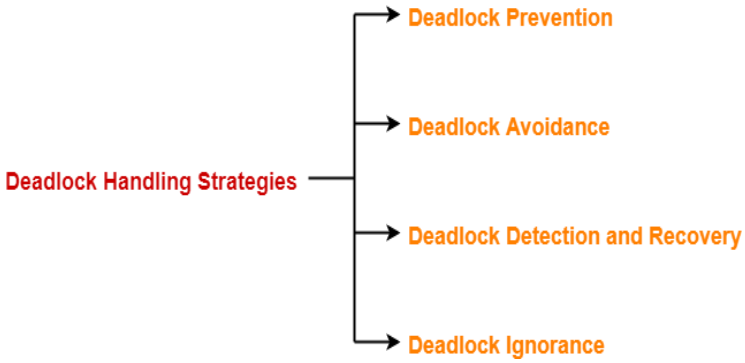


It gives the following information-

- There exist three processes in the system namely P1, P2 and P3.
- There exist two resources in the system namely R1 and R2.
- There exists a single instance of resource R1 and two instances of resource R2.
- Process P1 holds one instance of resource R1 and is waiting for an instance of resource R2.
- Process P2 holds one instance of resource R2 and is waiting for an instance of resource R1.
- Process P3 holds one instance of resource R2 and is not waiting for anything

5.5 Methods of Deadlock handling

The various strategies for handling deadlock are-



1. Deadlock Prevention
2. Deadlock Avoidance
3. Deadlock Detection and Recovery
4. Deadlock Ignorance

5.6 Recovery &Prevention,Explain Bankers Algorithm & Safety Algorithm

Deadlock Prevention-

- This strategy involves designing a system that violates one of the four necessary conditions required for the occurrence of deadlock.
- This ensures that the system remains free from the deadlock.

The various conditions of deadlock occurrence may be violated as-

1. Mutual Exclusion-

- To violate this condition, all the system resources must be such that they can be used in a shareable mode.
- In a system, there are always some resources which are mutually exclusive by nature.
- So, this condition can not be violated.

2. Hold and Wait-

This condition can be violated in the following ways-

Approach-01:

In this approach,

- A process has to first request for all the resources it requires for execution.
- Once it has acquired all the resources, only then it can start its execution.
- This approach ensures that the process does not hold some resources and wait for other resources.

Drawbacks-

The drawbacks of this approach are-

- It is less efficient.
- It is not implementable since it is not possible to predict in advance which resources will be required during execution.

Approach-02:

In this approach,

- A process is allowed to acquire the resources it desires at the current moment.
- After acquiring the resources, it start its execution.
- Now before making any new request, it has to compulsorily release all the resources that it holds currently.
- This approach is efficient and implementable.

Approach-03:

In this approach,

- A timer is set after the process acquires any resource.
- After the timer expires, a process has to compulsorily release the resource.

3. No Preemption-

- This condition can by violated by forceful preemption.
- Consider a process is holding some resources and request other resources that can not be immediately allocated to it.
- Then, by forcefully preempting the currently held resources, the condition can be violated.

A process is allowed to forcefully preempt the resources possessed by some other process only if-

- It is a high priority process or a system process.
- The victim process is in the waiting state.

4. Circular Wait-

- This condition can be violated by not allowing the processes to wait for resources in a cyclic manner.
- To violate this condition, the following approach is followed-

Approach-

- A natural number is assigned to every resource.
- Each process is allowed to request for the resources either in only increasing or only decreasing order of the resource number.
- In case increasing order is followed, if a process requires a lesser number resource, then it must release all the resources having larger number and vice versa.
- This approach is the most practical approach and implementable.
- However, this approach may cause starvation but will never lead to deadlock.

Deadlock Detection and Recovery-

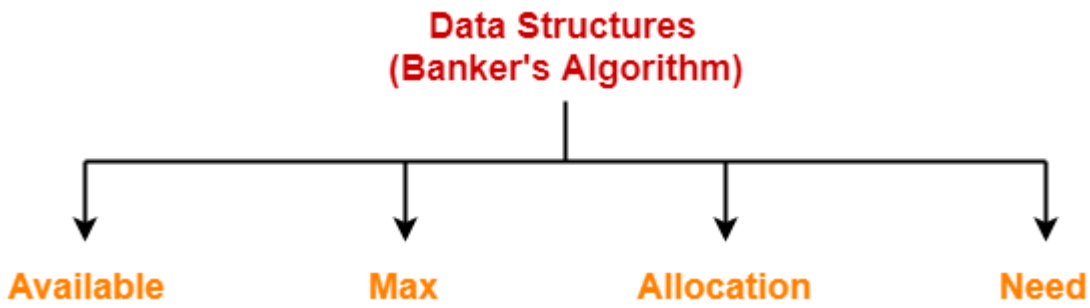
- This strategy involves waiting until a deadlock occurs.
- After deadlock occurs, the system state is recovered.
- The main challenge with this approach is detecting the deadlock.

Banker's Algorithm-

- Banker's Algorithm is a deadlock avoidance strategy.
- It is called so because it is used in banking systems to decide whether a loan can be granted or not.

Data Structures Used-

- To implement banker's algorithm, following four data structures are used-



- **Available**
- **Max**
- **Allocation**
- **Need**

Working-

- Banker's Algorithm is executed whenever any process puts forward the request for allocating the resources.
- It involves the following steps-

Step-01:

- Banker's Algorithm checks whether the request made by the process is valid or not.
- If the request is invalid, it aborts the request.
- If the request is valid, it follows step-02.

Step-02:

- Banker's Algorithm checks if the number of requested instances of each resource type is less than the number of available instances of each type.
- If the sufficient number of instances are not available, it asks the process to wait longer.
- If the sufficient number of instances are available, it follows step-03.

Step-03:

- Banker's Algorithm makes an assumption that the requested resources have been allocated to the process.
- Then, it modifies its data structures accordingly and moves from one state to the other state.

Available = Available - Request(i)
Allocation(i) = Allocation(i) + Request(i)
Need(i) = Need(i) - Request(i)

- Now, Banker's Algorithm follows the safety algorithm to check whether the resulting state it has entered in is a safe state or not.
- If it is a safe state, then it allocates the requested resources to the process in actual.

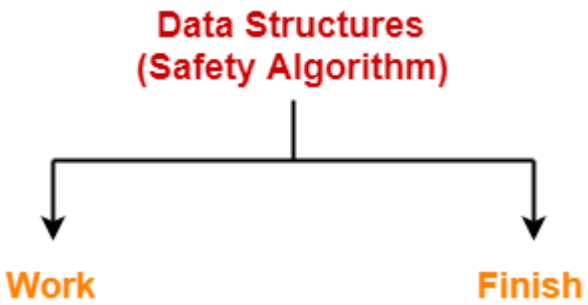
- If it is an unsafe state, then it rollbacks to its previous state and asks the process to wait longer.

Safe State

A system is said to be in safe state when- All the processes can be executed in some arbitrary sequence with the available number of resources.

Safety Algorithm Data Structures-

To implement safety algorithm, following two data structures are used-



Safety Algorithm-

Safety Algorithm is executed to check whether the resultant state after allocating the resources is safe or not.

Step-01: Initially-

- Number of instances of each resource type currently available = Available
- All the processes are to be executed.
- So, in Step-01, the data structures are initialized as-

Step-02:

- Safety Algorithm looks for an unfinished process whose need is less than or equal to work.
- So, Step-02 finds an index i such that-f such a process exists, then step-03 is followed otherwise step-05 is followed.

Step-03:

- After finding the required process, safety algorithm assumes that the requested resources are allocated to the process.
- The process runs, finishes its execution and the resources allocated to it gets free.

The resources are then added to the work and finish(i) of that process is set as true.

Step-04:

- The loop of Step-02 and Step-03 is repeated.

Step-05:

- If all the processes can be executed in some sequence, then the system is said to be in a safe state.
- In other words, if Finish(i) becomes True for all i, then the system is in a safe state otherwise not.
-

Short Questions with answer

Q1. Define deadlock.

Deadlock is a situation where-

- The execution of two or more processes is blocked because each process holds some resource and waits for another resource held by some other process.

Q2. Write down 4 conditions of deadlock.

1. Mutual exclusion
2. Hold and wait
3. Pre-emption
4. Circular wait

Long Questions

Q1. Explain Bankers algorithm.

Q2. Explain deadlock prevention method.

Q3. Explain deadlock with a suitable example.

Q4. What is safety algorithm. Explain with examples.

Q5. How to detect and recover deadlock?

CHAPTER 6
FILE MANAGEMENT

6.1 File organization, Directory & file structure, sharing of files

File

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.

File Oraganization

File organization refers to the way data is stored in a file. File organization is very important because it determines the methods of access, efficiency, flexibility and storage devices to use.

FILE DIRECTORIES:

Collection of files is a file directory. The directory contains information about the files, including attributes, location and ownership. Much of this information, especially that is concerned with storage, is managed by the operating system. The directory is itself a file, accessible by various file management routines.

File Structure

A File Structure should be according to a required format that the operating system can understand.

- A file has a certain defined structure according to its type.
- A text file is a sequence of characters organized into lines.
- A source file is a sequence of procedures and functions.

- ❖ An object file is a sequence of bytes organized into blocks that are understandable by the machine.

When operating system defines different file structures, it also contains the code to support these file structure. Unix, MS-DOS support minimum number of file structure.

6.2 File access methods, file systems, reliability

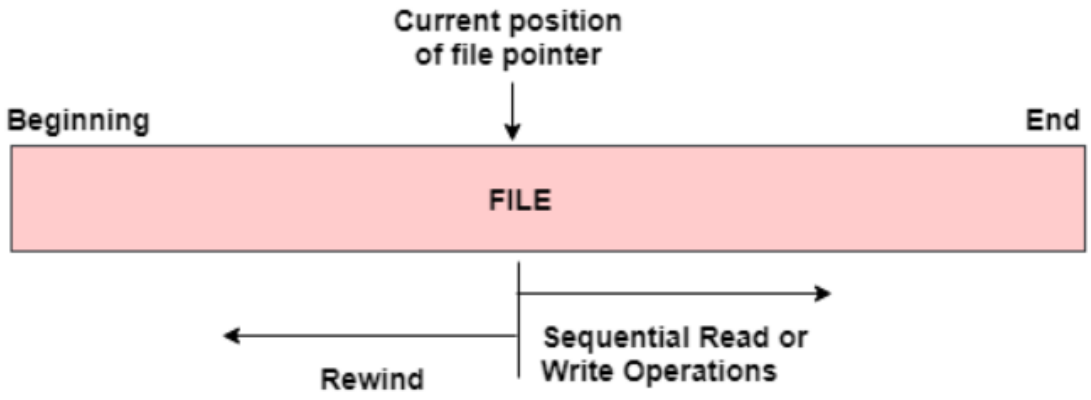
File access methods:

There are three ways to access a file into a computer system:

1. Sequential-Access,
2. Direct Access,
3. Index sequential Method.

1. Sequential-Access

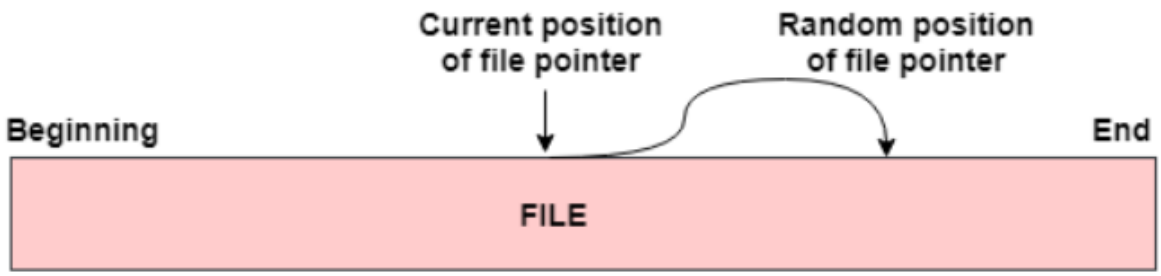
- The information in a file is processed in order using sequential access.
- The files records are accessed on after another.
- Most of the file systems such as editors, compilers etc. use sequential access.
- It is based on the tape model of a file and so can be used with sequential access devices as well as random access devices.



SEQUENTIAL ACCESS IN FILES

2. Direct Access

- In direct access or relative access files can be accessed in random for read and write operations.
- The direct access model is based on the disk model of a file, since it allows random accesses.
- In this method, the file is divided into numbered blocks.
- Any of these arbitrary blocks can be read or written.
- For example, we may read block 8, then write into block 10 and then read block 15. Direct access system is quite useful and mostly databases are of this type.



DIRECT ACCESS IN FILES

3.Index sequential method

- ❖ It is the other method of accessing a file which is built on the top of the sequential access method.
- ❖ These methods construct an index for the file.
- ❖ The index, like an index in the back of a book, contains the pointer to the various blocks. To find a record in the file, we first search the index and then by the help of pointer we access the file directly.

Key points:

- ➔ It is built on top of Sequential access.
- ➔ It control the pointer by using index.

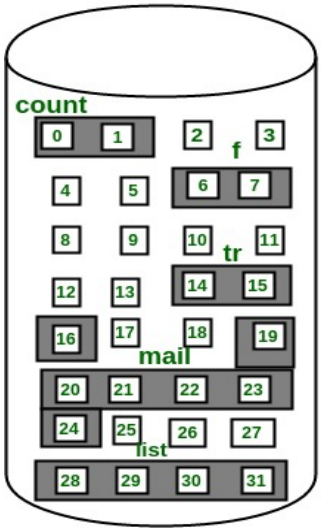
File System and reliability

6.3 Allocation of Disk Space

1. Continuous Allocation:

- ❖ In this scheme, each file occupies a contiguous set of blocks on the disk.
- ❖ For example, if a file requires n blocks and is given a block b as the starting location, then the blocks assigned to the file will be: *b, b+1, b+2,……b+n-1.*

- This means that given the starting block address and the length of the file (in terms of blocks required), we can determine the blocks occupied by the file.
- The directory entry for a file with contiguous allocation contains
  - Address of starting block
  - Length of the allocated portion.
- The *file 'mail'* in the following figure starts from the block 19 with length = 6 blocks. Therefore, it occupies *19, 20, 21, 22, 23, 24* blocks.



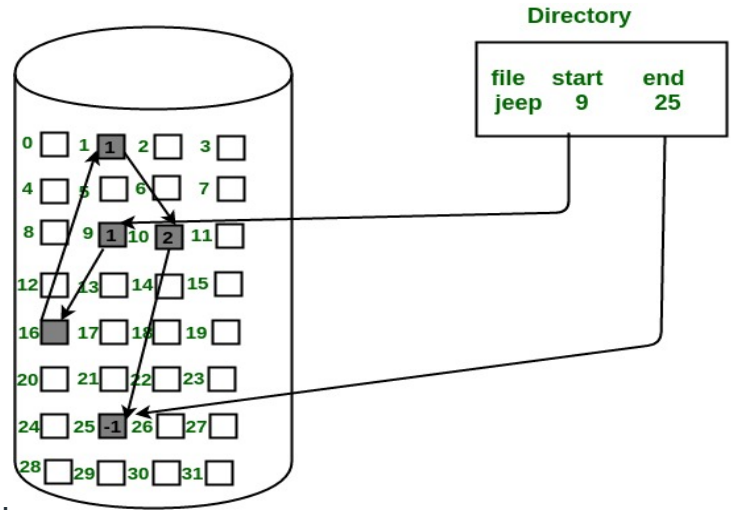
Directory

file	start	length
count	0	2
tr	14	3
mail	19	6
list	28	4
f	6	2

2. Linked Allocation(Non-contiguous allocation) :

- In this scheme, each file is a linked list of disk blocks which **need not be** contiguous.
- The disk blocks can be scattered anywhere on the disk.
- The directory entry contains a pointer to the starting and the ending file block.

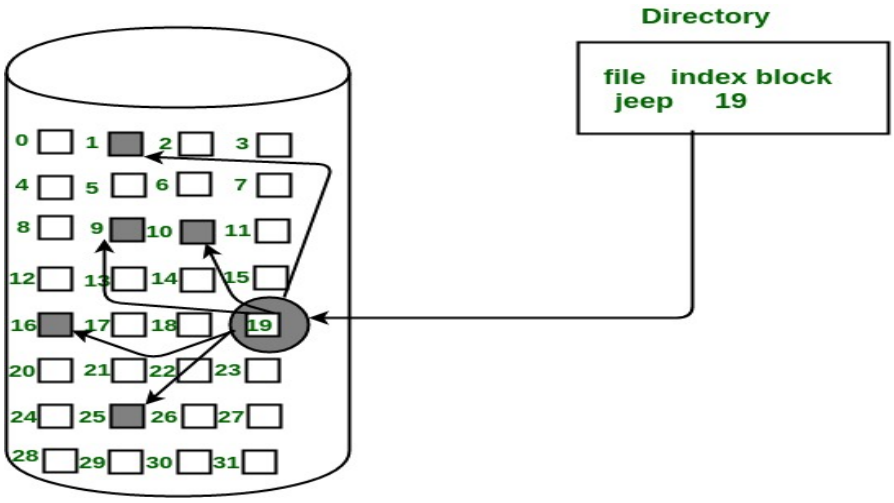
- Each block contains a pointer to the next block occupied by the file.
- The file ‘jeep’ in following image shows how the blocks are randomly distributed. The last block (25) contains -1 indicating a null pointer and does not point to any other block.



.

3. Indexed Allocation:

- In this scheme, 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.
- The ith entry in the index block contains the disk address of the ith file block.
- The directory entry contains the address of the index block as shown in the image:



6.4 File protection, secondary storage management

- Protection and security requires that computer resources such as CPU, softwares, memory etc. are protected.
- This extends to the operating system as well as the data in the system.
- This can be done by ensuring integrity, confidentiality and availability in the operating system.
- The system must be protected against unauthorized access, viruses, worms etc.

Threats to Protection and Security

A threat is a program that is malicious in nature and leads to harmful effects for the system. Some of the common threats that occur in a system are −

Virus

Viruses are generally small snippets of code embedded in a system. They are very dangerous and can corrupt files, destroy data, crash systems etc. They can also spread further by replicating themselves as required.

Trojan Horse

A trojan horse can secretly access the login details of a system. Then a malicious user can use these to enter the system as a harmless being and wreak havoc.

Trap Door

A trap door is a security breach that may be present in a system without the knowledge of the users. It can be exploited to harm the data or files in a system by malicious people.

Worm

A worm can destroy a system by using its resources to extreme levels. It can generate multiple copies which claim all the resources and don't allow any other processes to access them. A worm can shut down a whole network in this way.

Denial of Service

These type of attacks do not allow the legitimate users to access a system. It overwhelms the system with requests so it is overwhelmed and cannot work properly for other user.

Protection and Security Methods

The different methods that may provide protect and security for different computer systems are −

Authentication

This deals with identifying each user in the system and making sure they are who they claim to be. The operating system makes sure that all the users are authenticated before they access the system. The different ways to make sure that the users are authentic are:

- **Username/ Password**
  Each user has a distinct username and password combination and they need to enter it correctly before they can access the system.

- **User Key/ User Card**
  The users need to punch a card into the card slot or use they individual key on a keypad to access the system.
- **User Attribute Identification**
  Different user attribute identifications that can be used are fingerprint, eye retina etc. These are unique for each user and are compared with the existing samples in the database. The user can only access the system if there is a match.

One Time Password

These passwords provide a lot of security for authentication purposes. A one time password can be generated exclusively for a login every time a user wants to enter the system. It cannot be used more than once. The various ways a one time password can be implemented are −

- **Random Numbers**
  The system can ask for numbers that correspond to alphabets that are pre arranged. This combination can be changed each time a login is required.
- **Secret Key**
  A hardware device can create a secret key related to the user id for login. This key can change each time.

Short Questions

Q1.What is file?

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.

Q2. What is file organization?

File organization refers to the way data is stored in a file. File organization is very important because it determines the methods of access, efficiency, flexibility and storage devices to use

Q3. What are the file access methods?

There are three ways to access a file into a computer system:

1. Sequential-Access,
2. Direct Access,
3. Index sequential Method.

Long Questions

Q1. What are file allocation methods?

Q2. How Protection is maintained by OS?

CHAPTER 7

SYSTEM PROGRAMMING

7.1 Concept of system programming and show difference from Application Complier:

Concept Of System Programming:

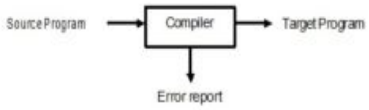
System Programming can be defined as the act of building Systems Software using System Programming Languages.

System Software	Application Software
1. System Software maintain the system resources and give the path for application software to run.	1. Application software is built for specific tasks.
2. Low level languages are used to write the system software.	2. While high level languages are used to write the application software.
3. Its a general purpose software.	3. While its a specific purpose software.
4. Without system software, system can't run.	4. While without application software system always runs.
5. System software runs when system is turned on and stop when system is turned off.	5. While application software runs as per the user's request..
6. Example of system software are operating system, etc.	6. Example of application software are Photoshop, VLC player etc.
7. System Software programming is complex than application software.	7. Application software programming is simpler as comparison to system software.

7.2 Compiler , functions of compiler.

Compiler :

Compiler scans the entire program and translates the whole of it into machine code at once.



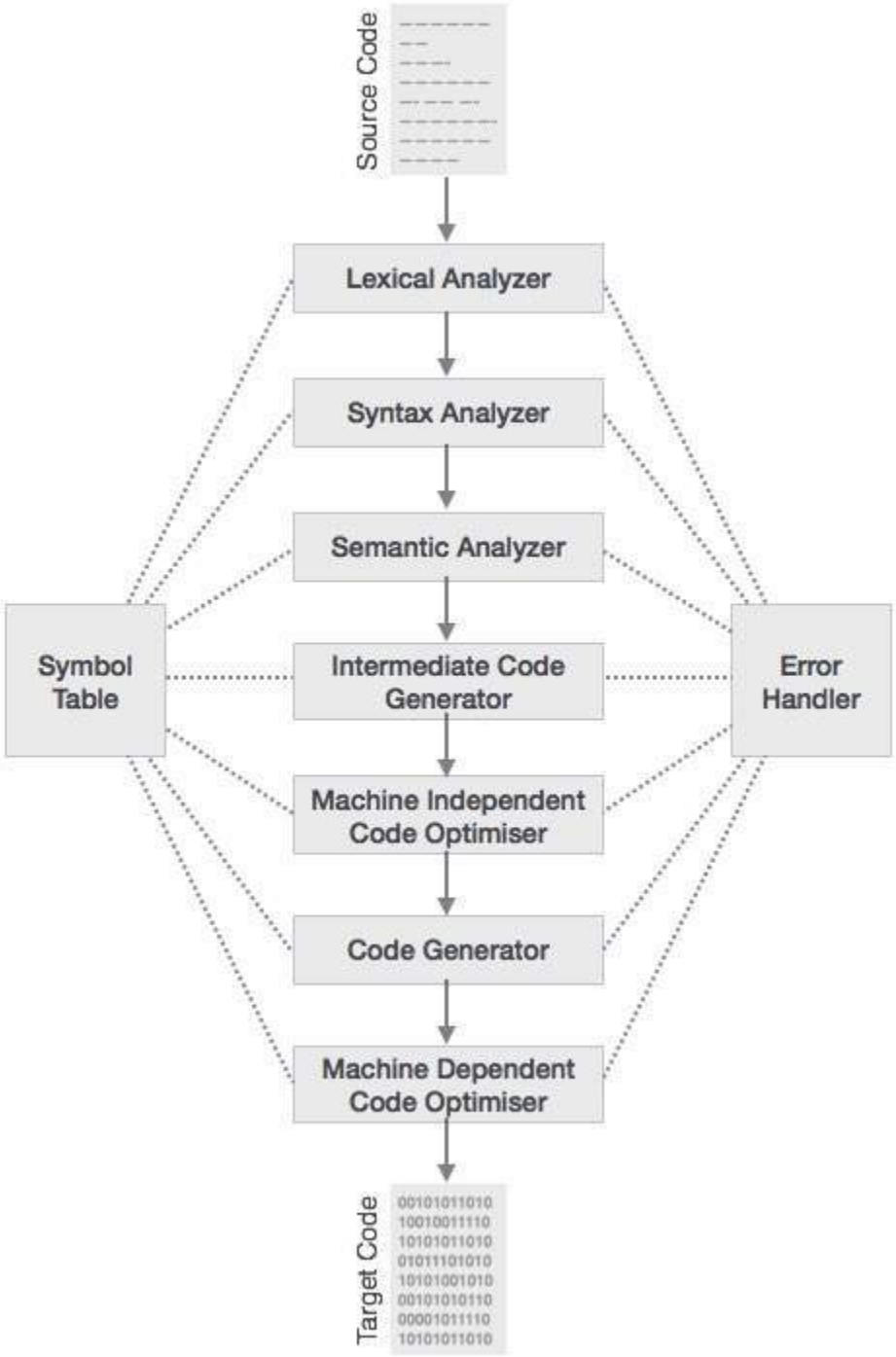
Functions Of compiler:

- Compiler is used to convert one form of program to another
- A compiler should convert the source program to a target machine code in such a way that the generated target code should be easy to understand
- Compiler should preserve the meaning of source code
- Compiler should report errors that occur during compilation process
- The compilation must be done efficiently

7.3 Compare compiler and interpreter

Compiler	Interpreter
1. Compiler scans the whole program in one go.	1. Translates program one statement at a time.
2. As it scans the code in one go, the errors (if any) are shown at the end together.	2.Considering it scans code one line at a time, errors are shown line by line.
3. Main advantage of compilers is it's execution time.	3.Due to interpreters being slow in executing the object code, it is preferred less..
4. It converts the source code into object code.	4. It does not convert source code into object code instead it scans it line by line. .
5. It does not require source code for later execution.	5.It requires source code for later execution.
6. Eg. C, C++, C# etc.	6. Python, Ruby, Perl, SNOBOL, MATLAB, etc.

7.4 Seven phases of compiler, brief description of each phase.



Lexical Analysis

The first phase of scanner works as a text scanner. This phase scans the source code as a stream of characters and converts it into meaningful lexemes. Lexical analyzer represents these lexemes in the form of tokens as:

<token-name, attribute-value>

Syntax Analysis

The next phase is called the syntax analysis or parsing. It takes the token produced by lexical analysis as input and generates a parse tree (or syntax tree). In this phase, token arrangements are checked against the source code grammar, i.e. the parser checks if the expression made by the tokens is syntactically correct.

Semantic Analysis

Semantic analysis checks whether the parse tree constructed follows the rules of language. For example, assignment of values is between compatible data types, and adding string to an integer. Also, the semantic analyzer keeps track of identifiers, their types and expressions; whether identifiers are declared before use or not etc. The semantic analyzer produces an annotated syntax tree as an output.

Intermediate Code Generation

After semantic analysis the compiler generates an intermediate code of the source code for the target machine. It represents a program for some abstract machine. It is in between the high-level language and the machine language. This intermediate code should be generated in such a way that it makes it easier to be translated into the target machine code.

Code Optimization

The next phase does code optimization of the intermediate code. Optimization can be assumed as something that removes unnecessary code lines, and arranges the sequence of statements in order to speed up the program execution without wasting resources (CPU, memory).

Code Generation

In this phase, the code generator takes the optimized representation of the intermediate code and maps it to the target machine language. The code generator translates the intermediate code into a sequence of (generally) re-locatable machine code. Sequence of instructions of machine code performs the task as the intermediate code would do.

Symbol Table

It is a data-structure maintained throughout all the phases of a compiler. All the identifier's names along with their types are stored here. The symbol table makes it easier for the compiler to quickly search the identifier record and retrieve it. The symbol table is also used for scope management.

Short Questions

Q1. What is System Programming:?

System Programming can be defined as the act of building Systems Software using System Programming Languages.

Q2. What are functions of compiler?

- Compiler is used to convert one form of program to another
- A compiler should convert the source program to a target machine code in such a way that the generated target code should be easy to understand
- Compiler should preserve the meaning of source code
- Compiler should report errors that occur during compilation process
- The compilation must be done efficientl

Q3. What are 7 phases of compiler?

- Lexical Analysis
- Syntax Analysis
- Semantic Analysis
- Intermediate Code Generation
- Code Optimization
- Code Generation
- Symbol Table

Long Questions

Q1. What are the 7 phases of Compiler?

Q2. What is difference between compiler and interpreter?

Q3. What is difference between system programming and application program?