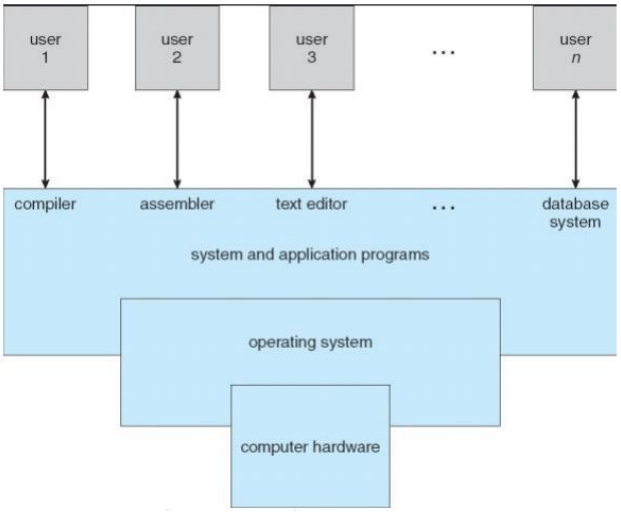
Operating System
By Er. Prashant Shrestha

Unit 1 Introduction to operating system

1.1 introduction to operating system

An Operating System (OS) is software that manages and handles hardware and software resources of a computing device. Operating system acts as an intermediary between a user of a computer and the computer hardware. A computer system can be divided roughly into four components: **the hardware, the operating system, the application program, and the users** as shown in the figure below



1.2 Functions of Operating system

i. Process Management

ii. Memory Management

iii. Device management

iv. File management

v. Resource Management

vi. User interface

vii. Security

i. Process Management

Process management is a crucial function of an operating system that deals with handling all the processes running on a computer. A process is simply a program in execution, and the OS is responsible for creating and terminating these processes. It also manages process scheduling to ensure that each one gets adequate CPU time. Additionally, the OS handles process synchronization and communication, ensuring that multiple processes can work together smoothly without interfering with one another.

ii. Memory Management

Memory management involves managing the computer's primary memory, or RAM. The operating system keeps track of every byte of memory in a system and allocates memory to processes when they need it. When the memory is no longer required, the OS deallocates it to make room for other processes. It also handles situations where the memory is full by using techniques like virtual memory, which temporarily uses hard drive space to simulate additional RAM.

iii. Device Management

Device management is the function of the operating system that controls and coordinates the use of hardware devices such as printers, scanners, keyboards, and disk drives. The OS uses special software called device drivers to communicate with these devices. It ensures that input and output operations are carried out efficiently and that devices are allocated properly among different programs or users.

iv. File Management

File management refers to the way the operating system handles data storage. It organizes data into files and directories, allowing users to easily store, retrieve, and manage their information. The OS also manages file permissions to ensure data security and prevent unauthorized access. It keeps track of where files are located on the storage device and maintains the file system structure.

v. Resource Management

Resource management involves the efficient allocation and control of the system's resources, including the CPU, memory, storage devices, and input/output devices. The operating system ensures that these resources are distributed fairly among all active processes and users. It also prevents conflicts and maximizes the performance of the system by managing resource requests and resolving any contention.

vi. User Interface

The user interface is the interaction between the user and the computer system, provided by the operating system. This can be a Command Line Interface (CLI), where users type text commands, or a Graphical User Interface (GUI), where users interact with visual elements like icons and windows. The user interface makes it easier for people to use the computer without needing to understand complex commands or system operations.

vii. Security

Security is a vital function of the operating system that protects the system and user data from unauthorized access and threats. The OS implements user authentication, such as passwords or biometric verification, to ensure that only authorized individuals can access the system. It also enforces access controls, manages permissions, and may include features like encryption and firewalls to protect against malware, data breaches, and other cyber threats.

1.3 Types of operating system on the basis of processing method

i. Batch processing

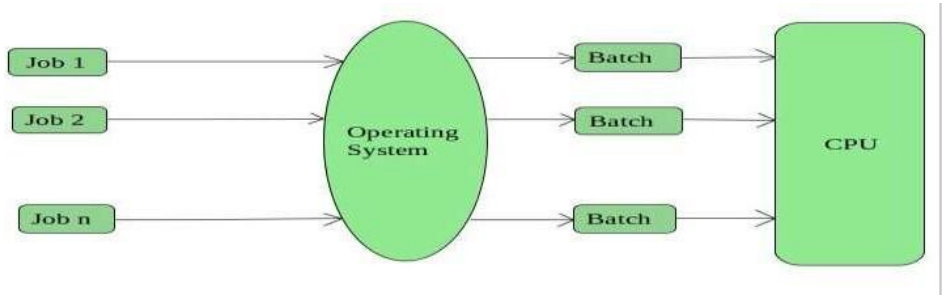
ii. Time sharing operating system

iii. Real time operating system

iv. Distributed Operating System

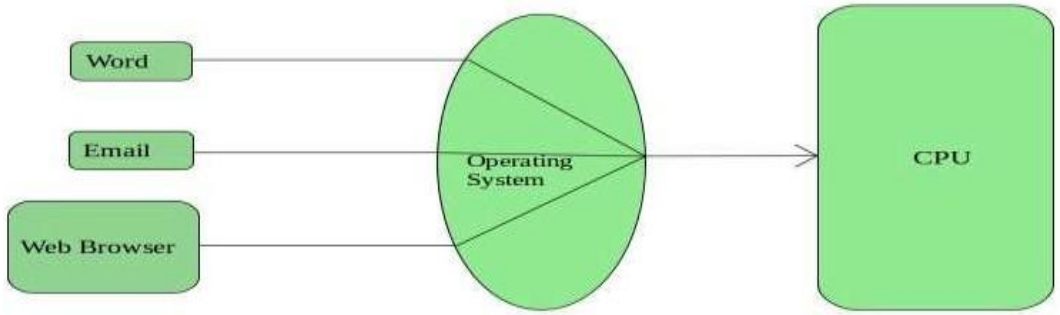
i. Batch Processing

This type of operating system do not interact with the computer directly. There is an operator which takes similar jobs having same requirement and group them into batches. It is the responsibility of operator to sort the jobs with similar needs.



ii. Time sharing operating system

Time-sharing systems allow many users or programs to use the computer at the same time. The CPU quickly switches between each task, giving the impression that everything is happening at once. This system makes computers more efficient and responsive for multiple users. Examples include Windows and UNIX, which allow several programs to run at once.



iii. real time operating system

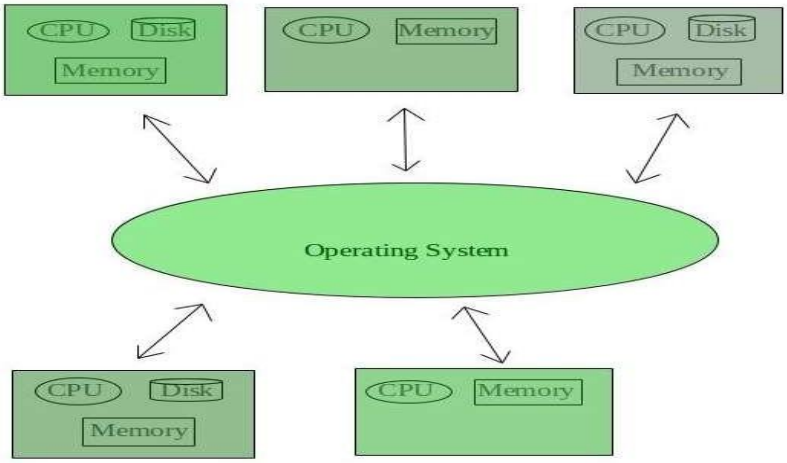
A Real-Time Operating System, or RTOS, is a type of computer system that reacts immediately when something happens. It doesn't wait or get delayed. It doesn't wait or get delayed. It's designed to respond to inputs or events exactly on time—not early, not late. The time taken by the system to respond to an input and display of required updated information is termed as

the **response time**. So in this method, the response time is very less as compared to online processing.

Eg:

Think of a **traffic light system** at a busy intersection. When the timer reaches zero, the light must turn green, or there could be an accident. The system controlling the lights **must** act at the right moment. An RTOS would be used here because it makes sure things happen at **precise times**.

iv. Distributed Operating System



A **Distributed Operating System** is a type of operating system that controls and manages a group of **independent computers** (called **nodes**) connected by a **network** and makes them appear to the user as **one single system**.

Even though the computers are in different places and work independently, the distributed OS lets them communicate, share resources, and work together to complete tasks. This gives users the feeling that they are working on a single, very powerful machine.

1.4 Types of Operating system based on user interface

i. Command user interface

ii. Graphical user interface

i. Command user interface

- A text-based operating system where the user has to type specific commands.
- It Provides direct communication with the OS through command interpreters (e.g., Command Prompt, Bash).

- ❖ It is Very powerful and flexible but requires technical knowledge.
- ❖ Commands need to be memorized, making it less user-friendly for beginners.
- ❖ It is Highly efficient for system administrators, developers, and automation.
- ❖ Examples: MS-DOS, Unix Shell, Linux Terminal.

ii. Graphical user interface

- ❖ It allows interaction with the OS using graphical elements like windows, icons, menus, and pointers.
- ❖ It is User-friendly and easy to learn because it provides visual feedback.
- ❖ It Requires more resources (RAM, CPU, GPU) compared to CLI.
- ❖ It is widely used in personal computers, smartphones, and tablets.
- ❖ It is Suitable for general users who do not need technical command knowledge.
- ❖ Examples: Microsoft Windows, macOS, Linux distributions like Ubuntu

1.5 Types of OS based on mode of user

i. Single-user operating system

ii. Multiple-user operating system

i. Single-user operating system

- ❖ It Designed to be used by one user at a time.
- ❖ It Can either allow:

  a) Single tasking: Only one task can run at a time (e.g., MS-DOS).

  b) Multitasking: One user can run multiple applications simultaneously (e.g., Windows, macOS).
- ❖ It is Commonly used in personal computers, laptops, and smartphones.
- ❖ Advantages: Simple, easy to use, lower resource requirement.
- ❖ Limitation: Cannot support multiple users simultaneously.

ii. Multi-user operating system

- ❖ It Allows multiple users to access the system simultaneously.
- ❖ Each user gets a separate environment and resources are managed efficiently.
- ❖ It Requires strong hardware and proper resource allocation (CPU scheduling, memory management).

- Advantages: Maximizes resource utilization, supports collaborative work.
- Limitation: More complex and requires powerful systems.

Language processors/translator

- A translator or programming language processor is a computer program that converts the programming instructions written in human convenient form into machine language codes that the computers understand and process.
- The program which is used is written by high level programming language is called source code
- Translator Translates source code into machine code

Types of language translator

i. Assembler
ii. Compiler
iii. Interpreter

i. Assembler

- Assembler Converts **Assembly Language** (low-level) into **Machine Code (binary)**.
- Input is Assembly code
- Output is Machine code
- It makes One-to-one translation code.
- Very fast execution of code.
- It makes Efficient use of hardware.
- Assembly language is difficult to learn and not portable.
- Example
- MOV AX, 5 ; Load 5 into AX register
- → Machine code **1011 0000 0000 0101**

ii. Compiler

- A translator that converts the entire source code (high-level language like C, Java) into machine code before execution is called compiler.
- It Translates whole program at once.

- It produces an independent executable file.
- It Shows all errors after compilation.
- Example: C compiler, java compiler

iii. Interpreter

- A translator that converts high-level code into machine code line-by-line and executes immediately is called Interpreter
- It Shows errors immediately and execution stops at error.
- It Does not produce a separate executable file.

(Note: For other topics follow your book)

Unit 2 Process and Process Scheduling

2.1 Introduction to Process and Program

Program

A program is a piece of code which may be a single line or millions of lines. A computer program is usually written by a computer programmer in a programming language. A **computer program** is a collection of instructions that performs a specific task when executed by a computer. When we compare a program with a process, we can conclude that a process is a dynamic instance of a computer program.

Process

The process is an instance of a program in execution. A program becomes process when executable file is loaded in the main memory. **A process** is defined as an entity which represents the basic unit of work to be implemented in the system. Each process has its own address space and process control block (PCB).

	Program	Process
Basic	Program is a set of instruction.	When a program is executed, it is known as process.
Nature	It is Passive in nature.	It is Active in nature.
Lifespan	Its lifespan is Longer.	Its lifespan is Limited.
Required resources	Program is stored on disk in some file and does not require any other resources.	Process holds resources such as CPU, memory address, disk, I/O etc.

2.2 Process Control Block (PCB)

A Process Control Block is a data structure maintained by the Operating System for every process. The PCB is identified by an integer process ID (PID). The PCB is maintained for a process throughout its lifetime, and is deleted once the process terminates. A PCB keeps all the information needed to keep track of a process as listed in the figure:

Process ID: It is unique identification for each of the process in the operating system.

Process State: It is the current state of the process i.e., whether it is ready, running, waiting, or whatever.

Pointer: It is a pointer to parent process.

Priority: It Contains the priority numbers of each process.

Program Counter: Program Counter is a pointer to the address of the next instruction to be executed for this process.

CPU registers: They are Various CPU registers where process need to be stored for execution for running state.

IO status information: This includes a list of I/O devices allocated to the process.

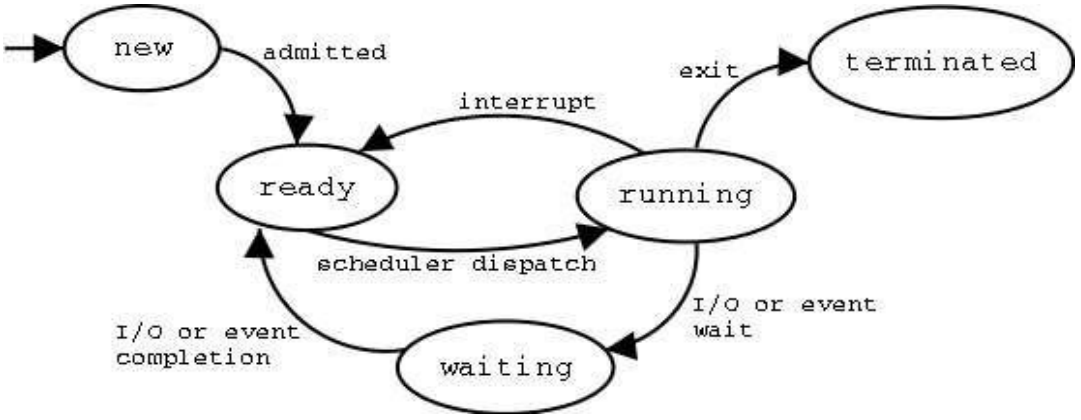
Accounting information: This includes the amount of CPU used for process execution, time limits, execution ID etc.

CPU Scheduling Information: These are process priority and other scheduling information which is required to schedule the process.

Memory management information: This includes the information of page table, memory limits, Segment table depending on memory used by the operating system.

2.3 Process States

When a process executes, it passes through different states. These stages may differ in different operating systems, and the names of these states are also not standardized. In general, a process can have one of the following five states at a time.



i. Start: This is the initial state when a process is first started/created.

ii. Ready: The process is waiting to be assigned to a processor. Ready processes are waiting to have the processor allocated to them by the operating system so that they can run. Process may come into this state after Start state or while running it by but interrupted by the scheduler to assign CPU to some other process.

iii. Running: Once the process has been assigned to a processor by the OS scheduler, the process state is set to running and the processor executes its instructions.

iv. Waiting: Process moves into the waiting state if it needs to wait for a resource, such as waiting for user input, or waiting for a file to become available.

v. Terminated or Exit: Once the process finishes its execution, or it is terminated by the operating system, it is moved to the terminated state where it waits to be removed from main memory.

Operation on the Process:

i. Creation:

It will be ready and come into the ready queue (main memory) and will be ready for execution once the process is created.

ii. Scheduling

Operating system chooses one process and starts executing it, out of the many processes present in the ready queue. Selecting the process which is to be executed next is called scheduling

iii. Execution

Once the process is scheduled for the execution, the processor starts executing it. Process may come to the blocked or wait state during the execution then in that case the processor starts executing the other processes.

iv. Deletion/killing

Once the purpose of the process gets over then the OS will kill the process. The context of the process (PCB) will be deleted and the process gets terminated by the operating system.

2.4 Introduction to 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. Process scheduling allows the operating system to assign each process a time interval for CPU execution. Some of the essential Process Scheduling objectives are

a. Increase the number of interactive users while keeping response times to a minimum.

c. Avoid delaying indefinitely and stick to priorities.

d. It should also mention the procedures that hold the important resources

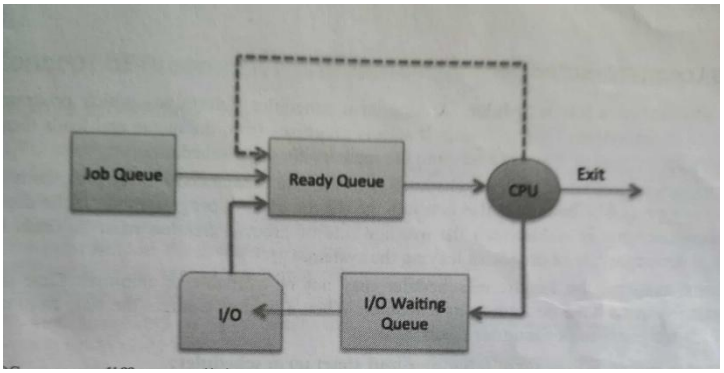
2.5 Process scheduling queues

The OS maintains all Process Control Block (PCB) in Process Scheduling Queues. The OS maintains a separate queue for each of the process states and PCBs of all processes in the same execution state are placed in the same queue. The Operating System maintains the following important process scheduling queues

Job queue: This queue keeps all the processes in the system.

Ready queue: This queue keeps a set of all processes residing in main memory, ready and waiting to execute. A new process is always put in this queue.

Device queues: The processes which are blocked due to unavailability of an I/O device constitute this queue.



Types of Process schedular

i. Short-term schedular

ii. Medium-term schedular

iii. Long-term schedular

i. Short-term schedular

- It is also known as **CPU schedular.**
- Its objective is to increase the system performance in accordance with the chosen set of criteria.
- It is also known as **dispatchers** as it make the decision of which process to execute next.
- It is faster than long.

ii. Medium-term schedular

- It is also called **swapping** schedular.
- It manages process by temporarily moving from between main memory and secondary memory to optimize resource utilization.
- It moves inactive or low- priority processes from main memory to secondary storage to free up space for other tasks.
- It reduces the number of processes in main memory to prevent overcrowding, ensuring proper CPU utilization.

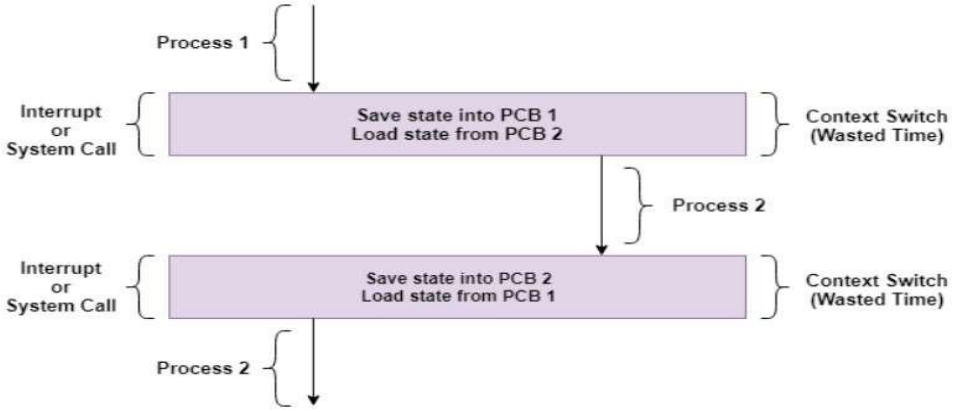
iii. Long- term schedular

- It is also known as **job schedular.**
- it determines which process should be loaded into the main memory from job pool (secondary storage), allowing system to manage the process effectively.
- it Controls the degree of multiprogramming (i.e., how many processes are in memory at once).
- It helps to manage system loaded by balancing **CPU-bound** and I/O **bound processes**.

Context Switching

A context switch is a procedure that a computer's CPU (central processing unit) follows to change from one task (or process) to another while ensuring that the tasks do not conflict. Context Switching involves storing the context or state of a process so that it can be reloaded when required and execution can be resumed from the same point as earlier. This is a feature of a multitasking operating system and allows a single CPU to be shared by multiple processes.

A diagram that demonstrates context switching is as follows:



In the above diagram, Process 1 is running. Process 1 is switched out and Process 2 is switched in because of an interrupt or a system call. Context switching involves saving the state of Process 1 into PCB1 and loading the state of process 2 from PCB2. After some time again a context switch occurs and Process 2 is switched out and Process 1 is switched in again. This involves saving the state of Process 2 into PCB2 and loading the state of process 1 from PCB1.

2.6 Concept of Preemptive and Non-Preemptive scheduling

Preemptive Scheduling

- Preemptive scheduling is used when a process switches from running state to ready state or from waiting state to ready state.
- The resources are allocated to the process for the limited amount of time and then is taken away, and the process is again placed back in the ready queue if that process still has CPU burst time remaining. That process stays in ready queue till it gets next chance to execute.

- Algorithms based on preemptive scheduling are: Round Robin (RR), Shortest Job First (SJF preemptive version) and Priority (preemptive version), etc.

Non-Preemptive Scheduling

- Non-preemptive Scheduling is used when a process terminates, or a process switches from running to waiting state.
- In this scheduling, once the resources (CPU) is allocated to a process, the process holds the CPU till it gets terminated or it reaches a waiting state.
- In case of non-preemptive scheduling it does not interrupt a process running CPU in middle of the execution. Instead, it waits till the process complete its CPU burst time and then it can allocate the CPU to another process.
- Algorithms based on preemptive scheduling are: First come first serve (FCFS), Priority (non-preemptive version), SJF (non- preemptive version).

2.7 Concepts of Threads

In computer programming and operating systems, a thread is the smallest unit of execution within a process. A thread is often referred to as a lightweight process because it shares the same memory space as the process it belongs to and can execute independently. Threads within a process can perform tasks concurrently, which can lead to improved performance and responsiveness in multithreaded applications. Eg: a web browser running, loading images, and playing audio simultaneously.

2.8 Scheduling Algorithms

Scheduling

- 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

Scheduling Algorithms/Techniques

Terminologies

- **Arrival time (AT):** Time at which the process arrives at ready queue
- **Burst Time (BT):** Time required by a process for CPU execution.
- **Completion Time (CT):** Time at which process completes its execution.
- **Turn Around Time (TAT):** Time Difference between completion time and arrival time. i.e.
  TAT= CT – AT
- **Waiting Time (WT):** Time Difference between turnaround time and burst time. i.e. **WT = TAT – BT**

What is Scheduling Algorithm then?

Ans: **The rule or set of rules that is followed to schedule the process is known as process scheduling algorithm.**

There are different Scheduling Algorithm

i. FCFS

ii. SJF

iii. Priority

iv. SRT

v. RR

vi. Multi-level Queue

i. FCFS

- It is a non-preemptive scheduling algorithm.
- The process which arrives first, gets executed first or the process which requests the CPU first, gets the CPU allocated first.
- First Come First Serve, is just like FIFO (First in First out) Queue data structure, where the data element which is added to the queue first, is the one who leaves the queue first.

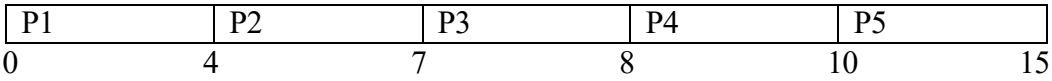
A perfect real-life example of FCFS scheduling is buying tickets

Example: Consider the following set of processes having their arrival time and CPU-burst time. Find the average waiting time and turnaround time using FCFS.

Process	Arrival Time (ms)	Burst Time (ms)
P1	0	4
P2	1	3
P3	2	1
P4	3	2
P5	4	5

Solution:

Gantt Chart



We know,

TAT (Turnaround Time) = CT (Completion Time) – AT (Arrival Time) and WT (Waiting Time) = TAT (Turnaround Time) – BT (Burst Time)

Now finding TAT and WT,

PROCESSES	ARRIVAL TIME	BRUST TIME
A	0	3
B	2	6
C	4	4
D	6	5
E	8	2
F	3	4

Process	AT	BT	CT	TAT	WT
P1	0	4	4	4	0
P2	1	3	7	6	3
P3	2	1	8	6	5
P4	3	2	10	7	5
P5	4	5	15	11	6
				∑TAT= 34	**∑WT= 18**

Average Turnaround time (ATAT)

= **∑TAT/N**

= 34/ 5

= **6.8 ms**

Average Waiting Time (AWT)

= **∑WT/ N**

= 18/5 = **3.6 m**

Classwork

Consider the given processes with arrival and burst time given below. Schedule the given processes using FCFS and find average waiting time and average turn-around time.

ii. Shortest Job First (SJF) Scheduling

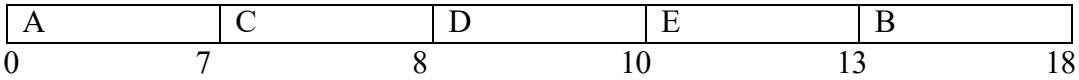
- It is a non-preemptive scheduling algorithm.
- It is also known as shortest job next (SJN)
- It is a scheduling policy that selects the waiting process with the smallest execution time to execute next.

Example 1: Consider the given processes with arrival and burst time given below. Schedule the given processes using SJF and find average waiting time (AWT) and average turnaround time (ATAT)

PROCESSES	ARRIVAL TIME	BRUST TIME
A	0	7
B	2	5
C	3	1
D	4	2
E	5	3

Solution:

According to the given question the Gantt chart for SJF given processes becomes:



Now,

Calculating of TAT and WT using: **TAT= CT – AT and WT = TAT - BT**

Process	Arrival time	Burst time	Completion time	Turn-around time	Waiting time
A	0	7	7	7	0
B	2	5	18	16	11
C	3	1	8	5	4
D	4	2	10	6	4
E	5	3	13	8	5
				∑TAT=42	∑WT=24

Average turn-around time = ∑(TAT) / n= 42/5 = 10.83

Average waiting time = ∑(WT) / n =24/5 = 4.8

iii. Shortest Remaining Time (SRT) Scheduling

- It is preemptive scheduling algorithm.
- It is also called Preemptive shortest job first or Shortest remaining time next or Shortest remaining time first
- It is impossible to implement an interactive system where required CPU time is not known.

iv. Priority Scheduling

- In this scheduling Priority is assigned for each process.
- Process with highest priority is executed first and so on.
- Processes with same priority are executed in FCFS manner.
- Priority can be decided based on memory requirements, time requirements or any other resource requirement.

Priority Scheduling is divided into 2 types.
i. Preemptive priority scheduling
ii. Non-Preemptive priority scheduling

i. Preemptive priority scheduling

- Sometimes it is important to execute higher priority tasks immediately even when a task is currently being executed.
- For example, when a phone call is received, the CPU is immediately assigned to this task even if some other application is currently being used. This is because the incoming phone call has a higher priority than other tasks. This is a perfect example of priority preemptive scheduling.
- If a task with higher priority than the current task being executed arrives then the control of the CPU is taken from the current task and given to the higher priority task.

ii. Non-Preemptive priority scheduling

- In the Non-Preemptive Priority scheduling, The Processes are scheduled according to the priority number assigned to them.
- Once the process gets scheduled, it will run till the completion.
- Unlike priority preemptive scheduling, even if a task with higher priority does arrive, it has to wait for the current task to release the CPU before it can be executed.

Round Robin Scheduling

- CPU is assigned to the process 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 scheduling algorithm

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 cannot be set for the processes.

Unit 3 Memory Management

Memory

Memory is the important part of computer system that stores the data and instruction that needs immediate access. In other words, Memory is the component which is used to store data and files for future use. High speed memory that embedded on the processor is known as **main memory**. Other memory devices which provide backup storage for longer period are called **secondary memory or auxiliary memory.**

Types of memory

i. Volatile memory

ii. Non-volatile memory

i. Volatile memory

Computer storage that only maintains its data while the device is powered is known as volatile memory. Volatile means it loses its data when power is off. It is also known as primary memory. Eg: RAM

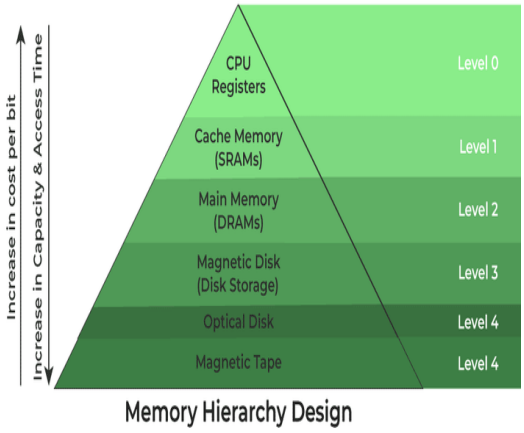
ii. Non-volatile memory

The type of memory in which data is stored permanently i.e. data and information is not lost even after power is off or shutdown is known as secondary memory. Eg: HDD, SSD

Homework

Write down difference between volatile and non-volatile memory

3.1 Memory hierarchy



Memory Hierarchy Design

Memory hierarchy is arranging different kinds of storage present on a computing device based on speed of access.

i. Registers

Registers are small, high-speed memory units located in the CPU. They are used to store the most frequently used data and instructions. Registers have the fastest access time and the smallest storage capacity, typically ranging from 16 to 64 bits.

ii. Cache Memory

Cache memory is a small, fast memory unit located close to the CPU. It stores frequently used data and instructions that have been accessed from the main memory. Cache memory is designed to minimize the time it takes to access data by providing the CPU with quick access to frequently used data.

iii. Main Memory

Main memory, also known as RAM (Random Access Memory), is the primary memory of a computer system. It has a larger storage capacity than cache memory, but it is slower. **Main memory** is used to store data and instructions that are currently in use by the CPU.

Types of Main Memory

- **Static RAM:** Static RAM stores the binary information in flip flops and information remains valid until power is supplied. Static RAM has faster access time and is used in implementing cache memory.
- **Dynamic RAM:** It stores the binary information as a charge on the capacitor. It requires refreshing circuitry to maintain the charge on the capacitors after a few milliseconds. It contains more memory cells per unit area as compared to SRAM.

iv. Secondary Storage

Secondary storage, such as hard disk (HDD) and solid-state drive (SSD) is a non-volatile memory unit that has a larger storage capacity than main memory. It is used to store data and instructions that are not currently in use by the CPU. Secondary storage has the slowest access time and is typically the least expensive type of memory in the memory hierarchy.

v. Magnetic Disk

Magnetic Disks are simply circular plates that are fabricated with either metal or a plastic or magnetized material. The Magnetic disks work at a high speed inside the computer and these are frequently used.

vi. Magnetic tape

Magnetic Tape is simply a magnetic recording device that is covered with a plastic film. Magnetic Tape is generally used for the backup of data. In the case of magnetic tape, the access time for a computer is a little slower and therefore, it requires some amount of time for accessing the strip.

Characteristics of memory hierarchy

- As we move from top to bottom in the Hierarchy, the capacity increases.
- As we move from top to bottom in the Hierarchy, the access time increases.
- As we move from bottom to top in the Hierarchy, the cost per bit increases i.e. Internal Memory is costlier than External Memory.

3.2 Memory function

In computing, memory function refers to various operations, and processes involved in managing a computer's memory resources. It includes how data is stored, accessed and manipulated within computer system.

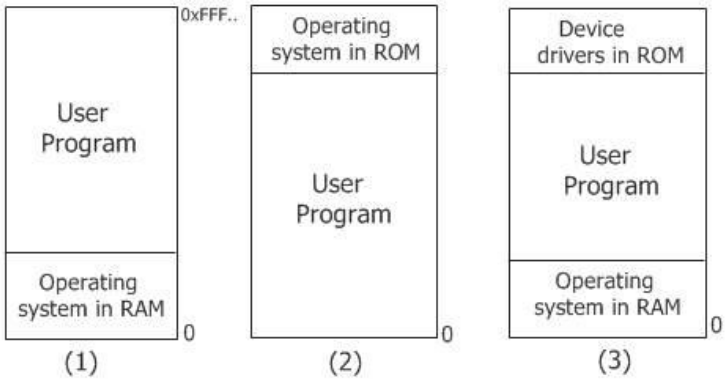
Functions

i. It stores data and instruction.

ii. It stores processing data

iii. It stores the final result of processing.

3.3 Mono-programming model and Multi-programming model



When memory processes only one program at a time is called mono-programming. In mono-programming main memory is divided into parts, i.e. operating system and currently running process.

Memory sharing variations

Fig 1:

OS is placed at the start(bottom) of memory, and the rest of memory is used for user program(the program you want to run).

Real world example:

Notebook = memory

- OS takes the first few pages
- User programs writes in remaining pages.

Fig 2:

OS are not located in Ram anymore. Instead, it, OS is stored in ROM. ROM is like permanent memory that keeps OS ready to use. Entire RAM is available for user program.

Real World Scenario:

- Rome has separate book that remains always same and contain OS.
- User program still uses the entire notebook (RAM) for writing.

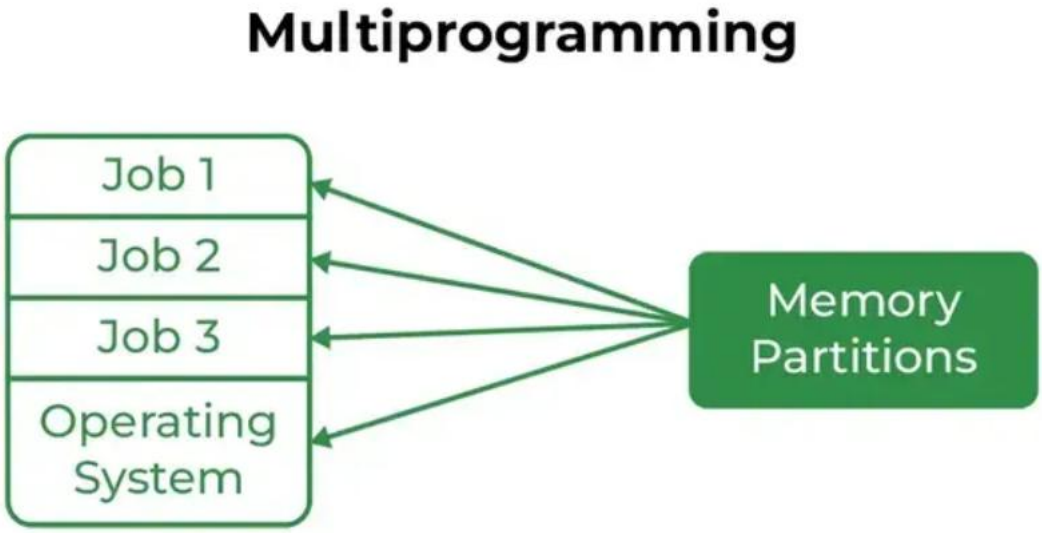
Fig 3:

In this memory variation OS is at end of RAM. Device drivers stored in ROM below it and running space Is available for user program.

Real World Scenario:

- Last few pages reserved for OS.
- Some extra pages near bottom for device drivers.
- Middle pages are for the user program.

Multi-Programing



.

Multi programming is an OS model that allows multiple programs to reside in memory and share CPU, enhancing resource utilization. It operates by executing different parts of programs in rotation, ensuring that while one program waits for I/O operations, the CPU can process another program. Multiprogramming increases CPU utilization.

3.3 Sharing and protection

Sharing and protection is too critical aspect that ensures efficient and secure utilization of computer memory resources.

Sharing

Sharing allows multiple processes to assist the same memory enhancing resource utilization. It enables processor users to access the same memory space or data simultaneously. Sharing enables inter-process communication (IPC). Eg: Code for common functions (like printf) is studying shared libraries to use the same code instead of having their own copies.

Protection

Protection prevents one process does not interfere with another process memory. One process cannot assess or modify another program's memory preventing interference. Eg: If you are running a browser and a word processor memory, protection ensures that the browser doesn't mess up the document you are writing.

3.6 Static and dynamic partition

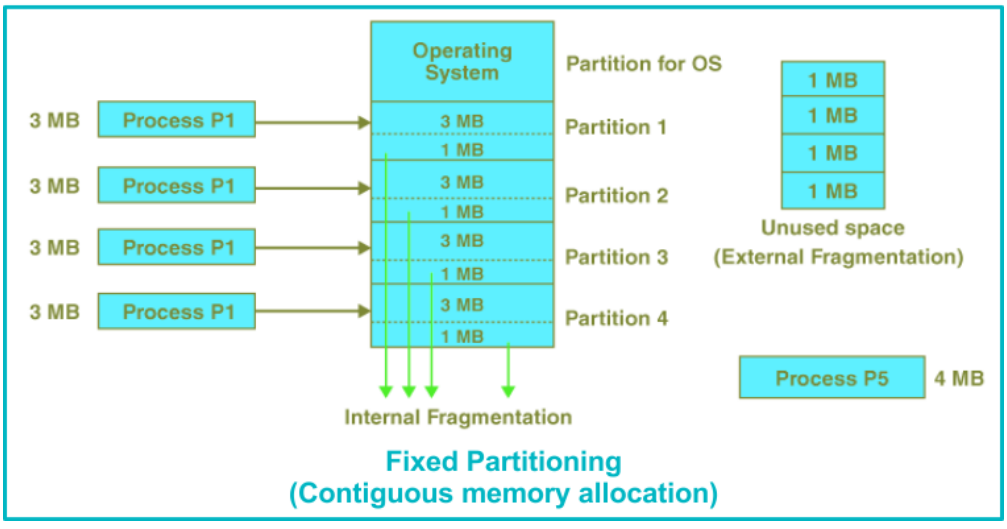
The process of allocating the block of memory to process is called storage allocation. When a process requests for the memory, a single contiguous section of memory block is allocated according to requirement by process.

Partitioning

Partitioning is the process of breaking the main memory into number of blocks. It is the creation of one or more regions on memory so that an operating system can manage information in each region separately.

Static or Fixed partitioning

The simplest way to manage the memory available for process is to partition it into reason with fixed boundaries. In partitioning, the partition may be equal size or unequal size. In this technique, the number of partitions is fixed but size of partition is not fixed. Partition is made before the configuration or before the execution. A process whose size is less than or equal to partition size can be loaded into partition. Main memory with utilization is extremely insufficient as there is occurrence of internal fragmentation.



As illustrated in the figure, the first process occupies only 1MB of the 4MB partition in the main memory. Therefore, the internal fragmentation in the first block is (4-1) MB = 3 MB

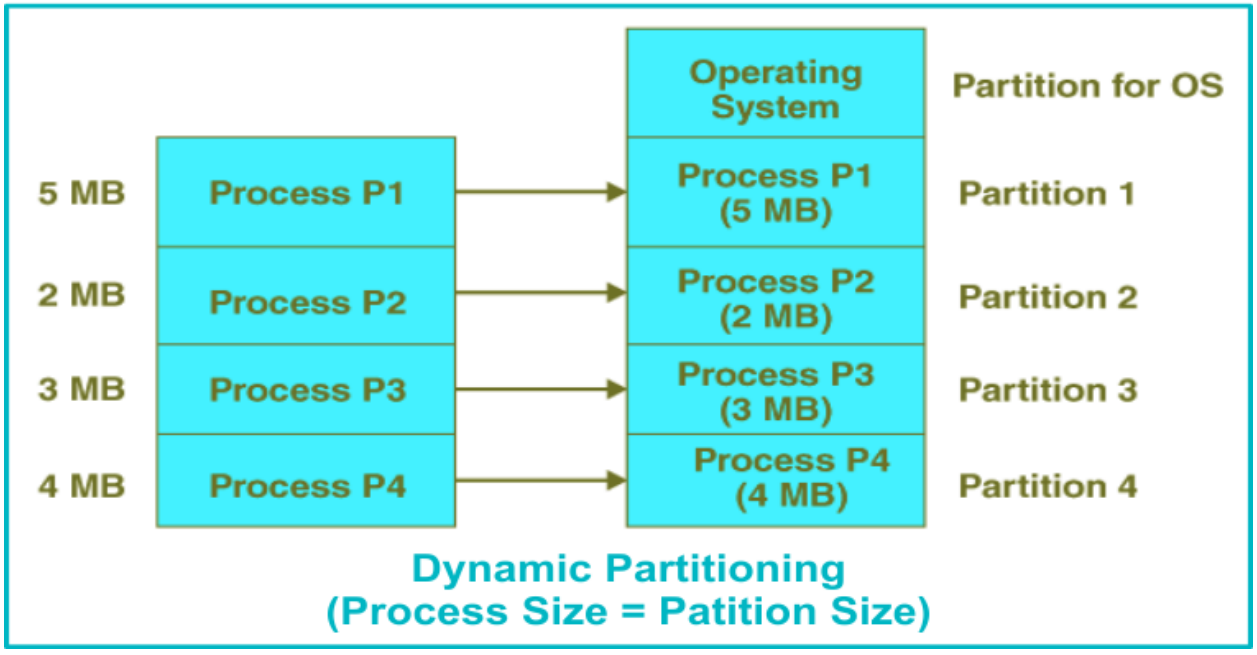
The total internal fragmentation in all blocks = (4-1) MB + (16-14) + (8-7) MB + (8-7) MB = 3 MB +2 MB +1 MB +1 MB = 7MB

Suppose we have a process P5 with a size of 7MB. Even though there is enough free space, this process cannot be accommodated due to the contiguous allocation requirement. As a result, 7 MB of memory is wasted due to external fragmentation.

Dynamic or variable partitioning

Dynamic partitioning plays a pivotal role in improving the problems created by fixed partitioning. In this method, the partition size isn't predetermined but rather established at the moment of process loading.

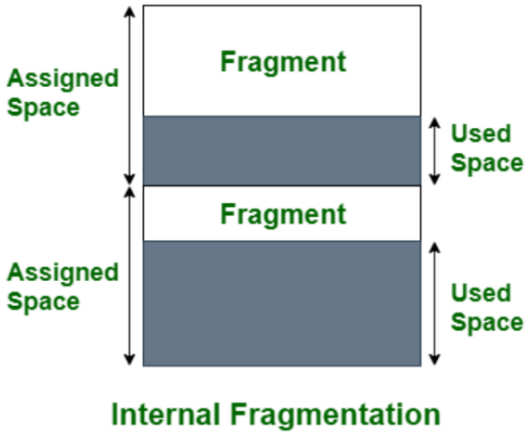
The first partition must be reserved for the Operating System. The remaining space is then subdivided into various parts. Each partition's size equals the size of the process. The partition size adjusts according to the process's needs, thereby avoiding internal fragmentation effectively.



3.7 Internal and External fragmentation

Internal fragmentation

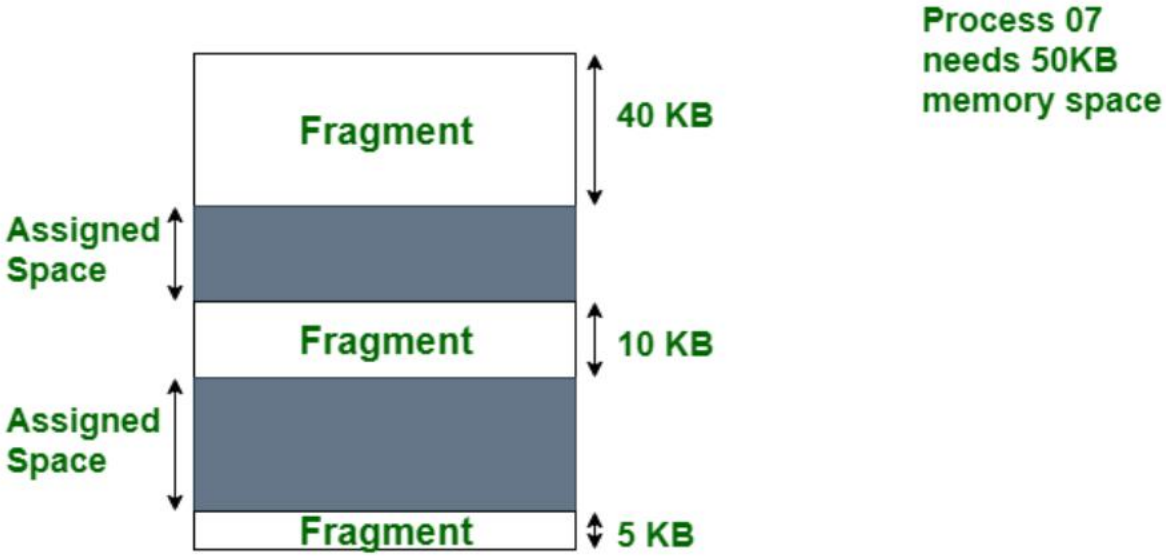
Internal fragmentation arises when a memory block allocated to a process is larger than the process itself, leaving unused space within the memory block. This space can't be used by other processes, leading to wastage of memory resources.



Internal Fragmentation

External fragmentation

External fragmentation when there is enough memory to fit process, but the size is not contagious. It exists between the block of allocated memory and internal fragmentation that has occurred within the allocated block of memory.



In the above diagram, we can see that there is enough space (55 KB) to run a process-07 (required 50 KB) but the memory (fragment) is not contiguous. Here, we use compaction, paging, or segmentation to use the free space to run a process.

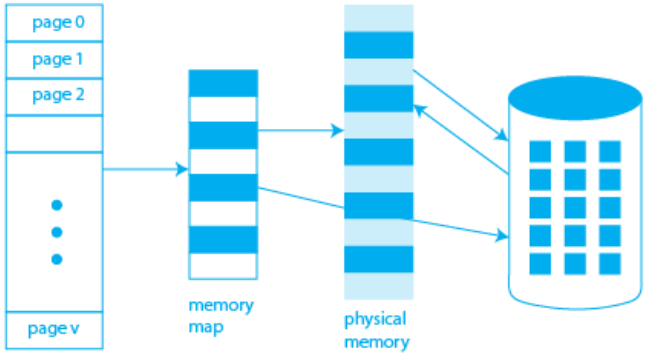
Homework

Write a difference between internal and external fragmentation

Virtual memory and paging

Virtual memory

Virtual memory is a memory management technique that allows the computer to use secondary memory(disks) as an extension of main memory. It enables programs to run even when the required memory is larger than the available RAM.



In the given figure, virtual memory is divided into pages (page 1, page 2, …… page n). These Pages represent the logical memory seen by the program.

A memory map (page table) is used to store the mapping between the virtual pages and physical memory (RAM). It keeps the track of :

- Which pages are present in RAM
- Which pages are stored on the disk

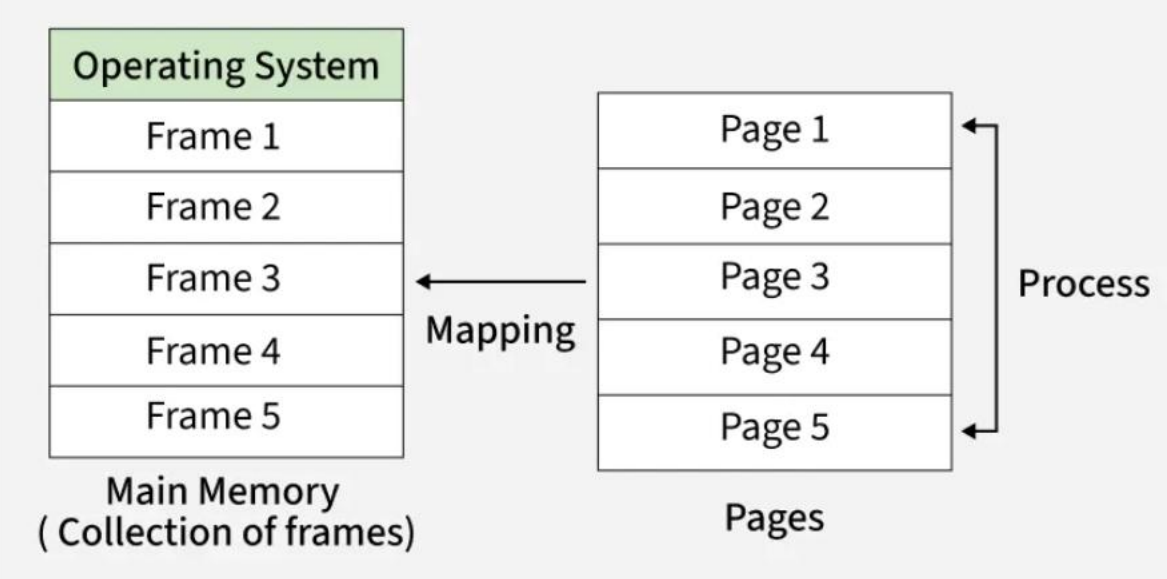
The physical memory (RAM) is divided into frames, and pages are loaded into these frames. The pages are not stored in order but placed wherever space is available.

If required page in not found in RAM, it is fetched from disk (secondary memory). This process is called page swapping.

Hence, virtual memory improves memory utilization and allows execution of large program efficiently.

Paging

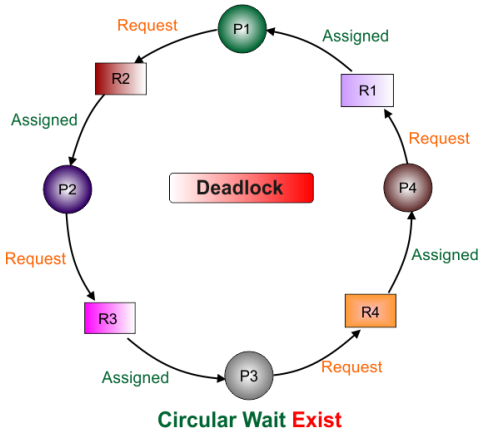
Paging is a storage mechanism used in OS to retrieve processes from secondary storage to the main memory as pages. The primary concept behind paging is to break each process into individual pages. Thus the primary memory would also be separated into frames. The chunks of a process are called pages whereas the chunks of memory are called frames. The basic idea behind paging is that to keep only that part of process in memory which are actually being used.



Unit 4 Deadlock Management

4.1 Introduction to deadlock

Deadlock is a condition where a set of processes are blocked because each process is holding a resource and waiting for another resource acquired by some other process. In other words, deadlock is a situation where several processes in CPU compete for finite number of resources available within the CPU. Each process holds a resource and waits to acquire a resource that is held by another process. All the process waits for resources in circular fashion. Resources may be CPU, memory, disk drives, printer and so on.



4.2 Necessary condition for deadlock

There are four necessary conditions for the occurrence of deadlock.

i) Mutual exclusion

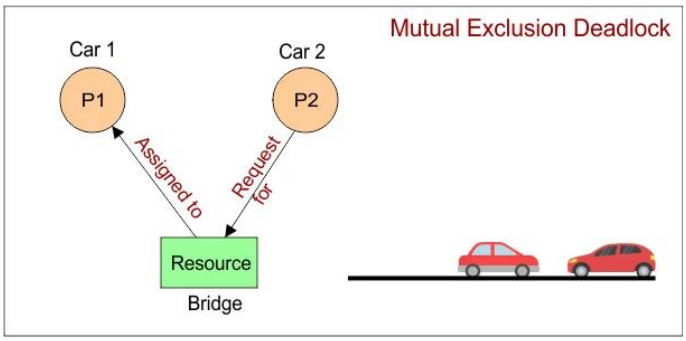
ii) Hold and wait

iii) No preemption

iv) Circular Wait

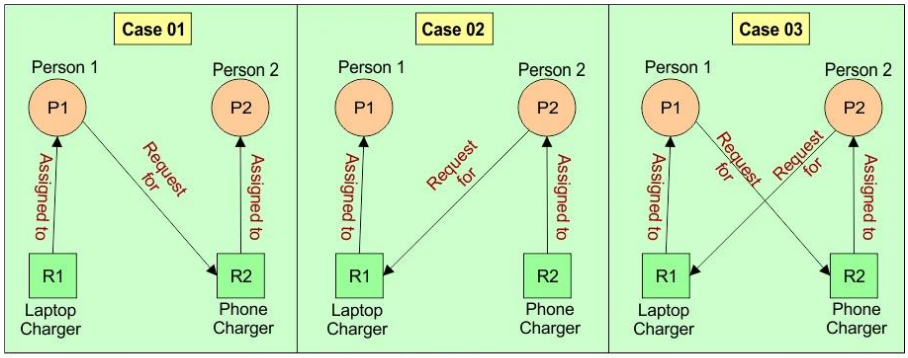
i) Mutual exclusion

Mutual exclusion is a situation in which at least one resource is held in non-sharable mode that is only one process at a time can use resource. If another process requests that resource which is acquired by another process, the requesting process must be delayed until the resource has been released. Eg: Printing a word.



ii) Hold and wait

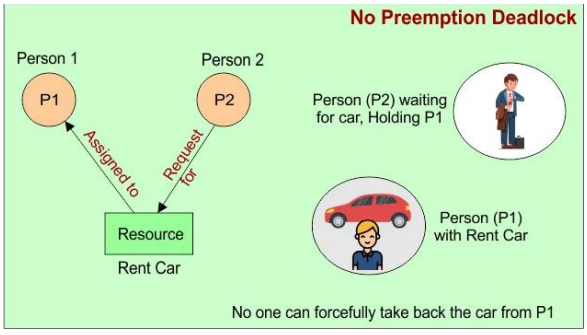
There must exist a process that is holding at least one resource and waiting to acquire additional resources that are currently being held by another resources.



Hold and Wait Deadlock (Exist in all 3 Cases)

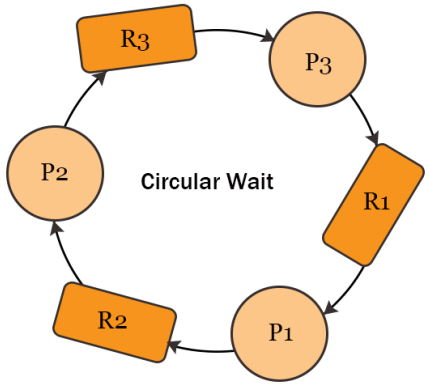
iii) No preemption

A resource cannot be taken from a process unless the process releases the resource. For Resolving the deadlock for man one can simply cancel one of the processes for other to continue. But operating system doesn't do so. It allocates the processor for as much time needed until the task is completed. Hence, there is no temporary reallocation of the resources i.e. no preemption.



iv) Circular wait

Circular Wait occurs when a group of processes are each **waiting for a resource** that is **held by the next process** in the chain, eventually forming a **closed loop**. This circular chain of waiting prevents all processes from proceeding, causing a **deadlock**.



P1 is waiting for P2 to release R2 , P2 is waiting for P3 to release R3 and P3 is waiting for P1 to release R1

4.3 Methods for handling deadlock

Deadlock occurs when a set of processes are stuck, each waiting for a resources held by another process. Key methods to handle deadlock process is explained below

i) Deadlock Prevention

ii) Deadlock avoidance

iii) Deadlock detection

iv) Recovery from deadlock

i) Deadlock Prevention

Deadlock prevention involves breaking at least one of the necessary conditions for deadlock that are mutually exclusive, hold and wait, no preemption and circular.

ii) Deadlock avoidance

we cannot always assume that a process will ask for all its required resources at once. The system must decide whether granting a resource is safe or not.

Safe state

A state is safe if the system can allocate all resources requested by all processes with entering a deadlock

Banker's algorithm

It is the algorithm which checks whether the system or process is in safe state or not. It allocates the resource when process are in safe state.

iii) Deadlock detection

For a system with one instance of each resource, we can detect deadlock by constructing Resource Allocation Graph (RAG).

Example of RAG with cycle

Processes: P_1, P_2, P_3

Resources: R_1, R_2, R_3

Each resource type has only one instance

1. P_1 holds R_1, and Requests R_2

Allocation edge: $R_1 \rightarrow P_1$

Request edge: $P_1 \rightarrow R_2$

2. P_2 holds R_2 and Requests R_3

Allocation edge: $R_2 \rightarrow P_2$

Request edge: $P_2 \rightarrow R_3$

3. P_3 holds R_3 and requests R_1

Allocation edge: $R_3 \rightarrow P_3$

Request edge: $P_3 \rightarrow R_1$

Resulting:

$P_1 \rightarrow R_2$, $R_2 \rightarrow P_2$, $P_2 \rightarrow R_3$, $R_3 \rightarrow P_3$, $P_3 \rightarrow R_1$, $R_1 \rightarrow P_1$

$P_1 \rightarrow P_2 \rightarrow P_3 \rightarrow P_1$ which forms a cycle, so deadlock occurs.

iv) Recovery from deadlock

Traditional operating systems such as windows don't deal with deadlock recovery as it is time and space consuming process. Real-time operating systems use deadlock recovery.

a) Killing the process

The simplest way to recover from deadlock is to kill one or more process. It involves killing the deadlocked process at a time until the deadlock cycle is eliminated.

b) Resource Preemption

Resources are preempted from the process involved in deadlock. Preempted resources are allocated to other processes so that there is a possibility of recovering the system from deadlock.

CHAPTER 5 File management

5.1 Introduction to File Management

File

- A file is sequence of logical records i.e. a sequence of bits and bytes.
- 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 file's creator and user.

File system

- A file system is the methods and data structures that an operating system uses to keep track of files on a disk or partition; that is, the way the files are organized on the disk.
- The word is also used to refer to a partition or disk that is used to store the files or the type of the file system.

File Attributes

- **Name –**only information kept in human-readable form.
- **Identifier**–unique tag (number) identifies file within file system
- **Type**–needed for systems that support different types
- **Location**–This information is a pointer to a device and to the location of the file on that device
- **Size**–current file size and possibly the maximum allowed size
- **Protection**–Access-control information determines who can do reading, writing, executing
- **Time, date, and user identification** –data for protection, security, and usage monitoring etc.

5.2 File Naming

- When a process creates a file, it gives the file a name. When the process terminates, the file continues to exist and can be accessed by other processes using its name.
- File naming = File name + File extension
- Eg. Example.doc In this example the Example is the file name and doc is the file extension.
- Some common file extensions are: DOC and DOCX, .HTML and .HTM,.ODT,.PDF,.XLS and XLSX, .ODS, .PPT and .PPTX,.TXT etc

5.3 File Operation

CREATE: A blank file is created.

DELETE: The purpose of this system call is to delete this file

OPEN: Open a file either for reading or writing.

CLOSE: When a file is no longer accessed.

READ: When a file is only to be read.

WRITE: To write some data on file.

APPEND: To add some data to the end of the file.

RENAME: To rename the file.

5.4 File Extension

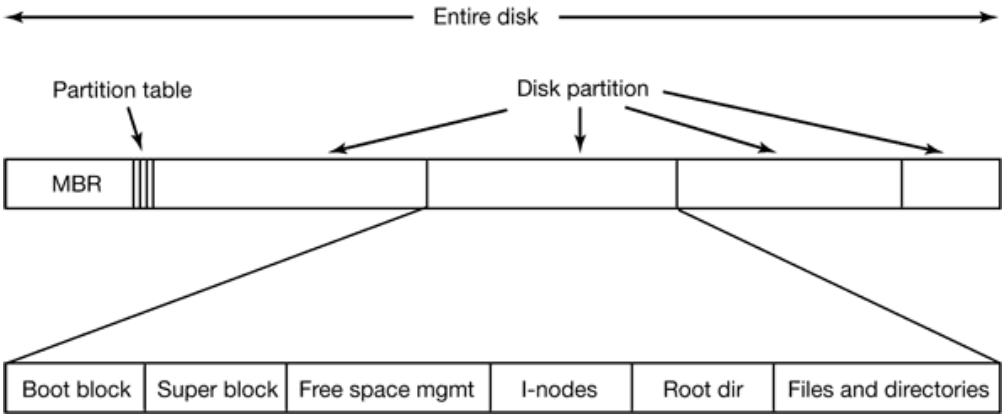
- A file extension is a suffix at the end of a filename, starting with a period, that identifies the file's format.
- It serves as a set of instructions for your operating system (like Windows or macOS) to understand what kind of data is in the file and which program should be used to open it

Common File Extension Categories

Category	Examples	Description
Documents	.docx , .pdf , .txt	Used for text-based files and universal document sharing.
Images	.jpg , .png , .gif	Formats for photos, graphics, and simple animations.
Audio/Video	.mp3 , .mp4 , .wav	Digital media formats for sound and moving pictures.
Spreadsheets	.xlsx , .csv	Data-heavy files used in programs like Microsoft Excel.
Executables	.exe , .apk , .bat	Files that run programs or scripts on a computer or device.
Compressed	.zip , .rar , .7z	Archives used to reduce file size or group multiple files.

5.5 File system Layout

- File system are stored on disks. The disks are separated into partitions.
- The sector 0 is Master Boot Record (MBR), used to boot the computer.
- BIOS reads in and executes MBR, MBR locates the active partition, reads in the boot block and execute.
- The boot block reads in the OS contained in the partition. The superblock contains key parameters about a file system.



1. Boot Block

- Think of this like the **library's front door**. It's the first place the computer looks when it starts up. It contains the information necessary to load the operating system, just like how a librarian uses a key to open the library and let people in.
- When you turn on your computer, it reads the boot block to know where everything is stored and how to load everything up.

2. Superblock

- This is like the **library's map**. It contains important information about the file system itself, such as the size of the library, how many bookshelves (blocks) there are, and how to organize them.
- It keeps track of the whole layout and ensures everything in the library is working properly, like where free space is and where files are stored.

3. Free Space Management

- Imagine a **list of available bookshelves** in the library. When you add a new book (file), you need an empty shelf (space). Free space management keeps track of which shelves are empty (free space) and which are full.
- It ensures that files don't get mixed up, and there's always a place to store new files.

4. Inodes

- An **inode** is like a **book's catalog card**. It doesn't store the actual content of the book (file), but it keeps information about the book like:
    - Who owns the book (file)
    - When the book was added to the library
    - Where the book is located on the shelves
    - The size of the book

an inode helps the system find the file and know what it is without having to open it.

5. Root Directory

- This is like the **main entrance** to the library. It's the first folder you see when you open your file system. Inside the root directory are all the **other sections or folders** in the library.
- For example, your "Documents" folder, "Pictures" folder, and "Music" folder would all be inside the root directory.

6. Files and Directories

- **Files** are like the **books** in the library. These contain information, like your essays, pictures, music, etc.
- **Directories** (also called **folders**) are like **sections or shelves** where the books (files) are organized. Just like how books are grouped into categories like Fiction, History, or Science, your files are organized into folders like Documents, Photos, etc.
-

NOTE

Conclusion

When you turn on your computer, the **boot block** helps start everything up. The **superblock** then tells the system how the file system is organized. The **inode** helps track each file, and when you want to open a file, the system looks up the inode to find out where it's stored. The **root directory** is the main place where everything starts, and inside it are different **directories** (folders) that keep everything organized. The system uses **free space management** to ensure there's always space for new files.

5.6 File Allocation:

File allocation defines how the files are stored in the disk blocks. There are three are three main types of file allocation management.

i.Contiguous Allocation

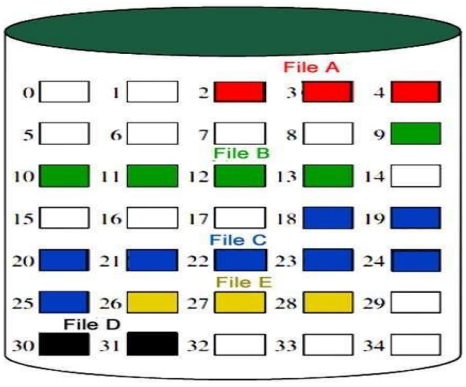
In this scheme, each file occupies a contiguous set of blocks on the disk. For example, if a file requires n blocks and is given a block b as the starting location, then the blocks assigned to the file will be: *b, b+1, b+2,……b+n-1.* This means that given the starting block address and the length of the file (in terms of blocks required), we can determine the blocks occupied by the file.

For example, If we have a disk with 1 KB blocks, then for storing a 50 KB file, we will require 50 consecutive blocks.

• Address of starting block

• Length of the allocated portion.

The file allocation table in contiguous allocation contains



File allocation table

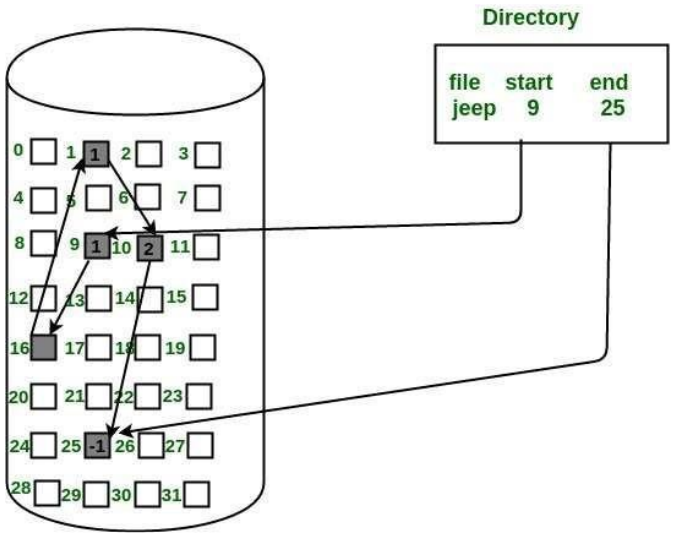
File name	Start block	Length
File A	2	3
File B	9	5
File C	18	8
File D	30	2
File E	26	3

The *file 'File A'* in the above figure starts from the block 2 with length = 3 blocks. Therefore, it occupies *2,3 and 4* blocks.

ii. Linked Allocation

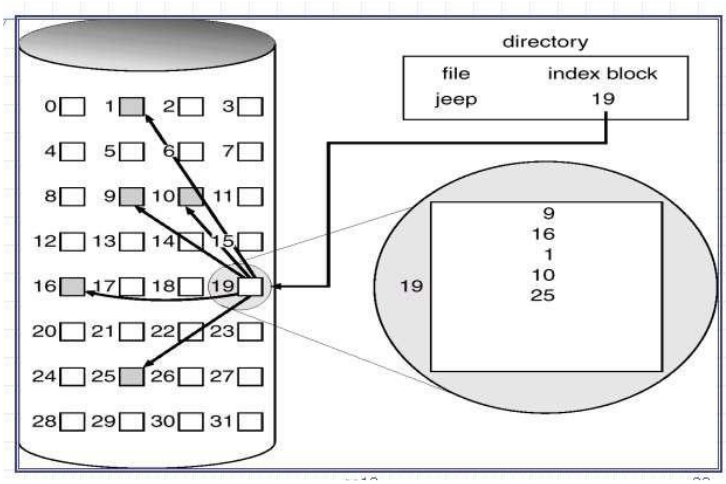
In this scheme, each file is a linked list of disk blocks which **need not be** contiguous. The disk blocks can be scattered anywhere on the disk. The file allocation table contains a pointer to the starting and the ending file block. Each block contains a pointer to the next block occupied by the file.

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.



iii. 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 below:



5.7 Free space management

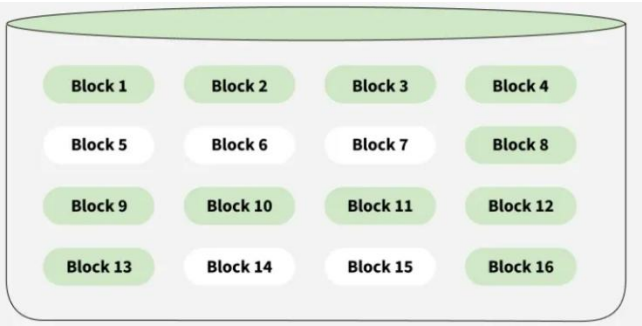
Free space management involves managing the available storage space on the hard disk or other secondary storage devices. To reuse the space released from deleting the files, a free space list is maintained.

The free space list can be maintained as

i. Bit map or Bit vector

In this approach, A Bitmap or Bit Vector is series or collection of bits where each bit corresponds to a disk block. The bit can take two values 0 and 1:

- 0 indicates that the block is free and 1 indicates an allocated block.
- The given instance of disk blocks on the disk in Figure can be represented by a bitmap of 16 bits as: 1111000111111001.



ii. Linked list

Free block is linked together, i.e. a free block contains a pointer to the next free block. The block number of the very first disk block is stored at a separate location on the disk. A drawback of this method is the I/O require for free space list traversal.

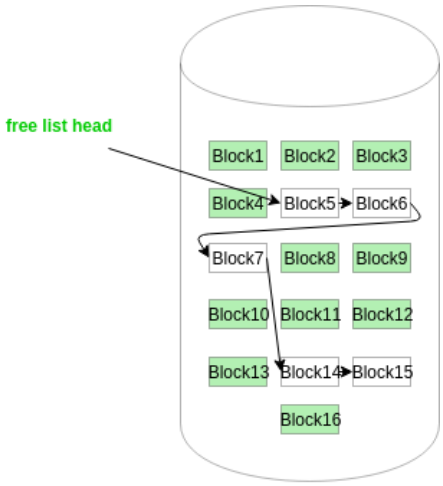


Figure - 2

iii. Grouping

The grouping stores the address of the free blogs in the first free blog.The first free block stores the address of some, say N free blocks. Out of these N blocks the first N-1 blocks are actually free and the last block contains the address of next free blocks.

iv. The counting stores the address of the first free disk block and a number of n of free contagious disk blocks that follow the first block. Every entry in the list contains:

- Address of the first free disk block
- A number n

Unit 6 Linux

6.1 Introduction to Linux

Linux is based on the UNIX operating system. UNIX is a powerful, multi-user, multitasking operating system originally developed in the 1970s at AT&T Bell Labs. It laid the foundation for many modern operating systems, including Linux.

- Linux is free and open-source, accessible to everyone.
- This promotes global collaboration and innovation.
- Linux offers efficient performance and strong security.
- It works well across many devices and industries.

6.2 Features of Linux

i. Free and open source

Linux is completely free of cost, and expenses are never a hindrance to using it as an operating system. Linux is open source. This means that modification of code, analysis of codes, redistribution of codes, or selling copies of the enhanced codes can be done by anyone in the world provided they come under the same license where the license also costs no charge.

ii. Extremely Flexible

Linux has incorporated itself into embedded products like watches, digital equipment and supercomputing servers. There are no prerequisites for installing an entire Linux suit. It allows a user to install only the components that are required by the user.

iii. Lightweight Infrastructure

Linux consumes less storage space, and its installation requires around 4GB to 8GB of disk space. Memory footprint or the amount of memory *(RAM)* used by the software while running is also less and it is compatible with all kinds of file formats like text files, audio files, video files, graphic formats, etc.

iv. Graphical User Interface (GUI)

Even though Linux works on using the command line interface but it can be converted to be used like windows having a Graphical user interface. This is done mostly by installing packages. The most common way of having a GUI on the Linux environment is to log in to Ubuntu server and install its desktop environment.

v. End-to-end encryption

Linux allows end-to-end encryption while accessing data thus storing public keys in the server. All data is password protected and provides authentication to users. It also offers many security features and provides file permissions, a secure shell, etc.

vi. Portable Environment

Linux works in any kind of environment and doesn't depend on the device being high-end or low-end. Many users can simultaneously use it anytime, anyplace, and on multiple devices. It supports all kinds of hardware to work on.

vii. Shell/ Command-line Interface

The Linux command line interpreter is known as *Shell* that provides an interface between the user and kernel which then executes programs known as commands. Hence, Linux uses Command-line interface to carry out the execution of tasks which is comparatively more efficient to execute and takes lesse time. It also takes less space in memory.

viii. Multi-user and Multi-programming

Linux allows multiple users to access the system resources at the same time and allows multiple applications to run at the same time.

6.3 Advantage and disadvantage of Linux

Advantage

i. Highly secure

ii. Stable

iii. Free and open source

iv. Easy to use

v. Absolute freedom over system

vi. High performance

vii. Proper use of resources

viii. Privacy friendly

ix. Easy to install software

x. Better software updates

Disadvantage

i. No standard Edition

ii. Hard learning curve

iii. Limited markets share

iv. Lack of proprietary software

v. Difficult do troubleshoot

vi. Poor support for games

vii. Unsupported hardware

viii. Lack of Technical Support

(Explain above points by yourself)

6.4 Linux Family

A Linux distribution or family is an operating system built on the Linux kernel, bundled with essential software, tools, and package managers, customized to serve different users like developers, enterprises, cybersecurity professionals, and general users.

Some of the papular Linux distribution are given below

i. Ubuntu

Ubuntu is a Linux based operating system and design for computers, smartphones, and network services. The system is developed by UK based company called Canonical Ltd. Ubuntu can be installed directly on hardware or run within a virtual machine. It is widely used for cloud computing, with integration support for platforms such as OpenStack. It is also one of the most popular Linux distributions for general desktop use.

ii. Red Hat Linux Enetrprise

RHEL is a comparative Linux distribution offered by Red Hat Inc. which is an open-source solutions provider company, it is an advanced Linux OS freely offered by Red Hat Inc. It is primarily intended for business applications to meet their niches while providing a highly stable platform to host and execute business-critical applications. Fedora, being an upstream Open Source project, integrates features as part of RHEL while improving the security program, and formal support for all interruptions with assurance of business stability and continuity.

iii. Arch Linux

Arch Linux is an independently developed, open-source Linux distribution known for its simplicity, bleeding-edge updates, and user-focused design. It does not come with pre-installed graphical tools; instead, it provides a clean base system that users can shape exactly the way they want.

Arch promotes a do-it-yourself (DIY) approach, offering:

- Full control over installed packages.
- Clean and minimal installations.
- Continuous updates through a rolling-release model.
- Extensive documentation through the famous Arch Wiki.

iv. Fedora

Fedora Linux is a powerful, open-source, community-driven Linux distribution known for its cutting-edge features, modern design, and strong security. Developed by the Fedora Project and

sponsored by Red Hat, Fedora is widely used for software development, enterprise systems, and general desktop use.

It provides a polished Linux experience and includes tools for:

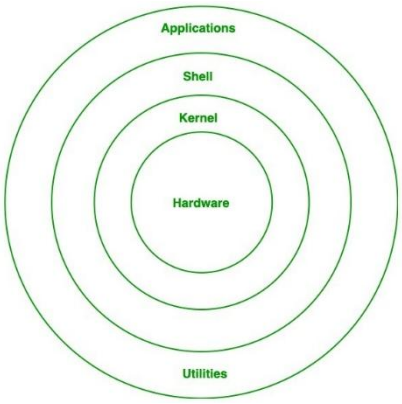
- Software development
- System administration
- Cloud computing
- Containerization
- Desktop productivity

6.5 difference between Windows and Linux

Linux	Windows
1. Some to Linux is an open-source operating system.	1. Windows are not the open-source operating systems.
2. It is free of cost	2. It is costly.
3. Its file name case sensitive	3. Its file name is not case sensitive
4. It uses monolithic kernel	4. It uses micro kernel
5. It is more efficient in comparison to windows.	5. It is less efficient.
6.It uses forward slash for separating the directories	6. It uses back slash for separating the directories.
7. It provides more security than Windows.	7. It provides less security than Linux.
8. It is widely used In hacking purpose based system.	8. It does not provide much efficiency in hacking.

6.6 Structure of Linux

The architecture of Linux consists of different elements which are described below:



i. Kernel

Kernel is the core component of the Linux operating system that sits between the hardware and user space, managing system resources and ensuring smooth communication between software and hardware. It controls how processes are executed, scheduled, and isolated to maintain system stability and security.

The kernel is responsible for:

- **Memory management –** Allocates and manages system memory efficiently
- **Process management –** Schedules processes and controls execution using queues
- **Resource allocation –** Distributes CPU, memory, and I/O resources among processes
- **Device management –** Controls hardware devices through device drivers
- **Application interaction –** Acts as a bridge between applications and hardware
- **Security –** Enforces access control and system-level security mechanisms

ii. System Library

System libraries provide predefined functions that allow application programs and system utilities to access kernel features without interacting with the kernel directly. They form the foundation for software development by offering reusable, standardized interfaces for system operations.

Common system libraries include:

- **GNU C Library (glibc):** Provides core system calls and built-in functions required for executing C programs.
- **libpthread (POSIX Threads):** Enables creation and management of multithreaded applications.
- **libdl (Dynamic Linker):** Supports dynamic loading and linking of shared libraries at runtime.
- **libm (Math Library):** Offers mathematical functions such as trigonometry, logarithms, and exponentiation.
- **Other libraries:** librt (Real-Time operations), libcrypt (Cryptographic functions), libnss (Name Service Switch), libstdc++ (C++ Standard Library).

iii. Shell

The Shell is also software or It can be determined as the interface to the kernel. It takes commands from the user and interprets them. The shell transmits these commands to the kernel, which then performs the requested operations. Users can just enter the commend and using the kernel's function that specific task is performed accordingly.

iv. Hardware layer

The hardware layer is the lowest level of the Linux operating system and forms the foundation on which all other components operate. It consists of physical devices and low-level controls that allow the system to function efficiently and reliably.

- Includes physical components such as CPU, memory, storage, and I/O devices.
- Works with device drivers to enable hardware communication.
- Supports memory access, CPU control, and I/O operations.
- Ensures stable and efficient interaction between hardware and the operating system.

v. System utility

System utilities are command-line tools that help users and administrators manage, configure, and monitor the Linux system. They simplify system administration by providing ready-to-use commands for common tasks.

- Perform file and directory management operations
- Monitor system performance and resource usage
- Manage users, groups, and permissions
- Configure and troubleshoot network settings

6.7 basic Linux commands

Linux commands are used to interact with the operating system through the terminal and perform tasks like file management, navigation, and system monitoring. Learning basic Linux commands helps beginners understand how Linux works and use it efficiently for daily tasks.

- Helps beginners understand and use the Linux terminal effectively
- Covers commonly used commands for files, directories, and system tasks
- Useful for students, developers, and system administrators
- Builds a strong foundation for advanced Linux and server management

Some of the basic Linux commands are described below

1. ls

The ls command in Linux is used to list files and directories present in the current working directory.

- Displays files and folders in a directory.
- Help users quickly understand directory contents.

Syntax:
ls [options] [directory]

2. pwd

The pwd command in Linux is used to display the present working directory.

- Shows the full path of the current directory.
- Helps users know their current location.

Syntax

Pwd

3. mkdir

The mkdir command in Linux is used to create new directories.

- Creates new folders in the file system.
- Helps organize files and directories.

Syntax

mkdir directory_name

4. cd

The cd command in Linux is used to change the current working directory.

- Allows navigation between directories.
- Moves the user to the home directory when used without arguments.

Syntax

cd directory_name

5. rmdir

The rmdir command in Linux is used to delete empty directories.

- Removes directories that do not contain files.
- Helps clean up unused folders.

Syntax

rmdir directory_name