OPERATING SYSTEMS

Unit – I:

Operating system introduction: Operating systems objectives and functions, computer System architecture, os structure, os operations, evolution of operating systems – simple Batch, multi programmed, time shared, parallel, distributed systems, real-time systems, Operating system services.

Unit – II:

Process and cpu scheduling - process concepts - the process, process state, process control Block, threads, process scheduling - scheduling queues, schedulers, context switch,

Preemptive scheduling, dispatcher, scheduling criteria, scheduling algorithms,

case studies: Linux, windows.

Process coordination - process synchronization, the critical section problem, synchronization Hardware, semaphores, and classic problems of synchronization, monitors, case studies: Linux, windows.

Unit –III:

Memory management and virtual memory - logical & physical address space, swapping, Contiguous allocation, paging, structure of page table. Segmentation, segmentation with Paging, virtual memory, demand paging, performance of demanding paging, page

Replacement page replacement algorithms, allocation of frames.

Unit – IV:

File system interface - the concept of a file, access methods, directory structure, file system Mounting, file sharing, protection, file system structure, Mass storage structure - overview of mass storage structure, disk structure, disk Attachment, Disk scheduling.

Unit –V:

Deadlocks - system model, deadlock characterization, methods for handling deadlocks, Deadlock prevention, deadlock avoidance, deadlock detection and recovery from deadlock.

.

References books:

1. Operating system principles, Abraham Silberchatz, peter b. Galvin, Greg Gagne 8th Edition, wiley student edition.

2. Principles of operating systems by Naresh chauhan, oxford universit

UNIT-I

OPERATING SYSTEMS INTRODUCTION

1. What is an Operating System? Explain objectives and functions of Operating System?

An operating system is a program that acts as an intermediary between a user of a computer and the computer hardware.

Operating system can also be defined as Resource allocator – manages and allocates Resources. Control program – controls the execution of user programs and operations of I/O devices.

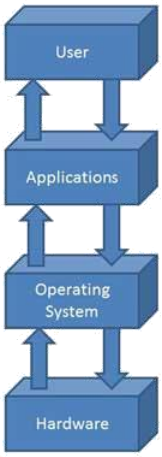
Kernel – the one program running at all times (all else being application programs).

Operating system goals:

i) Execute user programs and make solving user problems easier.

ii) Make the computer system convenient to use.

iii) Use the computer hardware in an efficient manner.



Objectives of Operating System:

The objectives of the operating system are –

- To make the computer system convenient to use in an efficient manner.
- To hide the details of the hardware resources from the users.
- To provide users a convenient interface to use the computer system.
- To act as an intermediary between the hardware and its users, making it easier for the users to access and use other resources.

- To manage the resources of a computer system.
- To keep track of who is using which resource, granting resource requests, and mediating conflicting requests from different programs and users.
- To provide efficient and fair sharing of resources among users and programs.

Functions of Operating System:

Following are some of important functions of an operating System.

- Process Management
- Memory Management
- File Management
- I/O System Management
- Secondary Storage Management
- Security
- Control over system performance
- Job accounting
- Error detecting aids
- Coordination between other software and users

1. Process Management:

A program does nothing unless their instructions are executed by a CPU.A process is a program in execution. A time shared user program such as a complier is a process. A word processing program being run by an individual user on a pc is a process. A system task such as sending output to a printer is also a process. A process needs certain resources including CPU time, memory files & I/O devices to accomplish its task. These resources are either given to the process when it is created or allocated to it while it is running. The OS is responsible for the following activities of process management.

- Creating & deleting both user & system processes.
- Suspending & resuming processes.
- Providing mechanism for process synchronization.
- Providing mechanism for process communication.
- Providing mechanism for deadlock handling.

2. Memory Management:

The main memory is central to the operation of a modern computer system. Main memory is a large array of words or bytes ranging in size from hundreds of thousand to billions. Main memory stores the quickly accessible data shared by the CPU & I/O device. The central processor reads instruction from main memory during instruction fetch cycle & it both reads &writes data from main memory during the data fetch cycle. The main memory is generally the only large storage device that the CPU is able to address & access directly. For example, for the CPU to

process data from disk. Those data must first be transferred to main memory by CPU generated E/O calls. Instruction must be in memory for the CPU to execute them.
The OS is responsible for the following activities in connection with memory management.

- Keeping track of which parts of memory are currently being used & by whom.
- Deciding which processes are to be loaded into memory when memory space becomes available.
- Allocating & deallocating memory space as needed.

3. File Management:

File management is one of the most important components of an OS computer can store information on several different types of physical media magnetic tape, magnetic disk & optical disk are the most common media. Each medium is controlled by a device such as disk drive or tape drive those has unique characteristics. These characteristics include access speed, capacity, data transfer rate & access method (sequential or random).For convenient use of computer system the OS provides a uniform logical view of information storage. The OS abstracts from the physical properties of its storage devices to define a logical storage unit the file. A file is collection of related information defined by its creator. The OS is responsible for the following activities of file management.

- Creating & deleting files.
- Creating & deleting directories.
- Supporting primitives for manipulating files & directories.
- Mapping files into secondary storage.
- Backing up files on non-volatile media.

4. I/O System Management:

One of the purposes of an OS is to hide the peculiarities of specific hardware devices from the user. For example, in UNIX the peculiarities of I/O devices are hidden from the bulk of the OS itself by the I/O subsystem. The I/O subsystem consists of:

- A memory management component that includes buffering, caching & spooling.
- A general device- driver interfaces drivers for specific hardware devices. Only the device driver knows the peculiarities of the specific device to which it is assigned.

5. Secondary Storage Management:

The main purpose of computer system is to execute programs. These programs with the data they access must be in main memory during execution. As the main memory is too small to accommodate all data & programs & because the data that it holds are lost when power is lost. The computer system must provide secondary storage to back-up main memory. Most modern computer systems are disks as the storage medium to store data & program. The operating system is responsible for the following activities of disk management.

- Free space management.
- Storage allocation.
- Disk scheduling

Because secondary storage is used frequently it must be used efficiently.

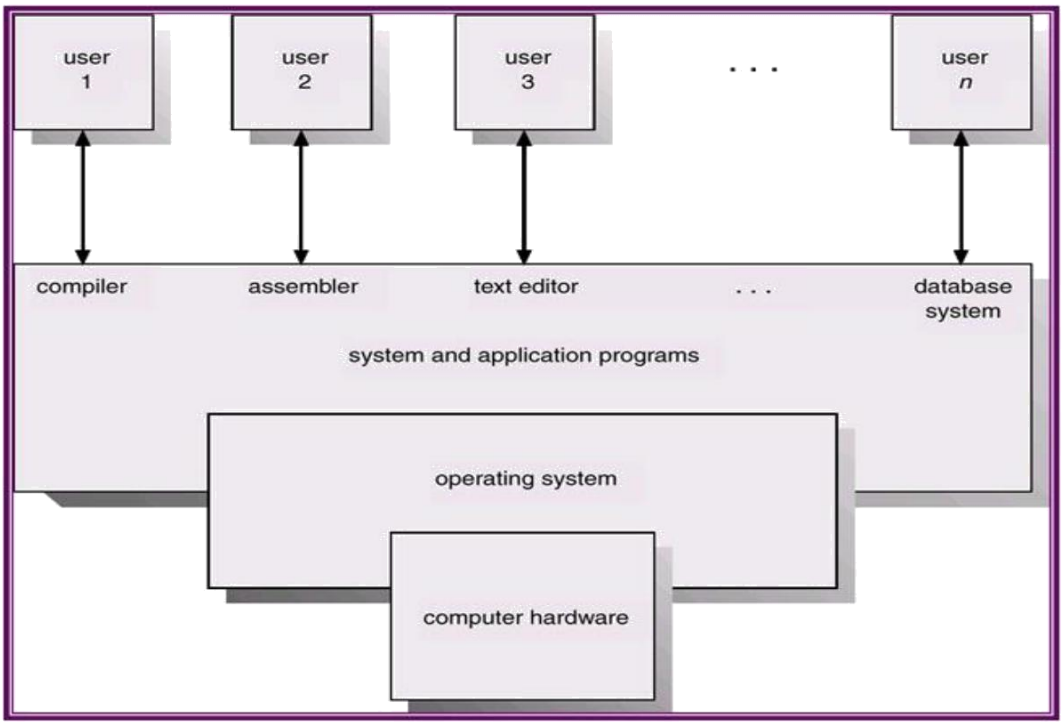
Following are some of the important activities that an Operating System performs –

- **Security** – By means of password and similar other techniques, it prevents unauthorized access to programs and data.
- **Control over system performance** – Recording delays between request for a service and response from the system.
- **Job accounting** – Keeping track of time and resources used by various jobs and users.
- **Error detecting aids** – Production of dumps, traces, error messages, and other debugging and error detecting aids.
- **Coordination between other softwares and users** – Coordination and assignment of compilers, interpreters, assemblers and other software to the various users of the computer systems.

2. Explain Computer System Components?

A computer system can be divided into four components: the hardware, the operating system, the application programs, and the users.

1. **Hardware** – provides basic computing resources (CPU, memory, I/O devices) for the system.
2. **Operating system** – controls and coordinates the use of the hardware among the various application programs for the various users.
3. **Applications programs** – define the ways in which the system resources are used to solve the computing problems of the users (compilers, database systems, video, games, business programs).
4. **Users** (people, machines, other computers).



Two Views of Operating System

1. User's View
2. System View

User View: The user view of the computer refers to the interface being used. Such systems are designed for one user to monopolize its resources, to maximize the work that the user is performing. In these cases, the operating system is designed mostly for ease of use, with some attention paid to performance, and none paid to resource utilization.

System View: Operating system can be viewed as a resource allocator also. A computer system consists of many resources like - hardware and software - that must be managed efficiently. The operating system acts as the manager of the resources, decides between conflicting requests, controls execution of programs etc.

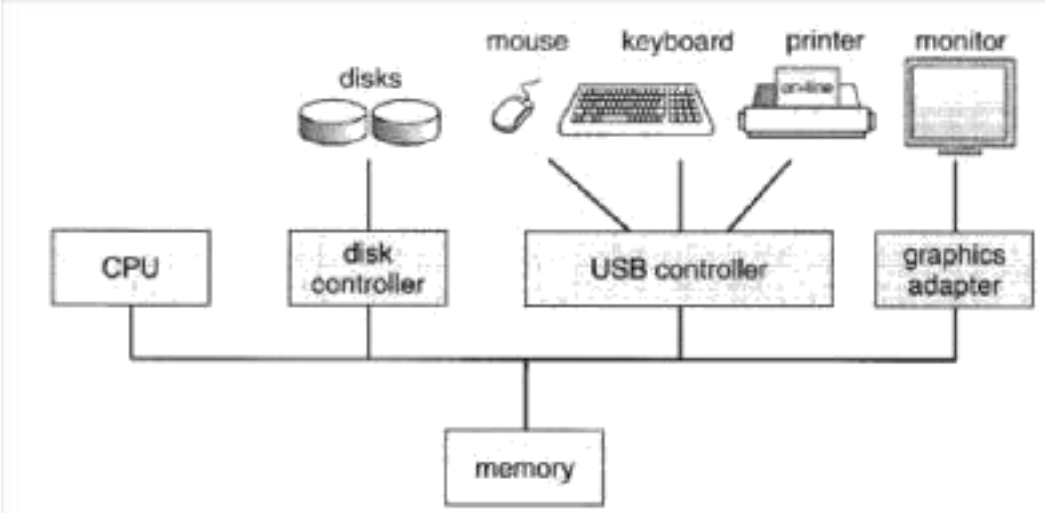
3. Explain about Computer System

Organization? Computer-System Operation

- A Modern general-purpose computer system consists of one or more CPUs and a number of device controllers connected through a common bus that provides access to shared memory.
- Each device controller is in charge of a specific type of device (for example, disk drives, audio devices, and video displays). The CPU and the device controllers can execute concurrently, competing for memory cycles.

❖ For a computer to start running-for instance, when it is powered up or rebooted-it needs to have an initial program to run. This initial program, or **bootstrap program**, tends to be simple. It is stored in read-only memory (ROM) .

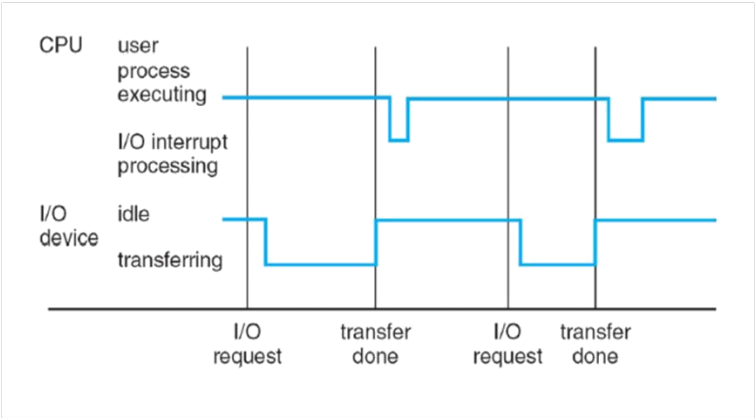
- The bootstrap program must know how to load the operating system and to start executing that system. To accomplish this goal, the bootstrap program must locate and load into memory the operating system kernel. The operating system then starts executing the first process, such as "init," and waits for some event to occur.



A modern computer system.

- The occurrence of an event is usually signaled by an **interrupt** from either the hardware or the software.
- Hardware may trigger an interrupt at any time by sending a signal to the CPU, usually by way of system bus.
- Software may trigger an interrupt by executing a special operation called a system call. When the CPU is interrupted, it stops what it is doing and immediately transfers execution to a fixed location.
- The fixed location usually contains the starting address where the service routine for the interrupt is located. The interrupt service executes; on completion, the CPU resumes the interrupted computation.
  A time line of this operation is shown in figure.

Interrupt Timeline



2. Storage Structure

Computer programs must be in main memory (also called RAM) to be executed. Main memory is the only large storage area that the processor can access directly. It forms an array of memory words. Each word has its own address. Interaction is achieved through a sequence of load or store instructions to specific memory addresses. The

load instruction moves a word from main memory to an internal register within the CPU, whereas the store instruction moves the content of a register to main memory. The **instruction-execution cycle** includes:

1) Fetches an instruction from memory and stores that instruction in the instruction register. And increment the PC register.

2) Decode the instruction and may cause operands to be fetched from memory and stored in some internal register.

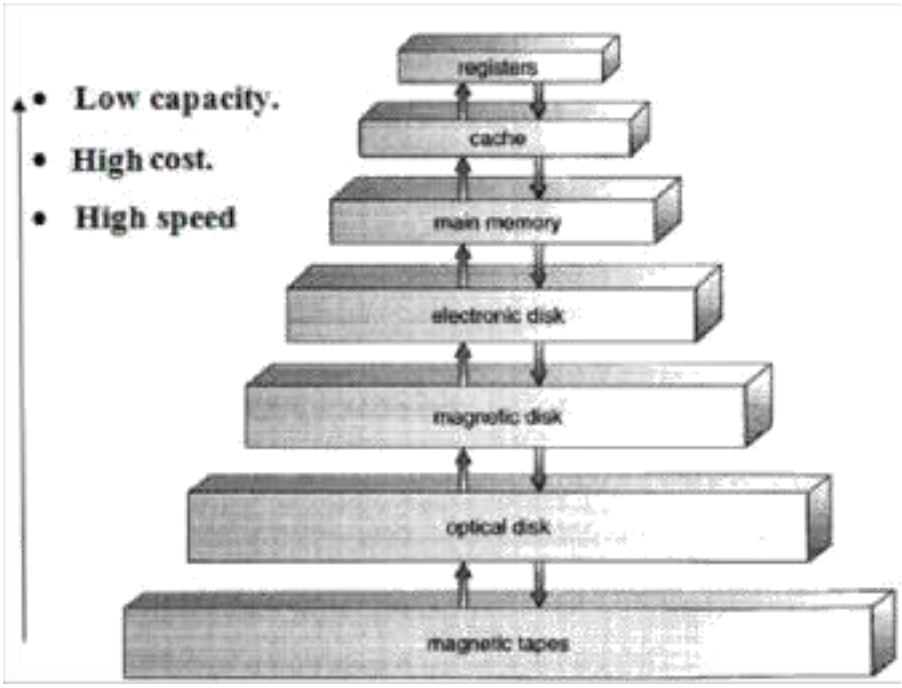
3) Execute the instruction and store the result in memory

The programs and data are not resided in main memory permanently for the following two reasons:

1) Main memory is usually too small to store all needed programs and. Data permanently.

2) Main memory is a volatile storage device that loses its contents when power is turned off or otherwise lost.

The wide variety of storage systems in a computer system can be organized in a hierarchy . **The main differences among the various storage systems lie in speed, cost, size, and volatility**. The higher levels are expensive, but they are fast.



Storage device hierarchy

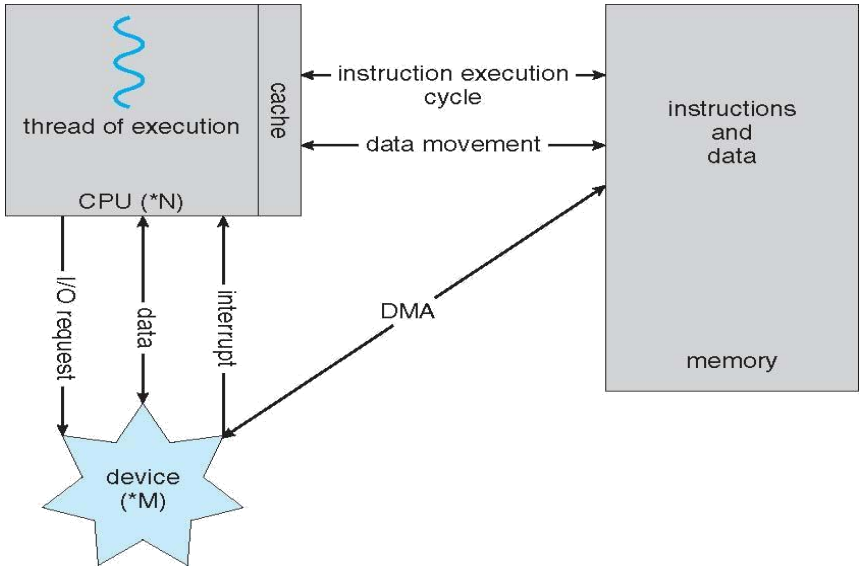
3. I/O Structure

Storage is only one of many types of I/O devices with in a computer. A computer system consists of **CPUs and multiple device controllers** that are connected through a common bus. **The device controller** is responsible for moving the data between the peripheral devices that it controls and its local buffer storage. Typically, operating systems have a device driver for each device controller.

- To start an I/O operation, the device driver loads the appropriate registers within the device controller.
- The device controller examines the contents of these registers to determine what action to take.
- The controller starts the transfer of data from the device to its local buffer. Once the transfer of data is complete, the device controller informs the device driver via an **interrupt** that it has finished its operation.
- The device driver then returns control to the operating system. For other operations, the device driver returns status information.

❖ This form of interrupt-driven I/o is fine for moving small amount of data but can produce high overhead when used for bulk data movement such as disk I/O.For moving bulk data, **direct memory access (DMA)** is used.

❖ After setting up buffers, pointers, and counters for the I/O device, the device controller transfers an entire block of data directly to or from its own buffer storage to memory, with no intervention by the CPU. Only one interrupt is generated per block, to tell the device driver that the operation has completed.



HOW MODERN computer WORKS

4. Explain Computer System Architecture?

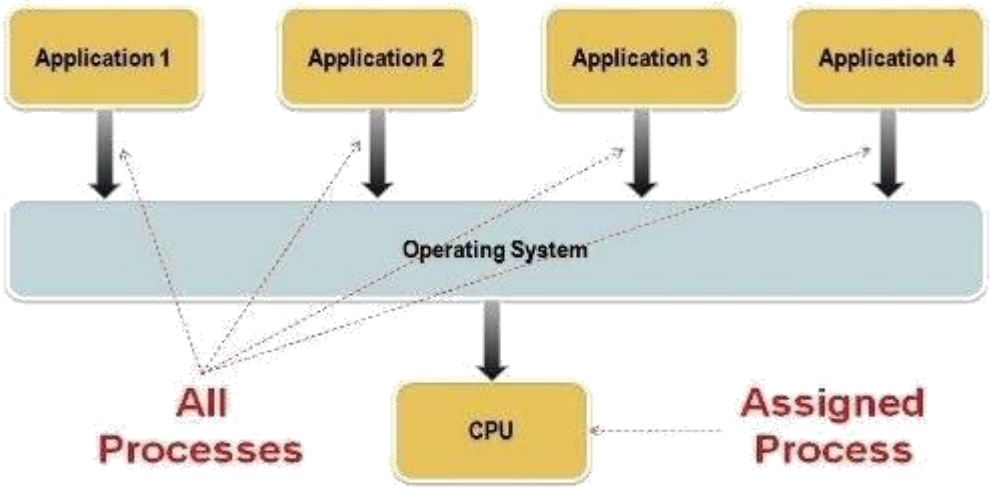
Computer is an electronic machine that makes performing any task very easy. In computer, the CPU executes each instruction provided to it, in a series of steps, this series of steps is called **Machine Cycle**, and is repeated for each instruction. One machine cycle involves fetching of instruction, decoding the instruction, transferring the data, executing the instruction.

- Most systems use a single general-purpose processor (PDAs through mainframes)
  - Most systems have special-purpose processors as well
- Multiprocessors systems growing in use and importance
  - Also known as parallel systems, tightly-coupled systems

- Advantages include
    1. Increased throughput
    2. Economy of scale
    3. Increased reliability – graceful degradation or fault tolerance
- Two types
    1. Asymmetric Multiprocessing
    2. Symmetric Multiprocessing

1. Single- processor systems:

Most systems use a single processor.On a single processor system, there is one main cpu capable of executing a general purpose instruction set,including instructions from user processes. A single processor system contains only one processor. So only one process can be executed at a time and then the process is selected from the ready queue. Single processor system can be further described using the diagram below:

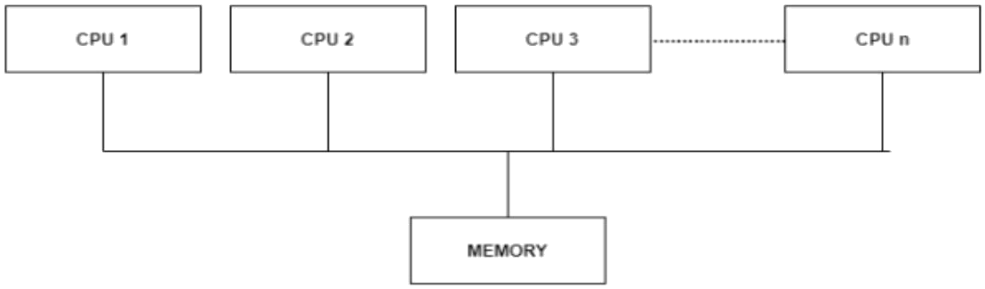


Single-processor system

As in the above diagram, there are multiple applications that need to be executed. However, the system contains a single processor and only one process can be executed at a time.

2. Multiprocessor systems

Most computer systems are single processor systems i.e they only have one processor. However, multiprocessor or parallel systems are increasing in importance nowadays. These systems have multiple processors working in parallel that share the computer clock, memory, bus, peripheral devices etc. It is also known as Parallel systems or tightly coupled systems. An image demonstrating the multiprocessor architecture is



Multiprocessing Architecture

Advantages of Multiprocessor Systems

There are multiple advantages to multiprocessor systems. Some of these are:

More reliable Systems

In a multiprocessor system, even if one processor fails, the system will not halt. This ability to continue working despite hardware failure is known as graceful degradation. For example: If there are 5 processors in a multiprocessor system and one of them fails, then also 4 processors are still working. So the system only becomes slower and does not ground to a halt.

Enhanced Throughput

By increasing the number of processors, we expect to get more work done in less time. If multiple processors are working , then the throughput of the system increases i.e. number of processes getting executed per unit of time increase.

More Economic Systems

Multiprocessor systems are cheaper than single processor systems in the long run because they share the data storage, peripheral devices, power supplies etc. If there are multiple processes that share data, it is better to schedule them on multiprocessor systems with shared data than have different computer systems with multiple copies of the data.

Types of Multiprocessor Systems:

There are two types of multiprocessing systems. **Symmetric Multiprocessing and Asymmetric Multiprocessing**.

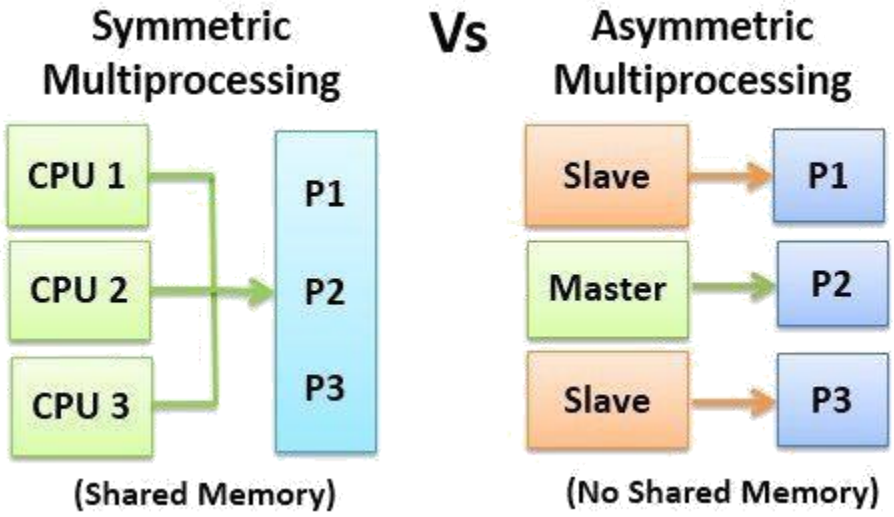
Asymmetric Multiprocessing

In asymmetric systems, each processor is assigned a specific task.There is a master processor that gives instruction to all the other processors. Asymmetric multiprocessor system contains a master slave relationship. There is one master processor that controls remaining slave processor. The master processor allots processes to slave processor, or they may have some predefined task to perform.

In case a master processor fails, one processor among the slave processor is made the master processor to continue the execution. In case if a slave processor fails, the other slave processor take over its job. Asymmetric Multiprocessing is **simple** as there only one processor that is controlling data structure and all the activities in the system.

Symmetric Multiprocessing

Symmetric Multiprocessing is one in which each processor performs all tasks within the operating system. All the processors are in a peer to peer relationship , it has **no master-slave** relationship like asymmetric multiprocessing. All the processors here communicate using the **shared memory**.



In Symmetric Multiprocessing, processors shares the same memory. In Asymmetric Multiprocessing there is a one master processor that controls the data structure of the system. The primary difference between Symmetric and Asymmetric Multiprocessing is that in **Symmetric Multiprocessing** all the processor in the system run tasks in OS. But, in **Asymmetric Multiprocessing** only the master processor run task in OS.

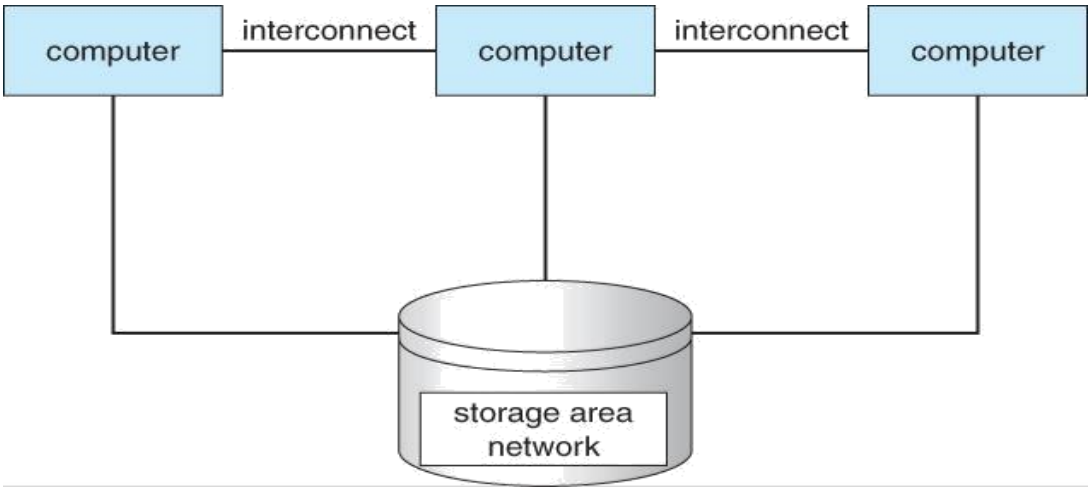
3. Clustered systems

Another type of multiple-CPU system is the Clustered System.

- Like multiprocessor systems, clustered systems gather together multiple CPUs to accomplish computational work.
- Clustered systems differ from parallel systems, however, in that they are composed of two or more individual systems coupled together.
- The definition of the term clustered is **not concrete;** the general accepted definitions is that clustered computers share storage and are closely linked via LAN networking.
- Clustering is usually performed to provide **high availability**.
- A layer of cluster software runs on the cluster nodes. Each node can monitor one or more of the others. If the monitored machine fails, the monitoring machine can take ownership of its storage, and restart the application(s) that were running on the failed machine. The failed machine can remain down, but the users and clients of the application would only see a brief interruption of service.

- Clustering can be structured asymmetrically or symmetrically.

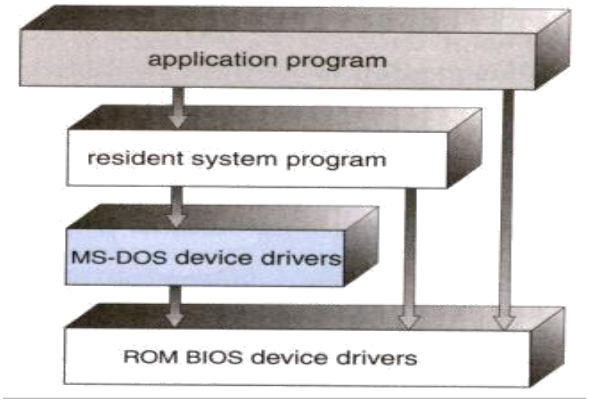
  - **Asymmetric Clustering -** In this, one machine is in hot standby mode while the other is running the applications. The hot standby host (machine) does nothing but monitor the active server. If that server fails, the hot standby host becomes the active server.
  - **Symmetric Clustering -** In this, two or more hosts are running applications, and they are monitoring each other. This mode is obviously more efficient, as it uses all of the available hardware.



5. EXPLAIN OPERATING SYSTEM STRUCTURE?

Simple Structure

There are several commercial system that don't have a well- defined structure such operating systems begins as small, simple & limited systems and then grow beyond their original scope. MS-DOS is an example of such system. It was not divided into modules carefully. Another example of limited structuring is the UNIX operating system.



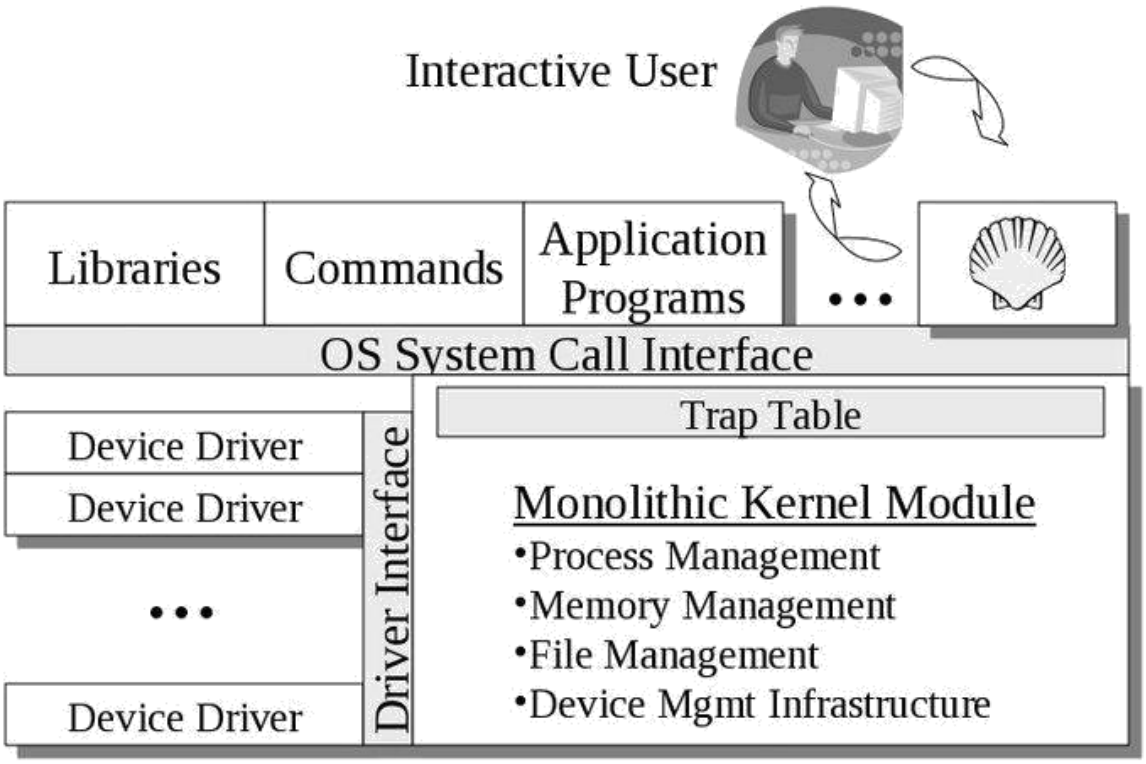
In MS-DOS, applications may by pass the operating system.

- Operating systems such as MS-DOS and the original UNIX did not have well-defined structures.

☐ There was no CPU Execution Mode (user and kernel), and so errors in applications could cause the whole system to crash.

Monolithic Approach

- Functionality of the OS is invoked with simple function calls within the kernel, which is one large program.
- Device drivers are loaded into the running kernel and become part of the kernel.

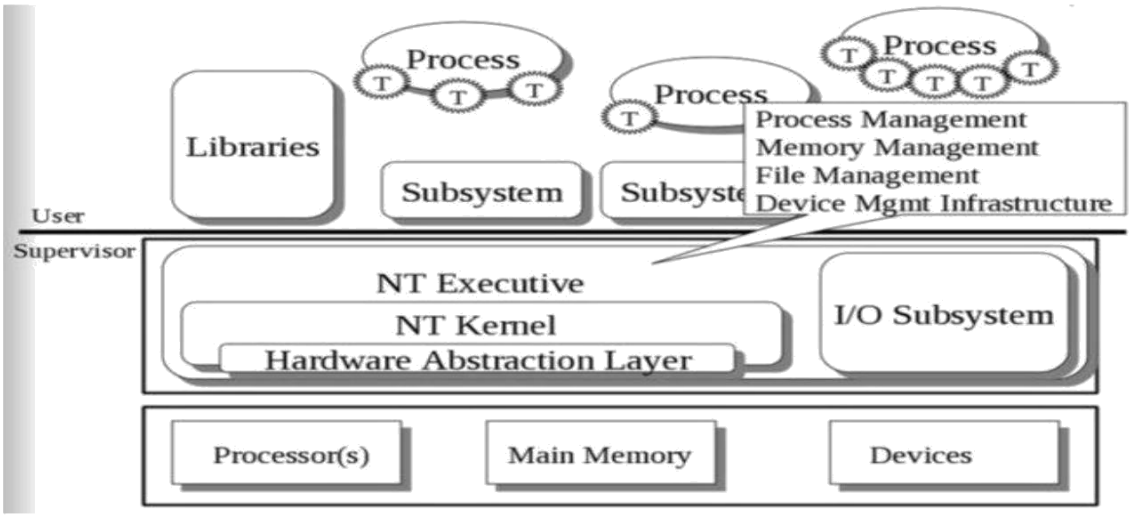


A monolithic kernel, such as Linux and other Unix systems.

Layered Approach

This approach breaks up the operating system into different layers.

- This allows implementers to change the inner workings, and increases modularity.
- As long as the external interface of the routines don't change, developers have more freedom to change the inner workings of the routines.
- With the layered approach, the bottom layer is the hardware, while the highest layer is the user interface.
    - The main advantage is simplicity of construction and debugging.
    - The main difficulty is defining the various layers.
    - The main disadvantage is that the OS tends to be less efficient than other implementations.



The Microsoft Windows NT Operating System. The lowest level is a monolithic kernel, but many OS components are at a higher level, but still part of the OS.

Microkernels

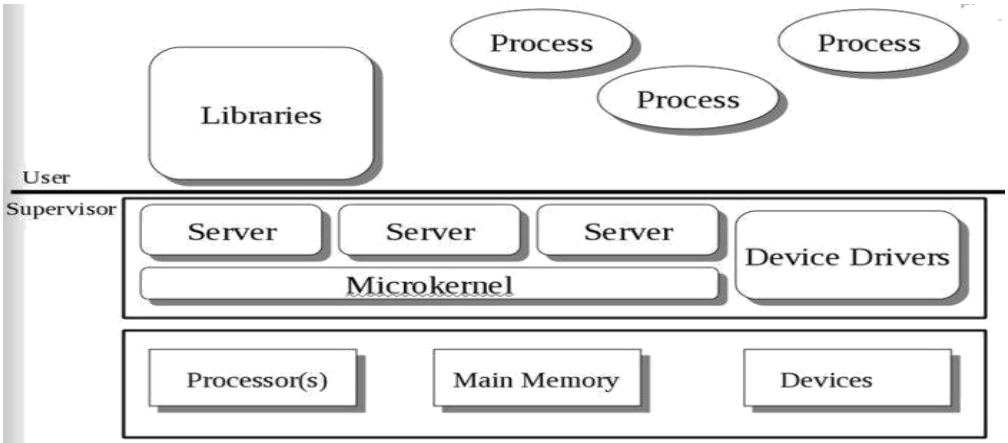
This structures the operating system by removing all nonessential portions of the kernel and implementing them as system and user level programs.

- Generally they provide minimal process and memory management, and a communications facility.
- Communication between components of the OS is provided by message passing.

The benefits of the microkernel are as follows:

- Extending the operating system becomes much easier.
- Any changes to the kernel tend to be fewer, since the kernel is smaller.
- The microkernel also provides more security and reliability.

Main disadvantage is poor performance due to increased system overhead from message passing.



A Microkernel architecture

.

6. EXPLAIN OPERATING SYSTEM OPERATIONS?

- Interrupt driven by hardware.
- Software error or request creates exception or trap.
- Division by zero, request for operating system service.
- Other process problems include infinite loop, processes modifying each Other or the operating system.
- Dual-mode operation allows OS to protect itself and other system Components.
- User mode and kernel mode.
- Mode bit provided by hardware.
- Provides ability to distinguish when system is running user code or kernel code.
- System call changes mode to kernel, return from call resets it to user.

Modern operating systems are interrupt driven.

- If there are no processes to execute,
- no I/O devices to service, and no users to whom to respond,

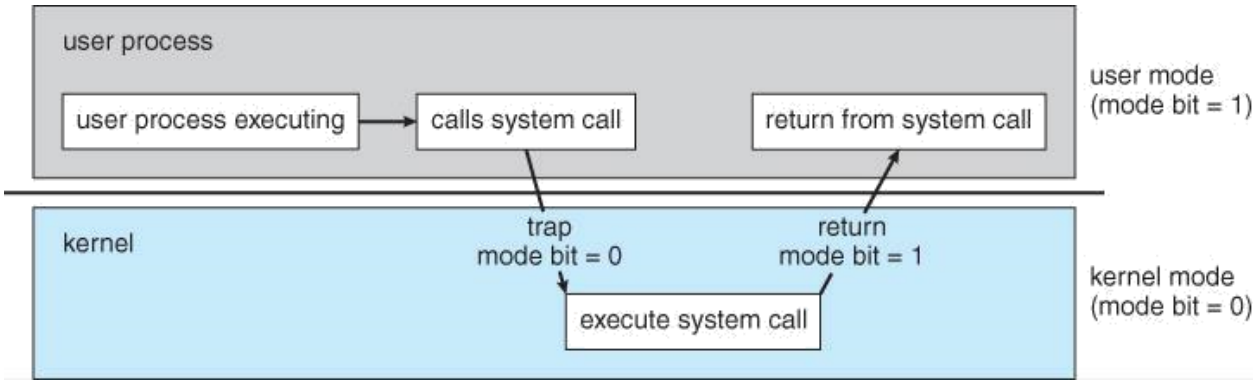
An operating system will sit quietly, waiting for something to happen. Events are almost always signalled by the occurrence of an **interrupt** or a **trap**. A trap (or an exception) is a software-generated interrupt caused either by an error (for example, division by zero or invalid memory access) or by a specific request from a user program that an operating-system service be performed.

- The interrupt-driven nature of an operating system defines that system's general structure. For each type of interrupt, separate segments of code in the operating system determine what action should be taken.
- An interrupt service routine is provided that is responsible for dealing with the interrupt. Since the operating system and the users share the hardware and software resources of the computer system, we need to make sure that an error in a user program could cause problems only for the one program that was running. With sharing, many processes could be adversely affected by a bug in one program. For example, if a process gets stuck in an infinite loop, this loop could prevent the correct operation of many other processes.

Dual-Mode Operation

In order to ensure the proper execution of the operating system, we must be able to distinguish between the execution of operating-system code and user defined code. The approach taken by most computer systems is to provide hardware support that allows us to differentiate among various modes of execution. At the very least, we need two separate modes of operation: **user mode** and **kernel mode** (also called **supervisor mode**, **system mode**, or **privileged mode**). A bit, called the **mode bit**, is added to the hardware of the computer to indicate the current mode: kernel (0) or user (1). With the mode bit, we are able to distinguish between a task that is executed on behalf of the operating system and one that is executed on behalf of the user.

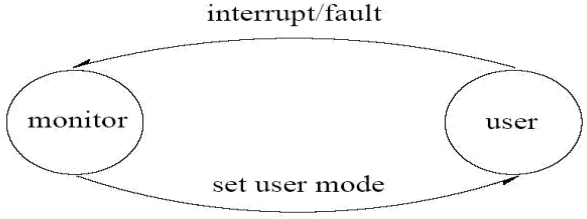
- When the computer system is executing on behalf of a user application, **the system is in user mode**.
- However, when a user application requests a service from the operating system (via a system call), it must **transition from user to kernel mode** to fulfill the request.
- At system boot time, the hardware starts in **kernel mode.**
- The operating system is then loaded and starts user applications in **user mode**.
- Whenever a trap or interrupt occurs, the hardware switches from **user mode to kernel mode** (that is, changes the state of the mode bit to 0).
- Thus, whenever the operating system gains control of the computer, it is in **kernel mode**.
- The system always **switches to user mode** (by setting the mode bit to 1) before passing control to a user program.



At system boot time, the hardware starts in kernel mode. The Operating system is then loaded and starts user applications in user mode. Whenever a trap or interrupt occurs, the hardware switches from user mode to kernel mode. Thus, whenever the operating system gains control of the computer, it is in kernel mode. The system always switches to user mode before passing control to a user program.

The dual mode of operation provides us with the means for protecting the operating system from errant users—and errant users from one another. We accomplish this protection by designating some of the machine instructions that may cause harm as privileged instructions. The hardware allows **privileged instructions** to be executed only in kernel mode. If an attempt is made to execute a privileged instruction in user mode, the hardware does not execute the instruction but rather treats it as illegal and traps it to the operating system.

System calls provide the means for a user program to ask the operating system to perform tasks reserved for the operating system on the user program's behalf. A system call is invoked in a variety of ways, depending on the functionality provided by the underlying processor.



When a system call is executed, it is treated by the hardware as software interrupt. Control passes through the interrupt vector to a service routine in the operating system, and the mode bit is set to kernel mode. The system call service routine is a part of the operating system. The kernel examines the interrupting instruction to determine what system call has occurred; a parameter indicates what type of service the user program is requesting. Additional information needed for the request may be passed in registers, on the stack, or in memory (with pointers to the memory locations passed in registers). The kernel verifies that the parameters are correct and legal, executes the request, and returns control to the instruction following the system call. The lack of a hardware-supported dual mode can cause serious shortcomings in an operating system.

When a program error occurs, the operating system must terminate the program abnormally. This situation is handled by the same code as is a user-requested abnormal termination. An appropriate error message is given, and the memory of the program may be dumped. The memory dump is usually written to a file so that the user or programmer can examine it and perhaps correct it and restart the program.

Timer

The operating system **maintains control over the CPU**. We must **prevent** a user program from **getting stuck in an infinite loop** or not calling system services and never returning control to the operating system. To accomplish this goal, we can use a timer. A timer can be set to interrupt the computer after a specified period. The period may be fixed (for example, 1/60 second) or variable (for example, from 1 millisecond to 1 second). A variable timer is generally implemented by a fixed-rate clock and a counter.

The operating system sets the counter. Every time the clock ticks, the counter is decremented. When the counter reaches 0, an interrupt occurs. For instance, a 10-bit counter with a 1-millisecond clock allows interrupts at intervals from 1 millisecond to 1,024 milliseconds, in steps of 1 millisecond. Before turning over control to the user, the operating system ensures that the timer is set to interrupt. If the timer interrupts, control transfers automatically to the operating system, which may treat the interrupt as a fatal error or may give the program more time. Thus, we can use the timer to prevent a user program from running too long. A simple technique is to initialize a counter with the amount of time that a program is allowed to run.

Every second, the timer interrupts and the counter is decremented by 1. As long as the counter is positive, control is returned to the user program. When the counter becomes negative, the operating system terminates the program for exceeding the assigned time limit.

7. EXPLAIN EVOLUTION OF OPERATING SYSTEMS?

i) Batch Systems

ii) Multiprogramming System

iii) Time Sharing System

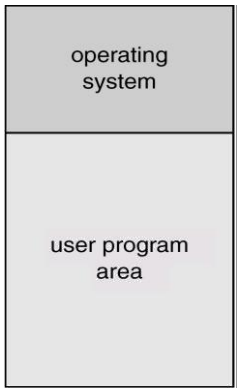
iv) Parallel Systems

v) Distributed Systems

vi) Real time Systems

vii) Networking System

BATCH SYSTEMS



Memory Layout for a Simple Batch System

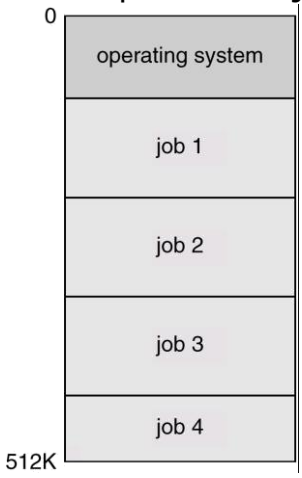
The users of a batch operating system do not interact with the computer directly. Each user prepares his job on an off-line device like punch cards and submits it to the computer operator. To speed up processing, jobs with similar needs are batched together and run as a group. The programmers leave their programs with the operator and the operator then sorts the programs with similar requirements into batches.

The problems with Batch Systems are as follows –

- Lack of interaction between the user and the job.
- CPU is often idle, because the speed of the mechanical I/O devices is slower than the CPU.
- Difficult to provide the desired priority.

MULTIPROGRAMMING SYSTEM

In the multiprogramming, CPU utilization is increased. A single user can't keep CPU and I/O devices busy at all times. The idea of multiprogramming is as follows: The operating system keeps several jobs in memory simultaneously.



Initially all the jobs are kept in job pool, a storage area in the disk. Some jobs from job pool are placed in memory. This requires some type of job scheduling. When operating system selects one job from the memory and begins to execute, this job may require some input from I/O device to complete. In the non-multiprogramming system, CPU will sit idle during this time. But in multiprogramming system, CPU will not sit idle, instead of, it switches to another job. When that jobs needs to wait, CPU switches to another job and soon. When the first finishes its I/O operation, CPU will go back to execute another job and the process continues and CPU will never sit idle.

TIME-SHARING OPERATING SYSTEMS

Time-sharing is a technique which enables many people, located at various terminals, to use a particular computer system at the same time. Time-sharing or multitasking is a logical extension of multiprogramming. Processor's time which is shared among multiple users simultaneously is termed as time-sharing.

The main difference between Multi-programmed Batch Systems and Time-Sharing Systems is that in case of Multi-programmed batch systems, the objective is to maximize processor use, whereas in Time-Sharing Systems, the objective is to minimize response time.

Multiple jobs are executed by the CPU by switching between them, but the switches occur so frequently. Thus, the user can receive an immediate response. For example, in a transaction processing, the processor executes each user program in a short burst or quantum of computation. That is, if **n** users are present, then each user can get a time quantum. When the user submits the command, the response time is in few seconds at most.

The operating system uses CPU scheduling and multiprogramming to provide each user with a small portion of a time. Computer systems that were designed primarily as batch systems have been modified to time-sharing systems.

Advantages of Timesharing operating systems are as follows –

- Provides the advantage of quick response.
- Avoids duplication of software.
- Reduces CPU idle time.

Disadvantages of Time-sharing operating systems are as follows –

- Problem of reliability.
- Question of security and integrity of user programs and data.
- Problem of data communication.

Parallel Systems:

Many systems are single processor systems i.e., they have only one CPU. However, **Multiprocessor Systems** (also known as **Parallel Systems**) have more than

one processor in close communication, sharing bus, the clock and sometimes memory and peripheral devices. Hence, these systems can also be called "Tightly Coupled Systems". The advantages of parallel systems are

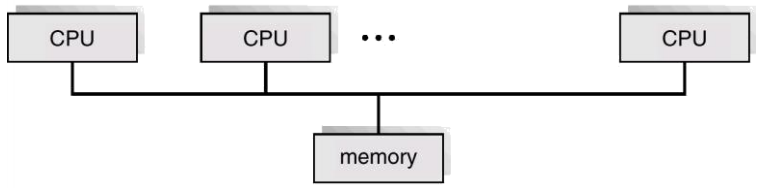
a) Increased throughput: The execution of the processes will be completed very fast when compare with single processor system.

b) Economy of scale: Multiprocessor systems save money while compare to the multiple single processor systems because sharing peripheral devices, memory, secondary storage, common bus etc.,

c) Increased reliability: If we have ten processors and one fails, then each of the remaining nine processors must pick up the share of a job and complete it. So the system works with less speed.

There are two types of multiprocessor systems. They are symmetric and asymmetric.

a) Symmetric Multiprocessor System: In this, each processor runs an identical copy of operating system and the processors communicate each other when they needed.



b) Asymmetric Multiprocessor System: In this, each processor has assigned a specific task and one processor controls all the remaining.

Distributed operating System

Distributed systems use multiple central processors to serve multiple real-time applications and multiple users. Data processing jobs are distributed among the processors accordingly.

The processors communicate with one another through various communication lines (such as high-speed buses or telephone lines). These are referred as **loosely coupled systems** or distributed systems. Processors in a distributed system may vary in size and function. These processors are referred as sites, nodes, computers, and so on.

The advantages of distributed systems are as follows –

- With resource sharing facility, a user at one site may be able to use the resources available at another.
- Speedup the exchange of data with one another via electronic mail.

- If one site fails in a distributed system, the remaining sites can potentially continue operating.
- Better service to the customers.
- Reduction of the load on the host computer.
- Reduction of delays in data processing

Real Time operating System

A real-time system is defined as a data processing system in which the time interval required to process and respond to inputs is so small that it controls the environment. 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.

Real-time systems are used when there are rigid time requirements on the operation of a processor or the flow of data and real-time systems can be used as a control device in a dedicated application. A real-time operating system must have well-defined, fixed time constraints, otherwise the system will fail. For example, scientific experiments, medical imaging systems, industrial control systems, weapon systems, robots, air traffic control systems, etc.

There are two types of real-time operating systems.

Hard real-time systems

Hard real-time systems guarantee that critical tasks complete on time. In hard real-time systems, secondary storage is limited or missing and the data is stored in ROM. In these systems, virtual memory is almost never found.

Soft real-time systems

Soft real-time systems are less restrictive. A critical real-time task gets priority over other tasks and retains the priority until it completes. Soft real-time systems have limited utility than hard real-time systems. For example, multimedia, virtual reality, Advanced Scientific Projects like undersea exploration and planetary rovers, etc.

Network operating System

A Network Operating System runs on a server and provides the server the capability to manage data, users, groups, security, applications, and other networking functions. The primary purpose of the network operating system is to allow shared file and printer access among multiple computers in a network, typically a local area network (LAN), a private network or to other networks.

Examples of network operating systems include Microsoft Windows Server 2003, Microsoft Windows Server 2008, UNIX, Linux, Mac OS X, Novell NetWare, and BSD.

The advantages of network operating systems are as follows –

- Centralized servers are highly stable.
- Security is server managed.
- Upgrades to new technologies and hardware can be easily integrated into the system.
- Remote access to servers is possible from different locations and types of systems.

The disadvantages of network operating systems are as follows –

- High cost of buying and running a server.
- Dependency on a central location for most operations.
- Regular maintenance and updates are required.

8. EXPLAIN OPERATING SYSTEM SERVICES?

An Operating System provides services to both the users and to the programs.

- It provides programs an environment to execute.
- It provides users the services to execute the programs in a convenient manner.

Following are a few common services provided by an operating system –

- Program execution
- I/O operations
- File System manipulation
- Communication
- Error Detection
- Resource Allocation
- Protection

Program execution Operating systems handle many kinds of activities from user programs to system programs like printer spooler, name servers, file server, etc. Each of these activities is encapsulated as a process.

A process includes the complete execution context (code to execute, data to manipulate, registers, OS resources in use). Following are the major activities of an operating system with respect to program management –

- Loads a program into memory.
- Executes the program.
- Handles program's execution.
- Provides a mechanism for process synchronization.
- Provides a mechanism for process communication.
- Provides a mechanism for deadlock handling.

I/O Operation An I/O subsystem comprises of I/O devices and their corresponding driver software. Drivers hide the peculiarities of specific hardware devices from the users.

An Operating System manages the communication between user and device drivers.

- I/O operation means read or write operation with any file or any specific I/O device.
- Operating system provides the access to the required I/O device when required.

File system manipulation A file represents a collection of related information. Computers can store files on the disk (secondary storage), for long-term storage purpose. Examples of storage media include magnetic tape, magnetic disk and optical disk drives like CD, DVD. Each of these media has its own properties like speed, capacity, data transfer rate and data access methods.

A file system is normally organized into directories for easy navigation and usage. These directories may contain files and other directions. Following are the major activities of an operating system with respect to file management −

- Program needs to read a file or write a file.
- The operating system gives the permission to the program for operation on file.
- Permission varies from read-only, read-write, denied and so on.
- Operating System provides an interface to the user to create/delete files.
- Operating System provides an interface to the user to create/delete directories.
- Operating System provides an interface to create the backup of file system.

Communication In case of distributed systems which are a collection of processors that do not share memory, peripheral devices, or a clock, the operating system manages communications between all the processes. Multiple processes communicate with one another through communication lines in the network.

The OS handles routing and connection strategies, and the problems of contention and security. Following are the major activities of an operating system with respect to communication −

- Two processes often require data to be transferred between them
- Both the processes can be on one computer or on different computers, but are connected through a computer network.
- Communication may be implemented by two methods, either by Shared Memory or by Message Passing.

Error handling Errors can occur anytime and anywhere. An error may occur in CPU, in I/O devices or in the memory hardware. Following are the major activities of an operating system with respect to error handling −

- The OS constantly checks for possible errors.
- The OS takes an appropriate action to ensure correct and consistent computing.

Resource Management In case of multi-user or multi-tasking environment, resources such as main memory, CPU cycles and files storage are to be allocated to

each user or job. Following are the major activities of an operating system with respect to resource management −

- The OS manages all kinds of resources using schedulers.
- CPU scheduling algorithms are used for better utilization of CPU.

Protection Considering a computer system having multiple users and concurrent execution of multiple processes, the various processes must be protected from each other's activities.

Protection refers to a mechanism or a way to control the access of programs, processes, or users to the resources defined by a computer system. Following are the major activities of an operating system with respect to protection −

- The OS ensures that all access to system resources is controlled.
- The OS ensures that external I/O devices are protected from invalid access attempts.
- The OS provides authentication features for each user by means of passwords.

Unit-II

PROCESS

1Q) Define process & explain about it?

A) Process is defined as execution of a program. So, it is called as program in execution. Every process contains process ID, memory, registers, states and so on...

There are various types of states in a process. They are:

1. New Born State
2. Ready State
3. Running State
4. Waiting/Blocked State
5. Terminator State

1. NEW BORN STATE: In an O.S, the process is being created. It is called as New BornState.

2. READY STATE:A process is ready to execute, but waiting for availability of resources. This is called Ready State.

3. RUNNING STATE: In a process, instructions are being executed. It is called as Running State.

4. WAITING/BLOCKED STATE: The process is waiting for a particular event. It is called as Waiting State.

5. TERMINATION STATE: In an O.S, the process has finished its execution. It is called as Termination State.

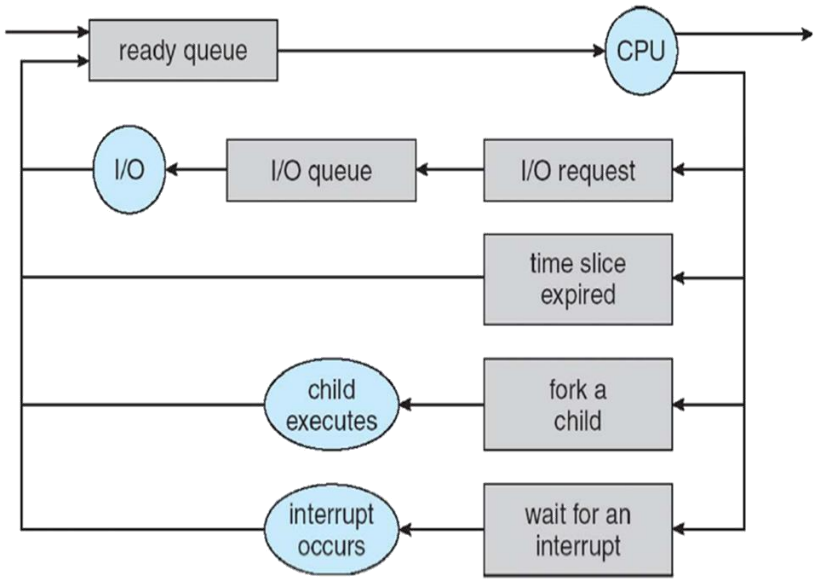
The state diagram of a process is represented as follows:

2Q) Write about Process Control Block (PCB) (OR) Task Control Block (TCB)

A)PCB:PCB stands for Process Control Block. It is also called as Task Control Block. In an O.S, the PCB contain process ID, process state, program counter, memory, accounting information, I/O status information and so on....The general structure of Process Control Block is represented as follows;

PROCESS I.D: In every O.S, each process has an identification
PROCESS STATE: Each process has a state. For example, the state may contain newborn state, ready state, running state, waiting state and termination state.
PROGRAM COUNTER: The functionality of memory is used to store the data and addresses. In the memory protection, it contains base register and unit register.
ACCOUNTING INFORMATION: In this functionality, it represents the data for time limits, cpu utilization, resources of I/O and memory and so on.....
I/O STATUS INFORMATION: The functionality of I/O status is used to represent the status of all I/O devices.
PCB: The functionality of PCB pointer is used to represent the next process.

3Q) Define process and Write about process scheduling ?
A) Process is defined as program under execution. In multiprogramming environment, the processes are executed based on scheduling. The scheduling is defined as order of execution. So with the help of scheduling the programs are executed in efficient manner.
In a process scheduling, it contains several queues. For example, the queues are job queue and ready queue. When a job is entered into a system, those jobs are placed/kept in a queue is called job queue. Whereas, in job queue, which job is ready to execute those jobs are kept in a queue is called ready queue.
In a process scheduling, there are 3 types of schedulers. They are:
1. LONG TERM SCHEDULER
2. SHORT TERM SCHEDULER
3. MIDDLE/MEDIUM TERM SCHEDULER
1. LONG TERM SCHEDULER: It is also called as job scheduler. It selects the process from the job pool and load into memory for execution.
2. SHORT TERM SCHEDULER: It is also called as cpu scheduler. The scheduler select the process from the ready queue and allocates the cpu one after another.
3. MIDDLE/MEDIUM TERM SCHEDULER: It is an intermediate level of long term scheduler and short term scheduler. The purpose of middle term scheduler is used to remove the process from main memory and disconnects the cpu if the process is completed.
In this process scheduling, the queueing diagram is represented as follows;



In this scheduling, the middle term scheduler contain swap in, swap out functionalities. It also contain context between the processes.
In the scheduling, it follows the header linked list. In the header linked list, the queue header contains two parts. They are;
1. HEAD NODE
2. TAIL
The head part contains the address part of the 1st PCB, whereas the tail part contains addresses part of the last.

4) Explain about threads?

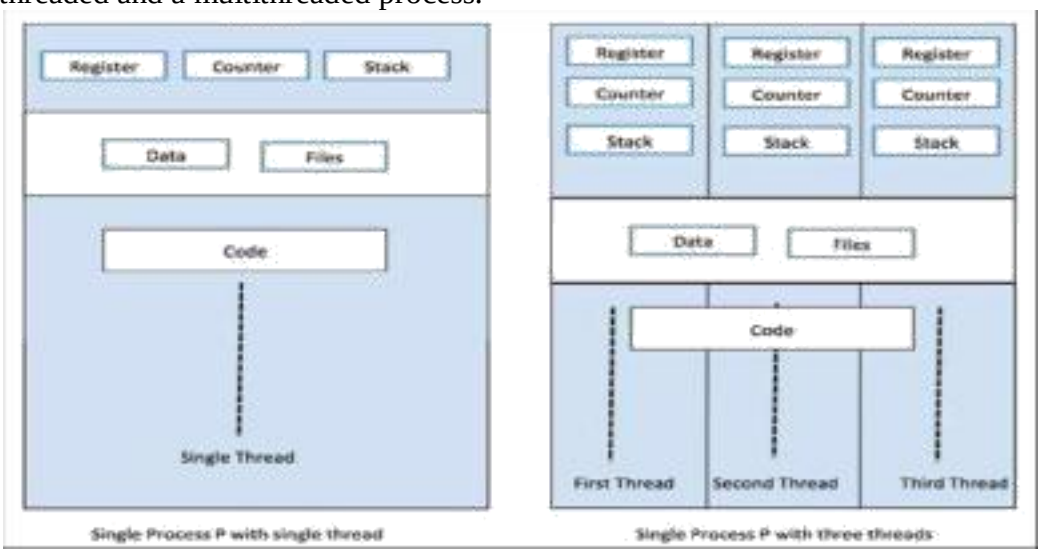
A)A thread is a flow of execution through the process code, with its own program counter that keeps track of which instruction to execute next, system registers which hold its current working variables, and a stack which contains the execution history.

A thread shares with its peer threads few information like code segment, data segment and open files. When one thread alters a code segment memory item, all other threads see that.

A thread is also called a **lightweight process**. Threads provide a way to improve application performance through parallelism. Threads represent a software approach to improving performance of operating system by reducing the overhead thread is equivalent to a classical process.

Each thread belongs to exactly one process and no thread can exist outside a process.

Each thread represents a separate flow of control. Threads have been successfully used in implementing network servers and web server. They also provide a suitable foundation for parallel execution of applications on shared memory multiprocessors. The following figure shows the working of a single-threaded and a multithreaded process.



Advantages of Thread

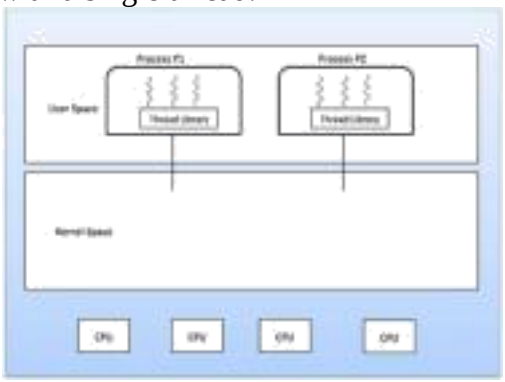
- Threads minimize the context switching time.
- Use of threads provides concurrency within a process.
- Efficient communication.
- It is more economical to create and context switch threads.
- Threads allow utilization of multiprocessor architectures to a greater scale and efficiency.

Types of Thread

Threads are implemented in following two ways:

- **User Level Threads** -- User managed threads
- **Kernel Level Threads** -- Operating System managed threads acting on kernel, an operating system core.

User Level Threads: In this case, the thread management kernel is not aware of the existence of threads. The thread library contains code for creating and destroying threads, for passing message and data between threads, for scheduling thread execution and for saving and restoring thread contexts. The application starts with a single thread.



Advantages

- Thread switching does not require Kernel mode privileges.
- User level thread can run on any operating system.
- Scheduling can be application specific in the user level thread.

- User level threads are fast to create and manage.

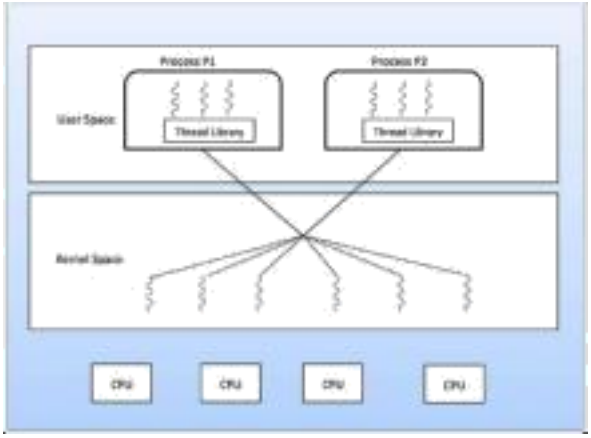
Disadvantages

- In a typical operating system, most system calls are blocking.
- Multithreaded application cannot take advantage of multiprocessing.

Kernel Level Threads

In this case, thread management is done by the Kernel. There is no thread management code in the application area. Kernel threads are supported directly by the operating system. Any application can be programmed to be multithreaded. All of the threads within an application are supported within a single process.

The Kernel maintains context information for the process as a whole and for individuals threads within the process. Scheduling by the Kernel is done on a thread basis. The Kernel performs thread creation, scheduling and management in Kernel space. Kernel threads are generally slower to create and manage than the user threads.



Advantages

- Kernel can simultaneously schedule multiple threads from the same process on multiple processes.
- If one thread in a process is blocked, the Kernel can schedule another thread of the same process.
- Kernel routines themselves can be multithreaded.

- Kernel threads are generally slower to create and manage than the user threads.
- Transfer of control from one thread to another within the same process requires a mode switch to the Kernel.

5Q) Explain about CPU Scheduling algorithms ?

A) The functionality of scheduling is represented by order of execution. It is the functionality of mulltiprogramming environment. There are various types of scheduling algorithms to support in the operatingsystem. They are;

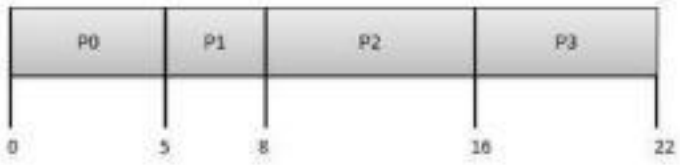
- First Come First Serve
- (FCFS) Shortest job first
- (SJF) Priority scheduling
- Round Robbin (RR) Multilevel queue
- scheduling Multilevel feedback queue
- scheduling

These algorithms are either **non-preemptive** or **preemptive**. Non-preemptive algorithms are designed so that once a process enters the running state, it cannot be preempted until it completes its allotted time, whereas the preemptive scheduling is based on priority where a scheduler may preempt a low priority running process anytime when a high priority process enters into a ready state.

First Come, First Served (FCFS)

- Jobs are executed on first come, first served
- basis. It is a non-preemptive scheduling
- algorithm. Easy to understand and implement.
- Its implementation is based on FIFO queue.
- Poor in performance, as average wait time is high.

Process	Arrival Time	Execute Time	Service Time
P0	0	5	0
P1	1	3	5
P2	2	8	8
P3	3	6	16

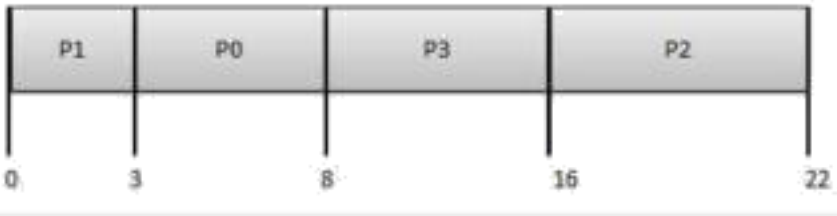


Wait time of each process is as follows:
Process Wait Time : Service Time - Arrival Time
P0 0 - 0 = 0
P1 5 - 1 = 4
P2 8 - 2 = 6
P3 16 - 3 = 13
Average Wait Time: (0+4+6+13) / 4 = 5.75

Shortest Job Next (SJN)

- This is also known as **shortest job first**, or SJF.
- This is a non-preemptive scheduling algorithm.
- Best approach to minimize waiting time.
- Easy to implement in Batch systems where required CPU time is known in advance.
- Impossible to implement in interactive systems where the required CPU time is not known.
- The processer should know in advance how much time a process will take.

Process	Arrival Time	Execute Time	Service Time
P0	0	5	3
P1	1	3	0
P2	2	8	16
P3	3	6	8

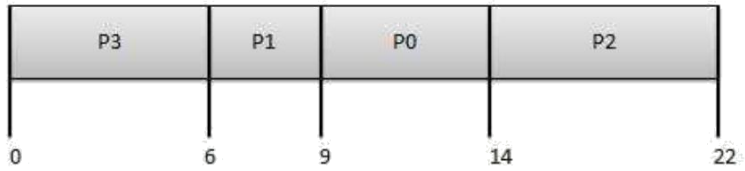


Wait time of each process is as follows:
Process Wait Time : Service Time - Arrival Time
P0 3 - 0 = 3
P1 0 - 0 = 0
P2 16 - 2 = 14
P3 8 - 3 = 5
Average Wait Time: (3+0+14+5) / 4 = 5.50

Priority Based Scheduling

- Priority scheduling is a non-preemptive algorithm and one of the most common scheduling algorithms in batch systems.
- Each process is assigned a priority. Process with highest priority is to be executed first and so on.
- Processes with same priority are executed on first come first served basis.
- Priority can be decided based on memory requirements, time requirements or any other resource requirement.

Process	Arrival Time	Execute Time	Priority	Service Time
P0	0	5	1	9
P1	1	3	2	6
P2	2	8	1	14
P3	3	6	3	0



Wait time of each process is as follows:

Process Wait Time : Service Time - Arrival Time

P0 9 - 0 = 9

P1 6 - 1 = 5

P2 14 - 2 = 12

P3 0 - 0 = 0

Average Wait Time: (9+5+12+0) / 4 = 6.5

Shortest Remaining Time

 Shortest remaining time (SRT) is the preemptive version of the SJN algorithm.

 The processor is allocated to the job closest to completion but it can be pre-empted by a newer ready job with shorter time to completion.

 Impossible to implement in interactive systems where required CPU time is not known.

 It is often used in batch environments where short jobs need to be given preference.

Round Robin Scheduling

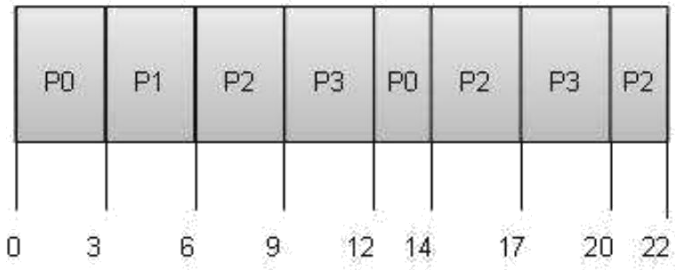
 Round Robin is a preemptive process scheduling algorithm.

 Each process is provided a fix time to execute; it is called a **quantum**.

 Once a process is executed for a given time period, it is preempted and other process executes for a given time period.

 Context switching is used to save states of preempted processes.

Quantum = 3



Wait time of each process is as follows:

Process Wait Time : Service Time - Arrival Time

P0 (0-0) + (12-3) = 9

P1 (3-1) = 2

P2 (6-2) + (14-9) + (20-17) = 12

P3 (9-3) + (17-12) = 11

Average Wait Time: (9+2+12+11) / 4 = 8.5

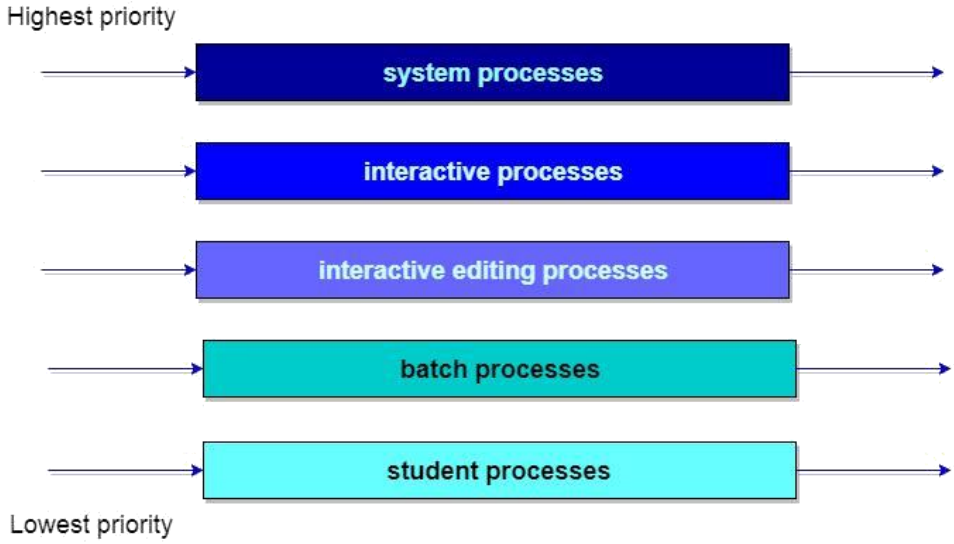
MULTILEVEL QUEUE SCHEDULING:

In the multilevel queue scheduling, the queue can be divided into several queues. It can be divided based on priority each queue follows its own algorithm.

For the following example, the queue can be divided into 5 separate queues.Thet are;

1. SYSTEM PROCESS
2. INTERACTIVE PROCESS
3. INTERACTIVE EDITING PROCESS
4. BATCH PROCESS
5. STUDENT PROCESS

It is represented as follows;



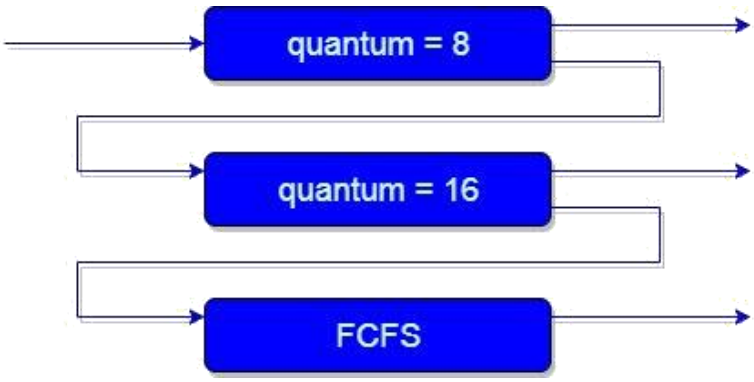
The processes are permanently assigned to one queue and executes its own scheduling algorithm

MULTILEVEL FEEDBACK QUEUE SCHEDULING:

In multilevel queue, the process are permanently assigned to a queue. The processes do not move between the queues. In multilevel queue scheduling, it is possible to move between the
Queues. In generally, a multi-level feed back queue scheduler is defined by the following parameters.

* the no of queues
* the scheduling algorithm for each queue
* the method used to determine which queue a process while enter when that process needs the service.

For example, It is represented as follows;



6) Write about Co-Operating process?

A) In an O.S, there are two types of processes. They are;

1. INDEPENDENT PROCESS
2. CO-OPERATING PROCESS

A process which doesn't depend on any other process is called independent process, whereas, a process which depends on any other process is called co-operating process.

The co-operating process is implemented with the help of shared memory or message passing. It can provide various facilities for execution of user applications. They are;

1. MODULARITY
2. COMMUNICATION SPEED UP
3. INFORMATION SHARING
4. CONVENIENCE

1. MODULARITY: The functionality of modularity is used to divide the process into blocks. The blocks are called as modules.

2. COMMUNICATION SPEEDUP: The functionality of communication speedup is used to, if a process run faster then the process is divided into sub processes.

3. INFORMATION SHARING: The functionality of information sharing is used to perform simultaneous access with the help of shared memory (or) message passing.
4. CONVENIENCE: The functionality of convenience is used to perform various operations like printing, compiling, editing and so on...
The message passing (or) shared memory can be implemented by using producer-consumer algorithm.
The producer & consumer can share the common variables.
The variables are represented as follows:

```
Var n;
Var buffer: array [0....n-1]
in, out : 0....n-1;
int n, in, out;
int buffer [10];
```

Producer algorithm:

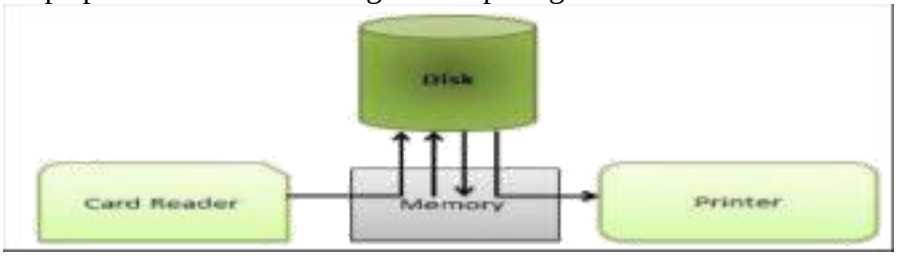
```
Repeat
Produce a value into x
while(in+1)%n=out do no operation
buffer [in]=x
in=(in+1)%n
.....
.....
until false
```

Consumer algorithm:

```
Repeat consume a value into x
While in=out do no operation
X=buffer [out]
Out = (out+1)%n
.....
.....
until false
```

- Buffer is a shared memory for producer and consumer.
- A producer produce a value, then that value is stored into buffer and the variable 'in' is incremented by 1 that is in=(in+1)%n
- A consumer consumes a value from the buffer, then the variable 'out' is incremented by 1 that is out=(out+1)%n.
- If (in = out, it indicates that the buffer is empty.
- In this algorithm, it follows circular queue
- It follows FIFO (First In First Out

7) Explain about spooling?
A)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.
An operating system does the following activities related to distributed environment:
- Handles I/O device data spooling as devices have different data access rates.
- Maintains the spooling buffer which provides a waiting station where data can rest while the slower device catches up.
- Maintains parallel computation because of spooling process as a computer can perform I/O in parallel fashion. It becomes possible to have the computer read data from a tape, write data to disk and to write out to a tape printer while it is doing its computing task.



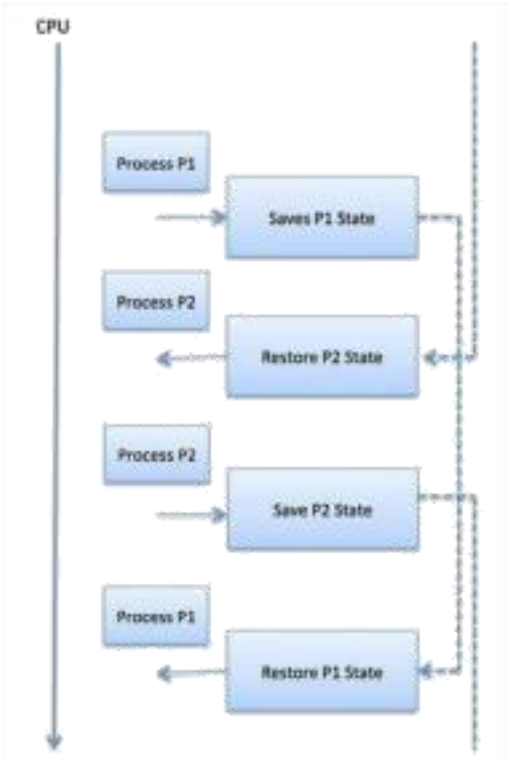
Advantages

- The spooling operation uses a disk as a very large buffer.
- Spooling is capable of overlapping I/O operation for one job with processor operations for another job.

8) Explain about context switching?

A) A context switch is the mechanism to store and restore the state or context of a CPU in Process Control block so that a process execution can be resumed from the same point at a later time. Using this technique, a context switcher enables multiple processes to share a single CPU. Context switching is an essential part of a multitasking operating system features.

When the scheduler switches the CPU from executing one process to execute another, the state from the current running process is stored into the process control block. After this, the state for the process to run next is loaded from its own PCB and used to set the PC, registers, etc. At that point, the second process can start executing.



Context switches are computationally intensive since register and memory state must be saved and restored. To avoid the amount of context switching time, some hardware systems employ two or more sets of processor registers. When the process is switched, the following information is stored for later use.

- Program Counter
- Scheduling information
- Base and limit register value
- Currently used register
- Changed State
- I/O State information
- Accounting information

9) Define O.S and write about system calls ?

A) An O.S is defined as interface between user and hardware. It is a system software. The purpose of O.S is used to execute the programs in efficient manner.

System calls is a collection of assembly language instructions. It provides the interface between process and O.S. There are 5 types of system calls to support in O.S. It is divided based on its functionality.

They are;

1. PROCESS CONTROL
2. FILE MANIPULATION
3. DEVICE MANIPULATION
4. INFORMATION MAINTAINANCE

5. COMMUNICATION

1.PROCESS CONTROL:

The system calls for the process control are represented as follows;

- Load,execute
- Create process, terminate process
- Wait for time
- Wait event, signal event
- Allocate and free memory

2. FILE MANIPULATION:

It is one of the function of operating system. The system calls are represented as follows;

- Create file, delete file
- Open,close
- Read, write, re-position
- Set the attributes

3. DEVICE MANIPULATION:

It is one of the function of O.S. The system calls are represented as follows;

- Request device, release device
- Read, write, re-position
- Logically attached or detached devices

4. INFORMATION MAINTAINANCE:

It is one of the function of O.S. It is used to store and retrieve the data. The system calls are represented as follows;

- Get time or date, set time or date
- Get system data, set system data
- Get process file, set process file
- Get device attributes, set devive attributes

5. COMMUNICATION:

The functionality of communication is used to share the data or transfer the data from one process to another. The system calls are represented as follows;

- Create, delete communication connection
- Send, receive messages
- Transfer information status
- Attach or detach remote devices

For example, the system calls are represented as follows;

- Fork( ) It creates child processes
- Fopen( ) It opens the file
- Fclose( ) It closes the file
- Create( ) It creates a file with specified format

10) Write about I/O structure and I/O interrupt ?

A) Interrupt is an error in the multiprogramming environment. In O.S, every process requires a processor and number of devices. In the modern system, the devices are connected to the common bus.

In the multiprogramming environment, there is a possibility of an interrupt. In general, interrupts are divided into two types. They are;

1. SOFTWARE INTERRUPT
2. HARDWARE INTERRUPT

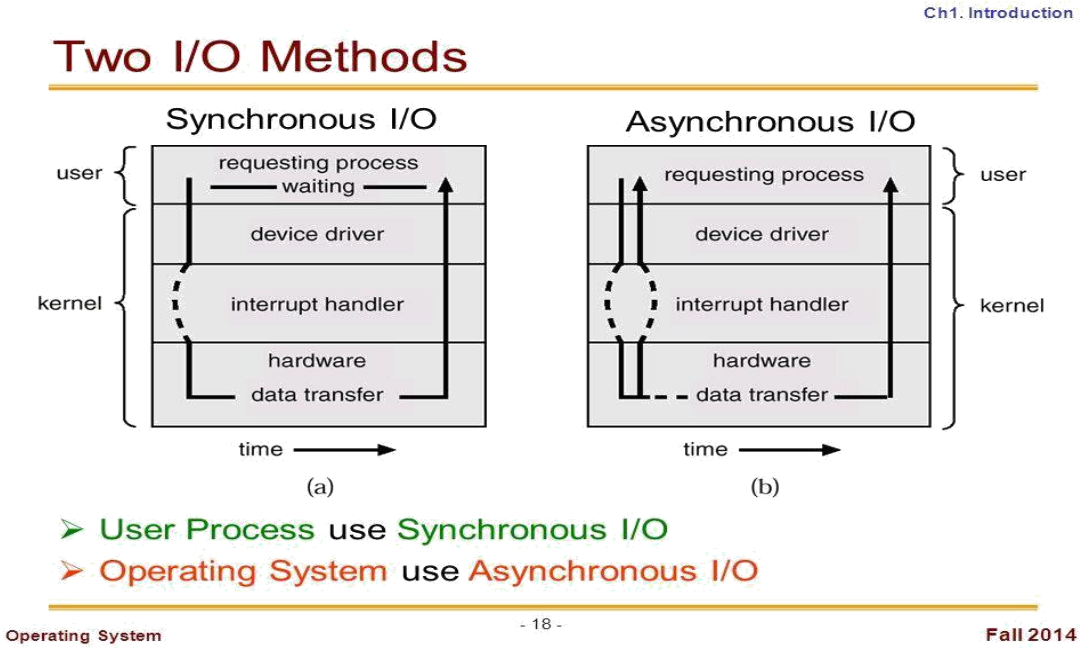
The software interrupt can be occurred during the fault in a program (Not proper input) whereas the hardware interrupt can be occurred due to physical components.

In O.S, there are two types of data transfers for I/O operations. They

- are; Synchronous I/O data transfer
- Asynchronous I/O data transfer

SYNCHRONOUS I/O DATA TRANSFER: In this functionality, once I/O is started and it is submitted to operating system. The control returns only if the task is completed i.e; after completion of I/O the control returns to the user process.

ASYNCHRONOUS I/O DATA TRANSFER: In this functionality, once I/O is started and it is submitted to operating system.The control returns if the task is not completed i.e;without completion of I/O the control returns to the user process. It can be represented as follows:



11) Explain about Interprocess Communication (IPC) ?

A) In operating system, processes are communicating in two ways. They are;

1. CO-OPERATING PROCESS
2. INTER-PROCESS COMMUNICATION

In co-operating process, the processes are communicating through a shared memory. In inter-process communication, the processes communicate with two functions. They are send & receive. There are various ways to implement IPC. They are;

1. DIRECT/INDIRECT COMMUNICATION
2. SYMMETRIC/ASYMMETRIC COMMUNICATION
3. AUTOMATIC/EXPLICITE COMMUNICATION
4. SEND BY COPY/SEND BY REFERENCE
5. FIXED SIZE/VARIABLE SIZE COMMUNICATION

DIRECT COMMUNICATION:

In the direct communication, each process is communicated with explicity name of the sender or receiver. In this mechanism, the send & receive functions are represented as follows;

- Send (A, message) Send a message to the process A.
- Receive (B, message) Receives the message from the process B.

In this communication the link has following properties;

- A link is associated with exactly two processes.
- Between each pair of processes, there exists exactly one link.
- The link may be unidirectional but it is usually bidirectional.

INDIRECT COMMUNICATION:

In this functionality, the messages are sent and received through mailbox. A mail box is an unique identification between the processes. It is a shared mailbox.

- Send (A, message) The message is sent to mailbox A.
- Receive (A, message) Receive the message from mailbox A.

In this communication, the link has following

- properties; It have a shared mailbox.
- A link may be either unidirectional (or) bidirectional.

SYMMETRIC/ASYMMETRIC COMMUNICATION:

In the symmetric communication, both sender & receiverknows the receiver name & receiver knows the sender name i.e;

- Send (p, message)
- Receive (q, message)

In asymmetric communication, only sender doesn't require name of the sender

- i.e; Send (p, message)
- Receive (ID, message)

BUFFERING:

It is used to represent the number of messages that can be stored temporarily. There are 3 ways.

1. ZERO CAPACITY:In this functionality, the queue length is zero i.e; the link doesn't have any messages.
2. BOUNDED CAPACITY: In this functionality, the queue has finite length i.e; the link contain n messages.
3. UNBOUNDED CAPACITY:In this functionality, the queue has infinite length i.e; the link contain any number of messages.

The producer and consumer can be implemented through interprocess communication (IPC) is represented as follows;

PRODUCER:

```
Repeat
Produce 'a' message
.....
Send (consumer, message)
.....
until false
```

CONSUMER:

```
Repeat
.....
Receive (Sender, message);
Consume the message
.....
until false
```

12Q) Define job queue, ready queue, waiting time, response-time, turn-around time, throughput?

A) Job queue: when a job enter into the system, those jobs are kept in a queue is called job queue.

Ready queue: In job queue, which job is ready to execute those jobs are kept in a queue is called ready queue. The queue follows FIFO.

Waiting time: Each job how much time spent in a ready queue is called waiting time.

Response time: if the process is executing, how much time taken to respond for the process(i.e.first reply) is called response time.

Turn around time: The delay between job submission and completion is called turn around time.

Throughput: The number of processes that are executed for the unit period of time is called throughput.

13Q) Difference between User-Level & Kernel-Level Thread

User-Level Threads	kernel-Level Threads
User-level threads are faster to create and manage.	Kernel-level threads are slower to create and manage
Implementation is by a thread library at the user level.	Operating system supports creation of Kernel threads
User-level thread is generic and can run on any operating system	Kernel-level thread is specific to the operating system
Multi-threaded applications cannot take advantage of multiprogramming	Kernel routines themselves can be multithreaded.

14Q) Difference between process and thread?

Process	Thread
Process is heavy weight or resource intensive.	Thread is lightweight, taking lesser resources than a process

Process switching needs interaction with operating system	Thread switching does not need to interact with operating system.
In multiple processing environments, each process executes the same code but has its own memory and file resources	All threads can share same set of open files, child processes
If one process is blocked, then no other process can execute until the first process is unblocked.	While one thread is blocked and waiting, a second thread in the same task can run.
Multiple processes without using threads use more resources	Multiple threaded processes use fewer resources.
Multiple threaded processes use fewer resources.	One thread can read, write or change another thread's data.

15Q) Comparison among Schedulers

Long-Term Scheduler	Short-Term Scheduler	Middle-Term Scheduler
It is a job scheduler	It is a CPU scheduler	It is a process swapping
Speed is lesser than short term scheduler	Speed is fastest among other two	Speed is in between both short and long term scheduler
It controls the degree of multiprogramming	It provides lesser control over degree of multiprogramming	It reduces the degree of multiprogramming.
It is almost absent or minimal in time sharing system	It is also minimal in time sharing system	It is a part of Time sharing systems.
It selects processes from pool and loads them into memory for execution	It selects those processes which are ready to execute	It can re-introduce the process into memory and execution can be continued

16Q) Differences between pre-emptive and non-pre-emptive scheduling?

Non-Pre-emptive scheduling	pre-emptive scheduling
Non-preemptive algorithms are designed so that once a process enters the running state(is allowed a process), it is not removed from the processor until it has completed its service time (or it explicitly yields the processor). context_switch() is called only when the process terminates or blocks.	Preemptive algorithms are driven by the notion of prioritized computation. The process with the highest priority should always be the one currently using the processor. If a process is currently using the processor and a new process with a higher priority enters, the ready list, the process on the processor should be removed and returned to the ready list until it is once again the highest-priority process in the system
In non-preemptive scheduling, a running task is executed till completion. It cannot be interrupted.	Tasks are usually assigned with priorities. At times it is necessary to run a certain task that has a higher priority before another task although it is running. Therefore, the running task is interrupted for some time and resumed later when the priority task has finished its execution. This is called preemptive scheduling.
Example: First In First Out, Shortest Job First	Example:, Round Robin

UNIT III

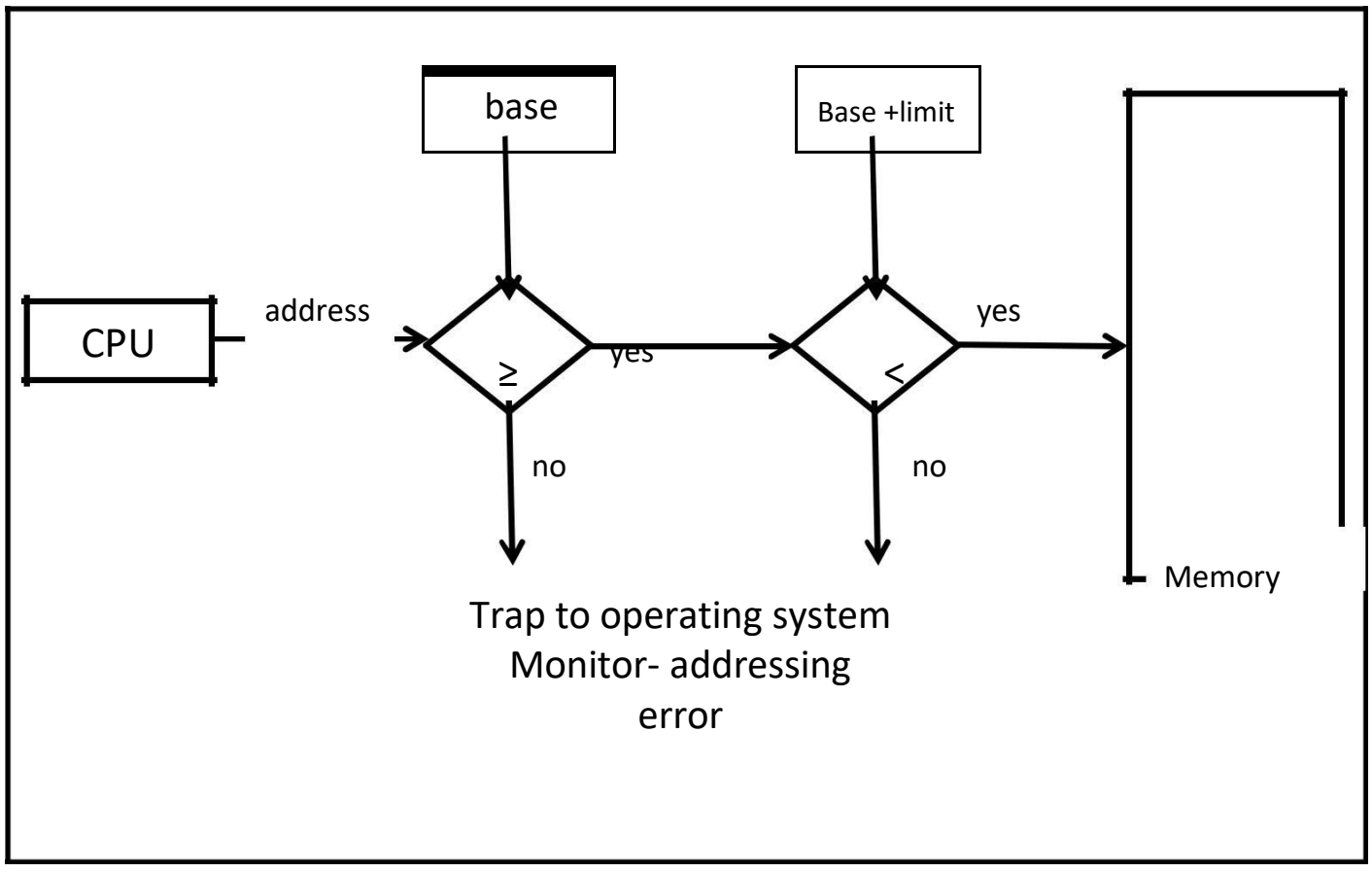
MEMORY MANAGEMENT

Logical and Physical Address space

- ➔ An address generated by the CPU is commonly referred to as a logical address, whereas an address seen by the memory unit – that is , the one loaded into the memory address register of the memory – is commonly referred to as a physical address.
- ➔ The set of all logical addresses generated by a program is a logical address space.
- ➔ The set of all physical addresses corresponding to these logical addresses is a physical address space.

Address binding

- ➔ To be executed, the program must be brought into memory and placed within a process.
- ➔ Depending on the memory management in use, the process may be moved between disk and memory during its execution

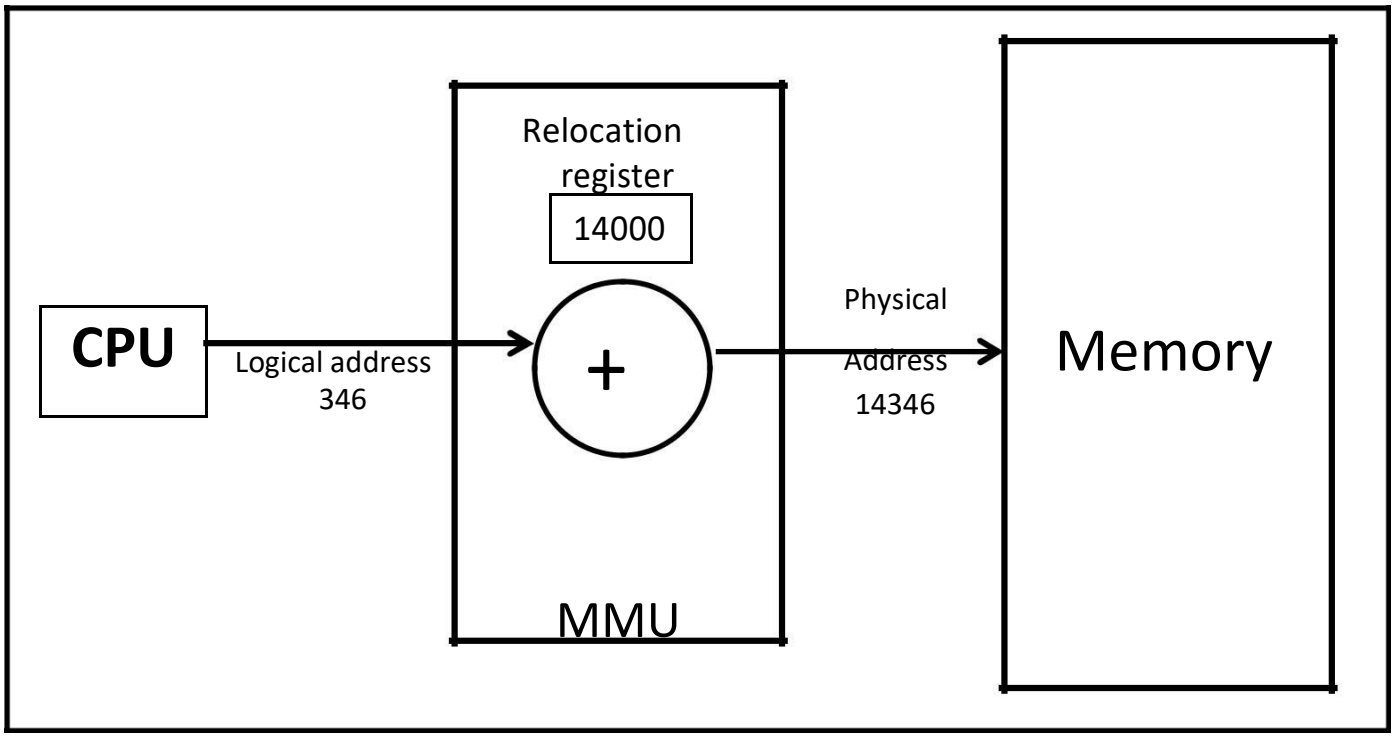


- ➔ Binding is a mapping from one address space to another.
- ➔ Classically , the binding of instructions and data to memory addresses can be done at any step along the way.
- ❖ Compile time : If you know at compile time where the process will reside in memory , then absolute code can be generated.

- ❖ Load time : If it is not known at compile time where the process will reside in memory, then the compile must generate relocatable code.
- ❖ Execution time : If he process can be moved during its execution from one memory segment to another, then binding must be delayed until run time.

Memory Management Unit (MMU)

- ➔ The run time mapping from virtual to physical addresses is done by a hardware device called memory management unit.
- ➔ The value in the relocation register is added to every address generated by a process at the time it is sent to the memory.
- ➔ For example, if the base is at 14000, then an attempt by the user to address location 0 is dynamically relocates to location 14000; an access to location 346 is mapped to location 14346.
- ➔ The user program never sees the real physical address.



Dynamic Loading

- ➔ The entire program and all data of a process must be in physical memory for the process to execute.
- ➔ To obtain better memory space utilization, we can use dynamic loading.
- ➔ With dynamic loading, a routine is not loaded until it is called.
- ➔ All routine are kept on disk in a relocatable load format.
- ➔ When a routine needs to call another routine, the calling routine first checks to see whether the other routine has been loaded.
- ➔ If not, the relocatable linking loader is called to load the desired routine into memory and to update the program's address tables to reflect this change.
- ➔ The advantage of dynamic loading is that an unused routine is never loaded.

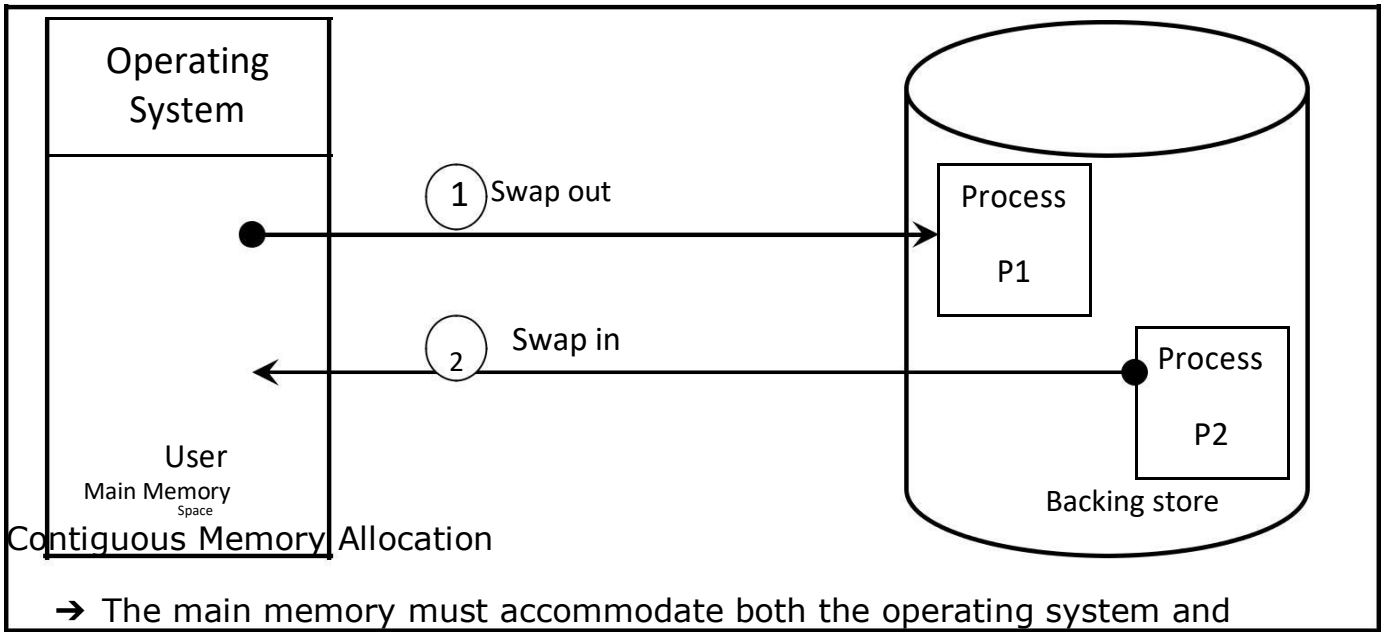
Swapping

- ➔ A process swapped temporary from memory to a backing store and then brought back into main memory for continued execution is called swapping.
- ➔ So, the swapping requires a backing store.
- ➔ The backing store is commonly a fast disk.
- ➔ For example, A multi programming environment with a round robin CPU scheduling algorithm.
- ➔ When the time slice expires, the memory manager will start to "swap out" the processes and "swap in" another processes into main memory.
- ➔ It can be represented as follows :

Backing store : fast disk large enough to accommodate copies of all memory images for all users; must provide direct access to these memory images.

Roll out, Roll in : Swapping variant used for priority –based scheduling algorithms; lower-priority process is swapped out so higher-priority process can be loaded ad executed.

- ➔ Major part of swap time is transfer time; total transfer time is directly proportional to the amount of memory swapped.
- ➔ Modified versions of swapping are found on many systems, i.e., UNIX, Linux, and windows.



Contiguous Memory Allocation

- ➔ The main memory must accommodate both the operating system and the various user processes.
- ➔ We therefore need to allocate different parts of the main memory in the most efficient way possible.
- ➔ The memory is usually divided into two partitions : one for the resident operating system and one for the user processes.
- ➔ We may place the operating system in either low memory.
- ➔ Programmers usually place the operating low memory
- ➔ In contiguous memory allocation , each process is contained in a single contiguous section memory.

Memory Allocation :

- ➔ One of the simplest methods for memory allocation is to divide memory into several fixed-sized partitions.
- ➔ Thus the degree of multiprogramming is bound by the number of partitions.
- ➔ In this multi partition methods, when a partition is free, a process is selected from the input queue and is loaded into the free partition.
- ➔ When the process terminates, the partition becomes available for another process.
- ➔ In the fixed-partition scheme, the operating system keeps a table indicating which parts of memory are available and which are occupied.
- ➔ Initially , all memory is available for user processes and is considered and considered one large block of a available memory, a hole.
- ➔ The first –fit, best –fit , and worst –fit strategies are the once most commonly used to select a free hole from the set of a available holes.
- ➔ First Fit : Allocate the first hole that is big enough.
    - Searching can start either at the beginning of the set of holes or where the previous first –fit search ended.
- ➔ Best fit : Allocate the smallest hole that is big enough.
    - We must search the entire list , unless the list is ordered by size.
- ➔ Worst fit : Allocate the largest hole.
    - Again, we must search the entire list, unless it is sorted by size.

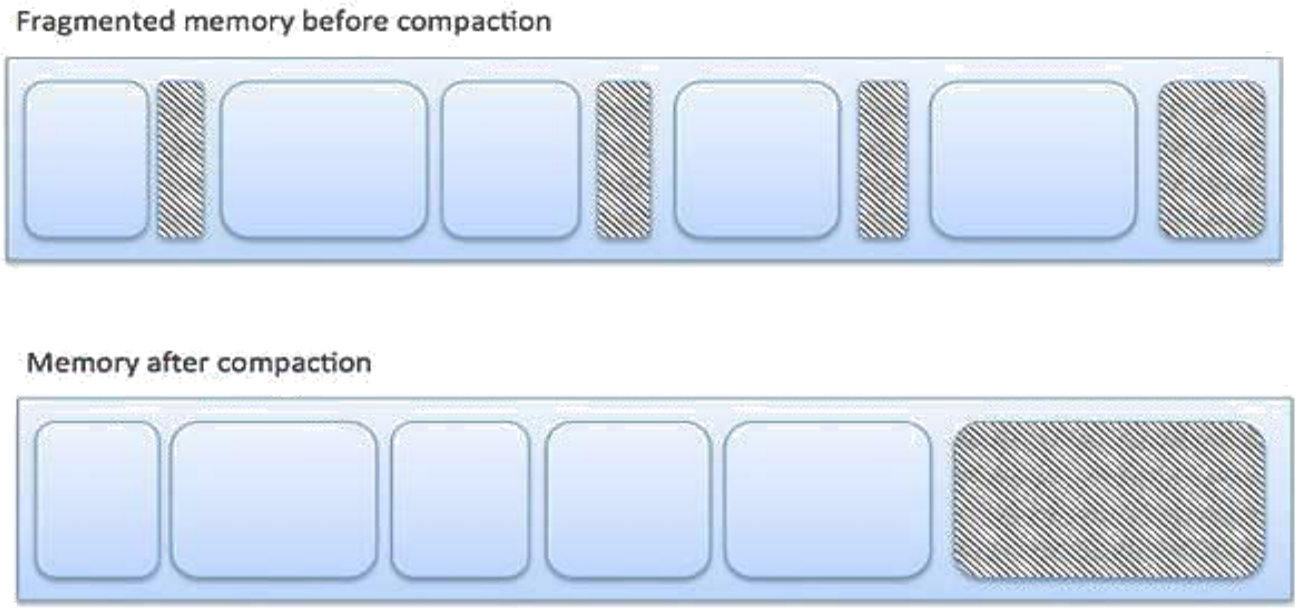
Fragmentation

- ➔ As processes are loaded and removed from memory, the free memory space is broken into little pieces.
- ➔ It happens after sometimes that processes cannot be allocated to memory blocks considering their small size and memory blocks remains unused.
- ➔ This problem is known as Fragmentation.

Fragmentation is of two types –

S.N.	Fragmentation & Description
1	**External fragmentation** Total memory space is enough to satisfy a request or to reside a process in it, but it is not contiguous, so it cannot be used.
2	**Internal fragmentation** Memory block assigned to process is bigger. Some portion of memory is left unused, as it cannot be used by another process.

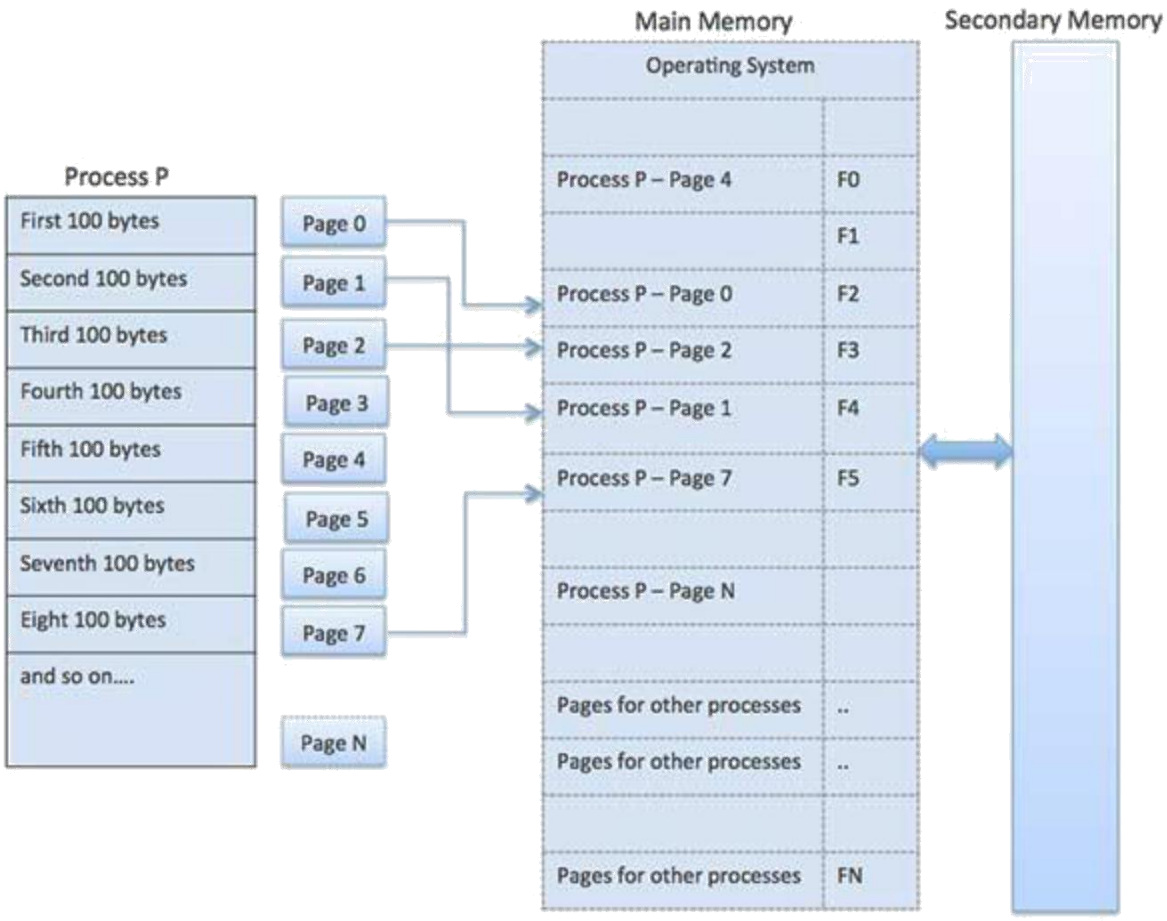
➔ The following diagram shows how fragmentation can cause waste of memory and a compaction technique can be used to create more free memory out of fragmented memory –



➔ External fragmentation can be reduced by compaction or shuffle memory contents to place all free memory together in one large block.
➔ To make compaction feasible, relocation should be dynamic.
➔ The internal fragmentation can be reduced by effectively assigning the smallest partition but large enough for the process.

Paging

➔ A computer can address more memory than the amount physically installed on the system.
➔ This extra memory is actually called virtual memory and it is a section of a hard that's set up to emulate the computer's RAM.
➔ Paging technique plays an important role in implementing virtual memory.
➔ Paging is a memory management technique in which process address space is broken into blocks of the same size called **pages** (size is power of 2, between 512 bytes and 8192 bytes). The size of the process is measured in the number of pages.
➔ Similarly, main memory is divided into small fixed-sized blocks of (physical) memory called **frames** and the size of a frame is kept the same as that of a page to have optimum utilization of the main memory and to avoid external fragmentation.



Address Translation

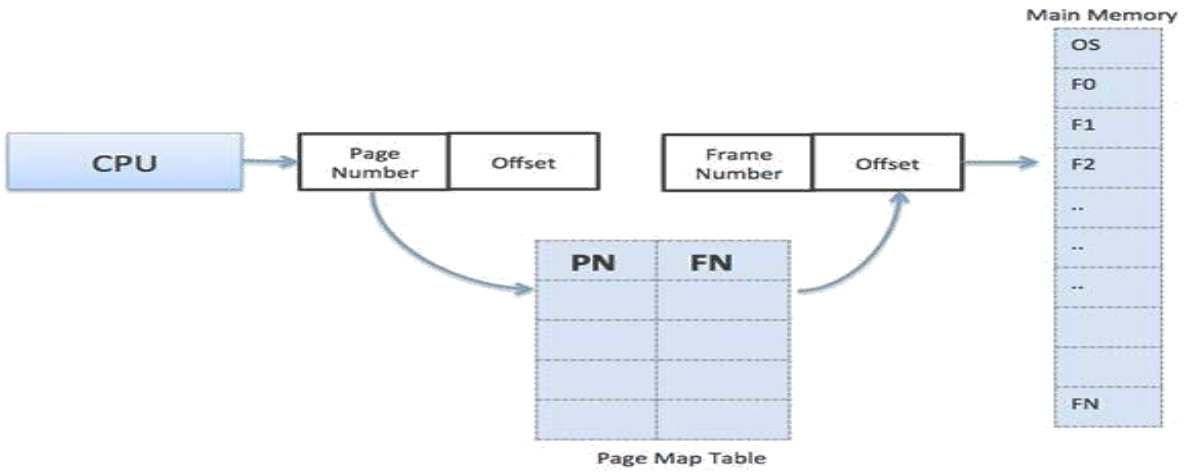
➔ Page address is called **logical address** and represented by **page number** and the **offset**.

Logical Address = Page number + page offset

Frame address is called **physical address** and represented by a **frame number**and the **offset**.

Physical Address = Frame number + page offset

A data structure called **page map table** is used to keep track of the relation between a page of a process to a frame in physical memory.



- ➔ When the system allocates a frame to any page, it translates this logical address into a physical address and create entry into the page table to be used throughout execution of the program.
- ➔ When a process is to be executed, its corresponding pages are loaded into any available memory frames.
- ➔ Suppose you have a program of 8Kb but your memory can accommodate only 5Kb at a given point in time, then the paging concept will come into picture.
- ➔ When a computer runs out of RAM, the operating system (OS) will move idle or unwanted pages of memory to secondary memory to free up RAM for other processes and brings them back when needed by the program.
- ➔ This process continues during the whole execution of the program where the OS keeps removing idle pages from the main memory and write them onto the secondary memory and bring them back when required by the program.

Advantages and Disadvantages of Paging

Here is a list of advantages and disadvantages of paging −

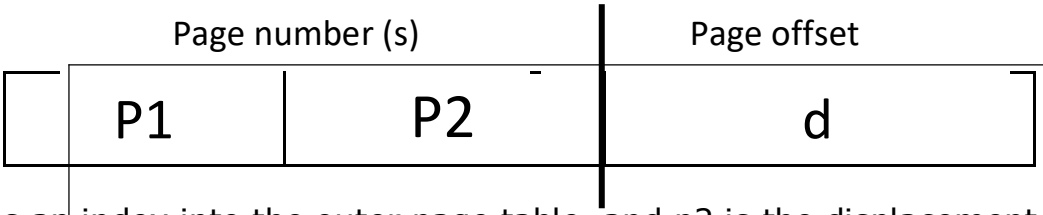
- Paging reduces external fragmentation, but still suffer from internal fragmentation.
- Paging is simple to implement and assumed as an efficient memory management technique.
- Due to equal size of the pages and frames, swapping becomes very easy.
- Page table requires extra memory space, so may not be good for a system having small RAM.

Page table structure
The following are the most common techniques for structuring the page table:

1. Hierarchical paging
2. Hashed page tables
3. Invented page tables

1) Hierarchical Paging (multilevel paging):

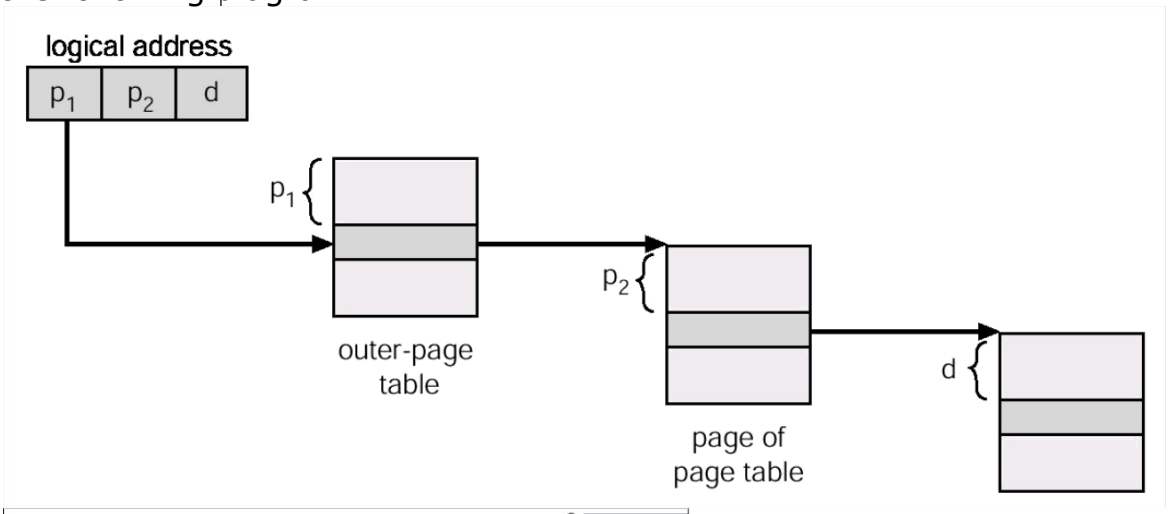
- ➔ Suppose a computer has a large amount of auxiliary memory space say 2^{32} words and we assume that page size is 1 KB that equal to 2^{10} , so we have $2^{32}/2^{10}$ = 222 pages.
- ➔ So our page table contains 2^{22} entities and we assume that the main memory capacity is 4 KB.
- ➔ So, at any given time only 4 pages out of 2^{22} pages are stored in the main memory i.e., our page table contains2^{22} – 4 null entries that is maximum amount of space in this page table is waste and it is not easy to store entire page table in contiguous memory location.
- ➔ This drawback is cleared in multilevel paging.
- ➔ In the two level paging scheme, page table is again paged.
- ➔ This is shown in the following diagram.

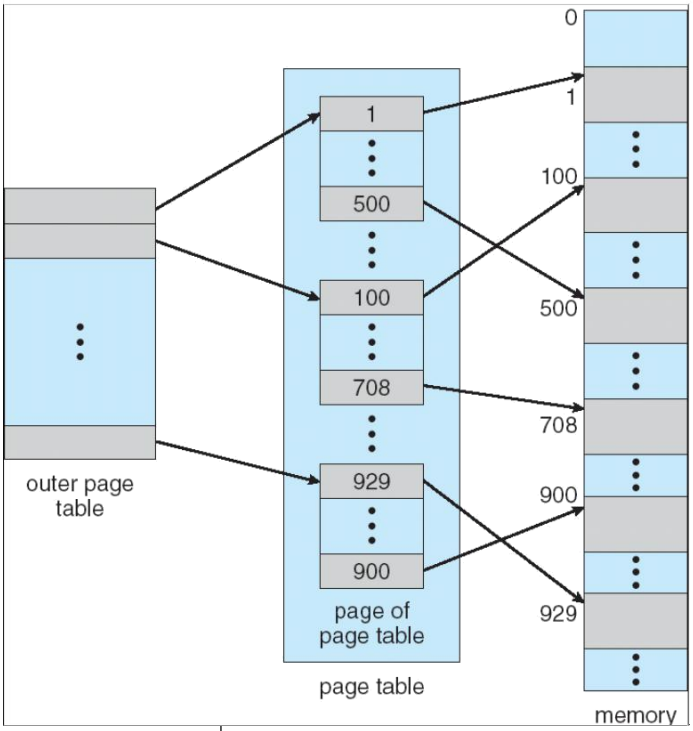


➔ P1 is an index into the outer page table, and p2 is the displacement within the page of the outer page table.

Address-Translation Scheme:

Address-translation scheme for a two-level 32-bit paging architecture and is shown in the following diagram:





2) Hashed Page Tables:

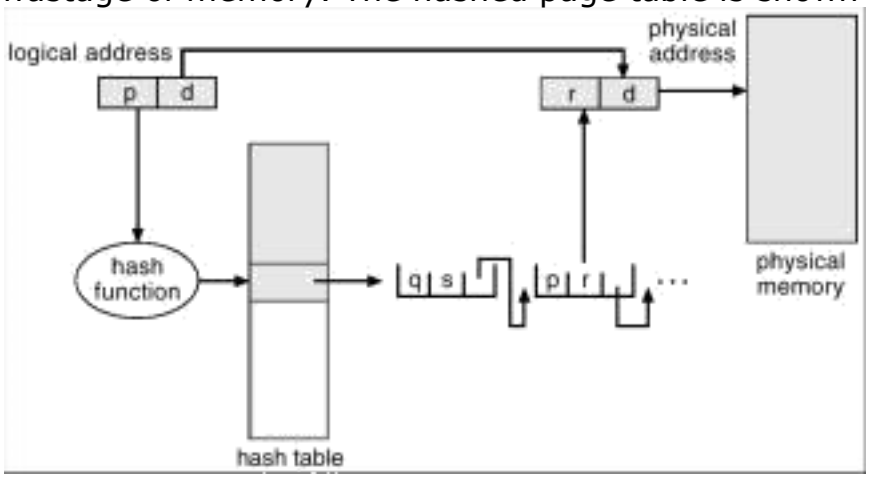
A common approach for handling address spaces larger than 32 bits is to use a *hashedpage table*, with the hash value being the virtual-page number. Each entry in the hash table contains a linked list of elements that hash to the same location (to handle collisions). Each element consists of 3 fields:

a) the virtual page number

b) the value of the mapped page frame and

c) a pointer to the next element in the linked list.

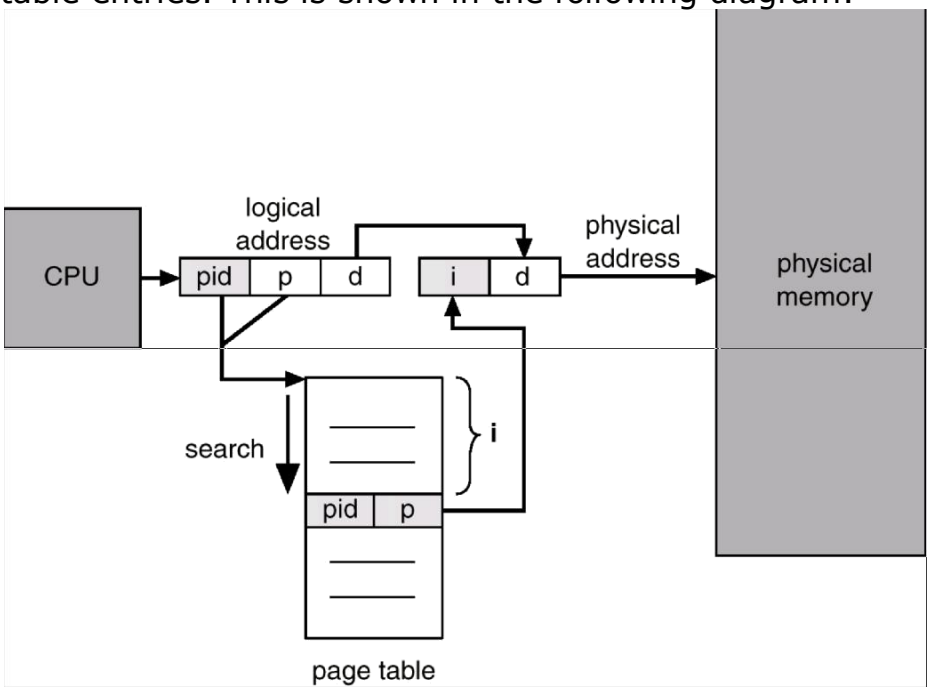
This method stores page table information in hash table for faster access & less wastage of memory. The hashed page table is shown in the following diagram:



3) Inverted Page Tables:

The modern computer allows a page table for each and every process. The entire page table is to be stored in the contiguous memory locations in the memory. Entry consists of the virtual address of the page stored in that real memory location, with information about the process that owns that page. Decreases memory needed to store each page table, but increases time needed to search the table when a page reference occurs.

It is better to use hash table to limit the search to one — or at most a few — page-table entries. This is shown in the following diagram:

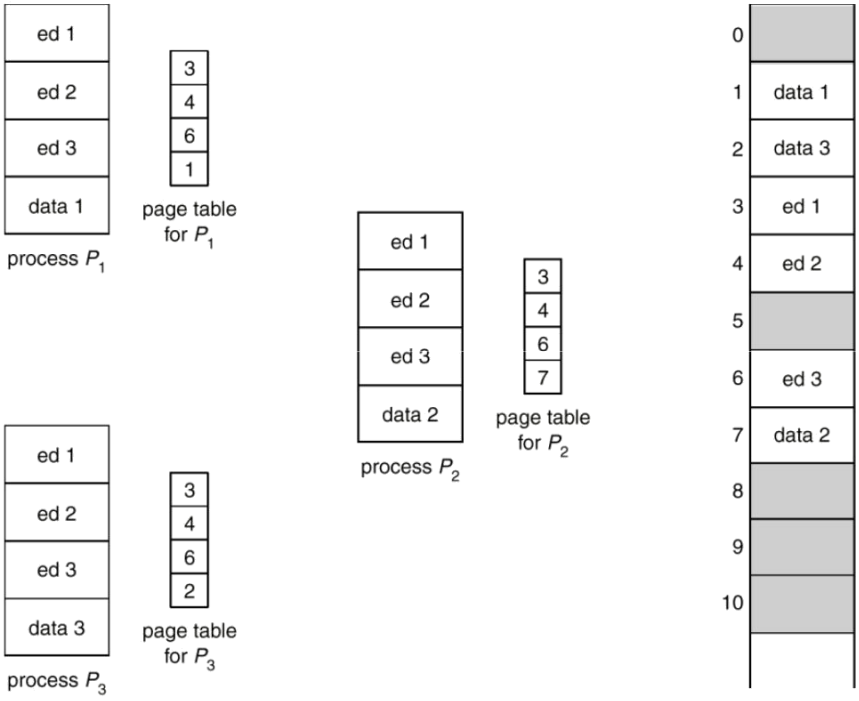


Shared Pages:

Another advantage of paging is the possibility of sharing common code. This concept is very useful in time sharing systems. Consider a system with 40 users and each process executes a text editor contains 150 KB

of code and 50 KB of data i.e., each text editor requires 200 KB and we require 8000 KB for 40 users. We assume that each text editor has a common code i.e., each text editor shares the common code.

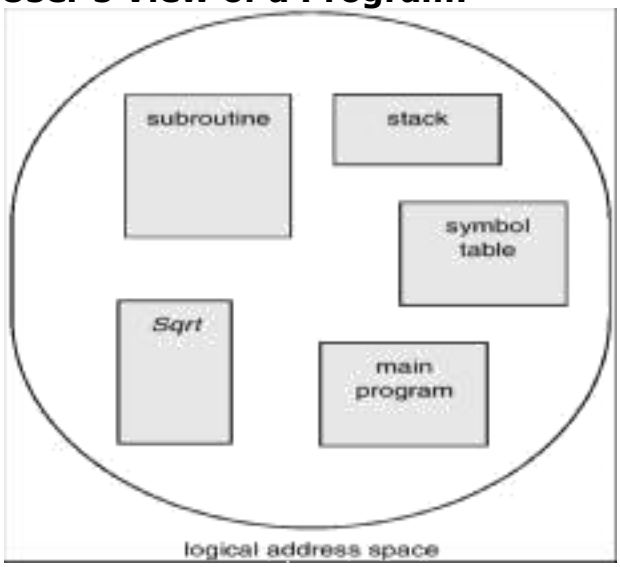
The following diagram shows sharing of pages for each process, For simplicity, we assume 3 processes, 3 page tables and 12 frames and each frame length is 50 KB.



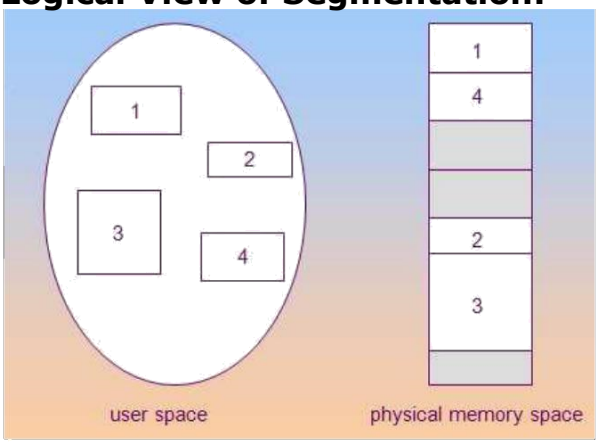
Q)Write about Segmentation?

A) Segmentation is a memory-management scheme that supports user view of memory. A program is a collection of segments. A segment is a logical unit such as main program, procedure, function, method, object, local variables, global variables, common block, stack, symbol table and arrays.

User's View of a Program:



Logical View of Segmentation:

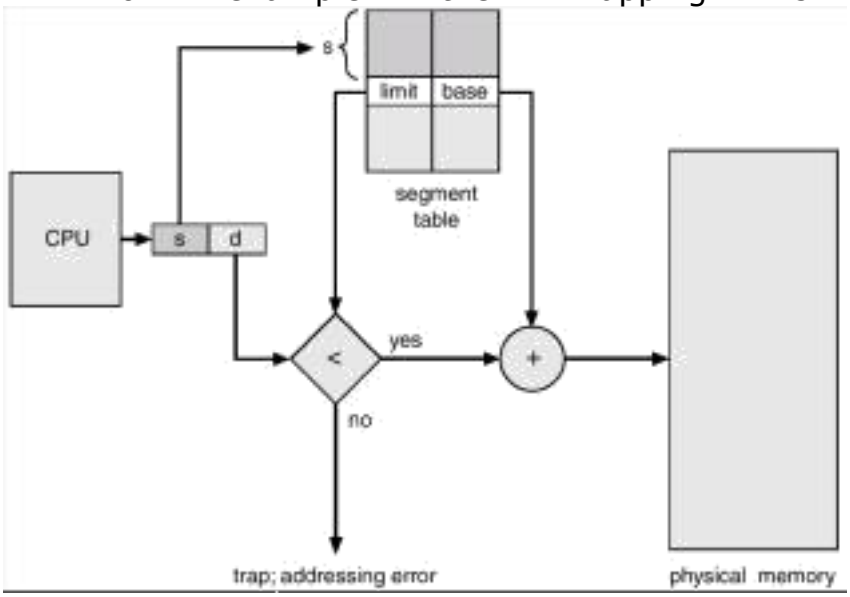


In the segmentation, the logical memory can be divided into fixed sized blocks. These blocks are called as segments. The segmentation is a memory management functionally that supports the user view of memory. In this, logical memory is divided into segments and each segment consists of segments number and offset.

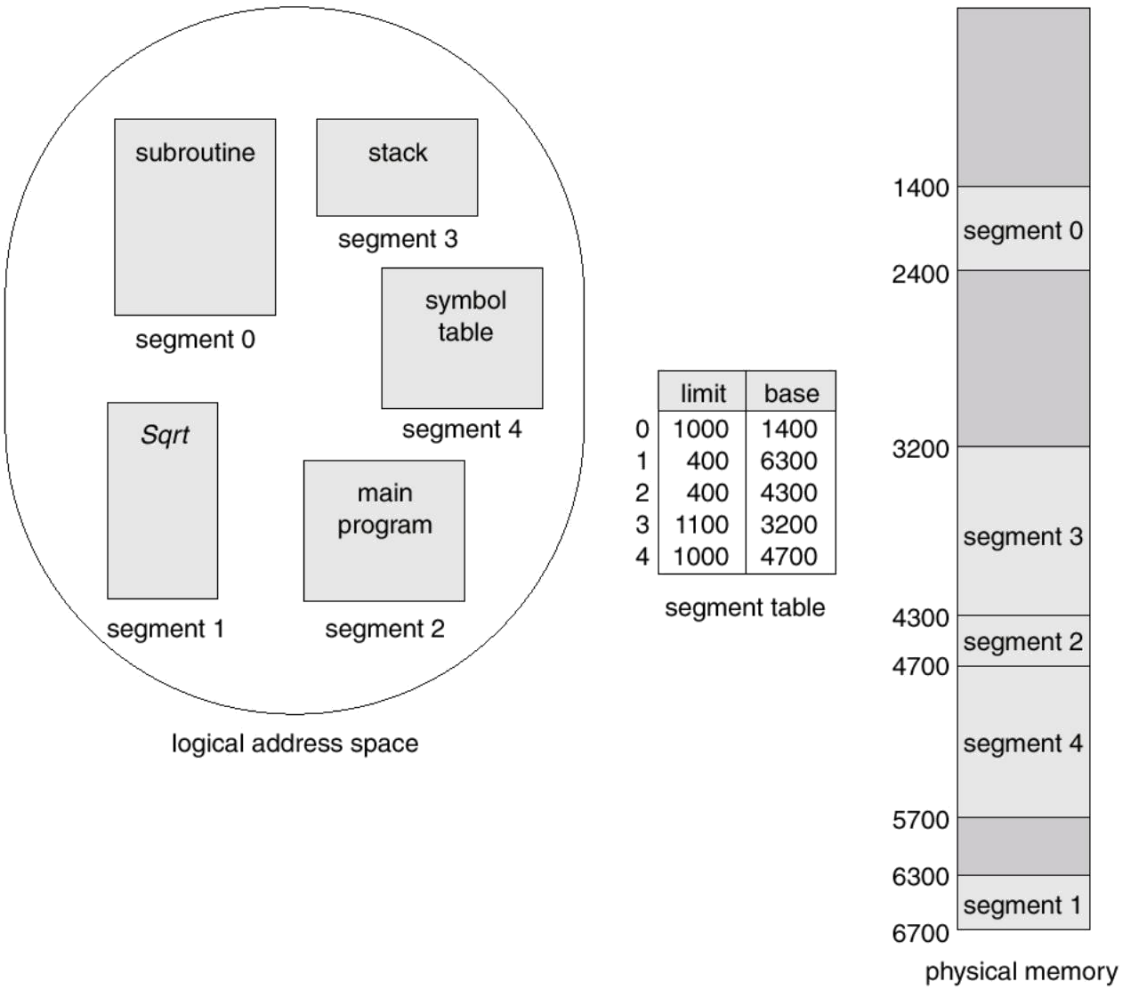
The segment table is used to mapping between the logical memory and physical memory. The segment table consists of two columns.

1) Base
2) Limit

For example the mapping is represented as follows;



The following segmentation example diagram shows segmentation table, physical memory for a logical address space which contains subroutine, sqrt, main program, stack and symbol table:



Advantages of segmentation:

1) view of memory is the user's view

2) segments are protected from one another each segment contains one type of information (e.g., instructions, stack, …)

3) sharing segments is logical and easy

->if all the instructions are in one segment and all data in another,

->the instruction segment can be shared freely by different processes (each with own data)

3) Segmentation with paging, can implement protection using a valid/invalid bit

Disadvantage:

1) Unlike paging, external fragmentation can be a problem

Q) Explain about Segmentation with Paging?

A) The IBM OS/2 32-bit operating system supports the segmentation with paging scheme. Here, the logical address space of a process is up to 16 KB and it is divided into 2 partitions. The first partition consists 8 KB and contains the segments which are also used by other processes.

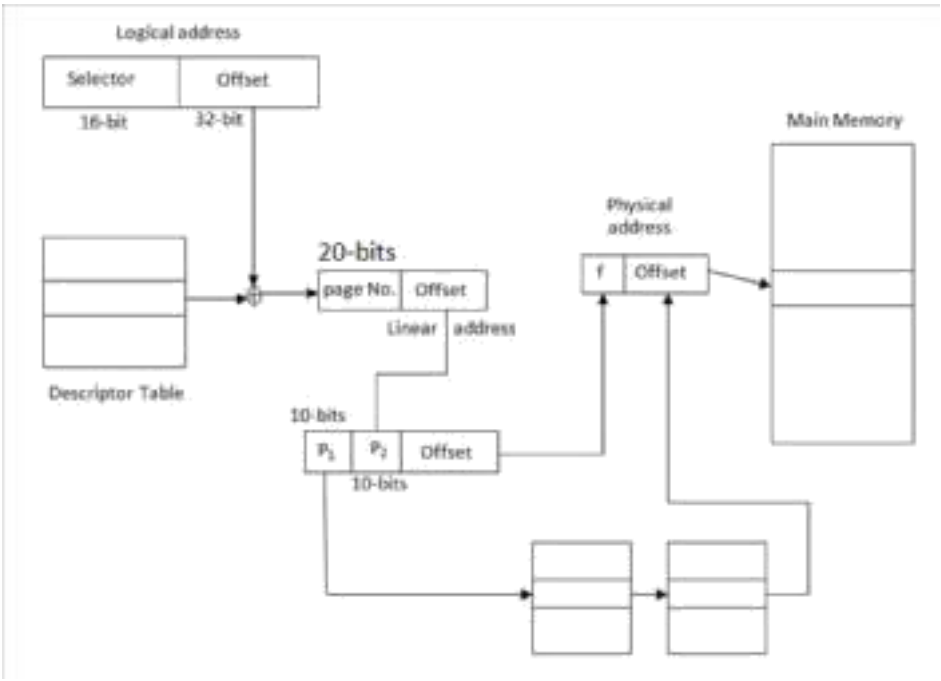
Information about the first partition is stored in a table called *LDT* (Local Descriptor Table). The information about the second partition is stored in a table called *GDT* (Global Descriptor Table).

Each entry in these tables occupies 8 bytes and contains full information about the segment base an limit.

The logical address space is a pair of <selector, offset>; Where *selector* is a 16-bit entry.

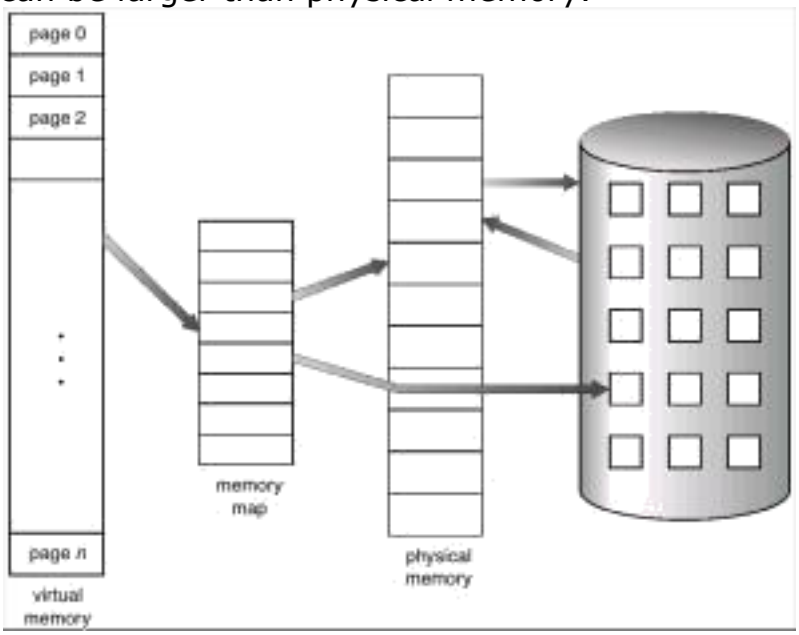
Where *S* is corresponding to segment no. and *G* indicates whether the segment is in LDT/GDT.

The 2-bits are used for protection *P*. The *offset* is 32-bit long and it denotes a particular word in the segment.
We use the following scheme to get the physical address from the logical address.



In the above diagram, the physical address is of 32-bits and it is formed as follows: Initially the segment no. and *G* are used to search in the descriptor table. The base and limit information of segment along with offset forms a linear address. This address is later transformed into physical address like in the *paging* scheme.

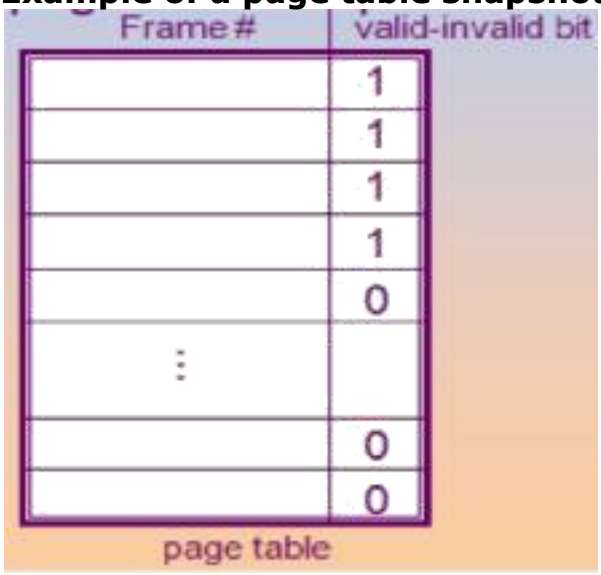
Q) Explain about virtual memory in detail?
A) Virtual memory is a technique that allows the execution of a process that may not be completely in memory. The main advantage of this scheme is that the programs can be larger than physical memory.



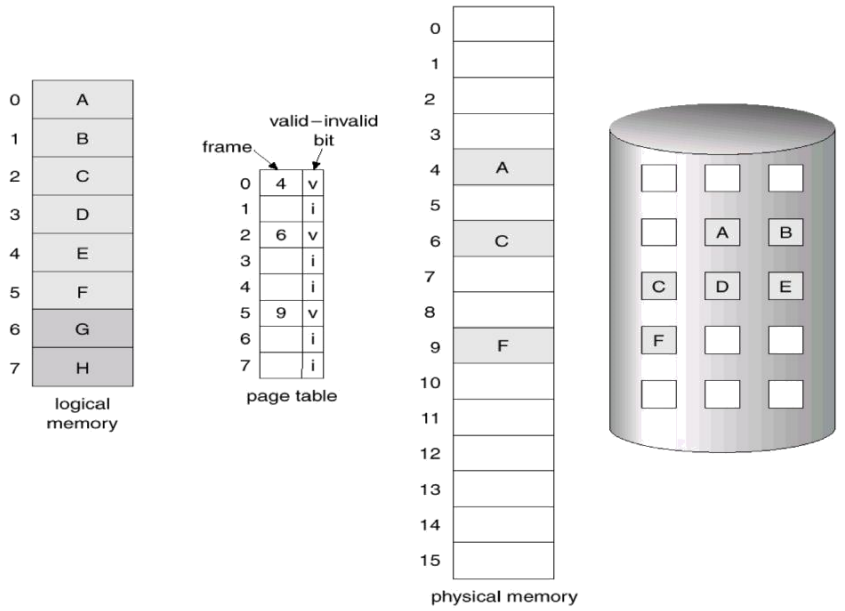
Here, an entire program is not loaded into memory at once. Instead of it, the program is divided into parts. The required part is loaded into main memory for the execution. This is not visible to the programmer. The programmer thinks that he has available lot main memory, but it is not true. Actually the system contains a small main memory and large amount of auxiliary(secondary) memory.

The address generated by CPU is known as logical address. The address seen by the memory unit is known as physical address. The set of all logical addresses are known as logical address space and set of physical addresses are known as physical address space. Logical addresses are also known as virtual address. Virtual memory can be implemented via:
a) Demand paging
b) Demand segmentation
Suppose we have 32 KB main memory and 1024 KB auxiliary memory. Then there are
215 words in main memory, 220 words in auxiliary memory. The physical address space is 215 and
logical address space is 220. The physical address contains 15-bits and logical address contains
20-bits.

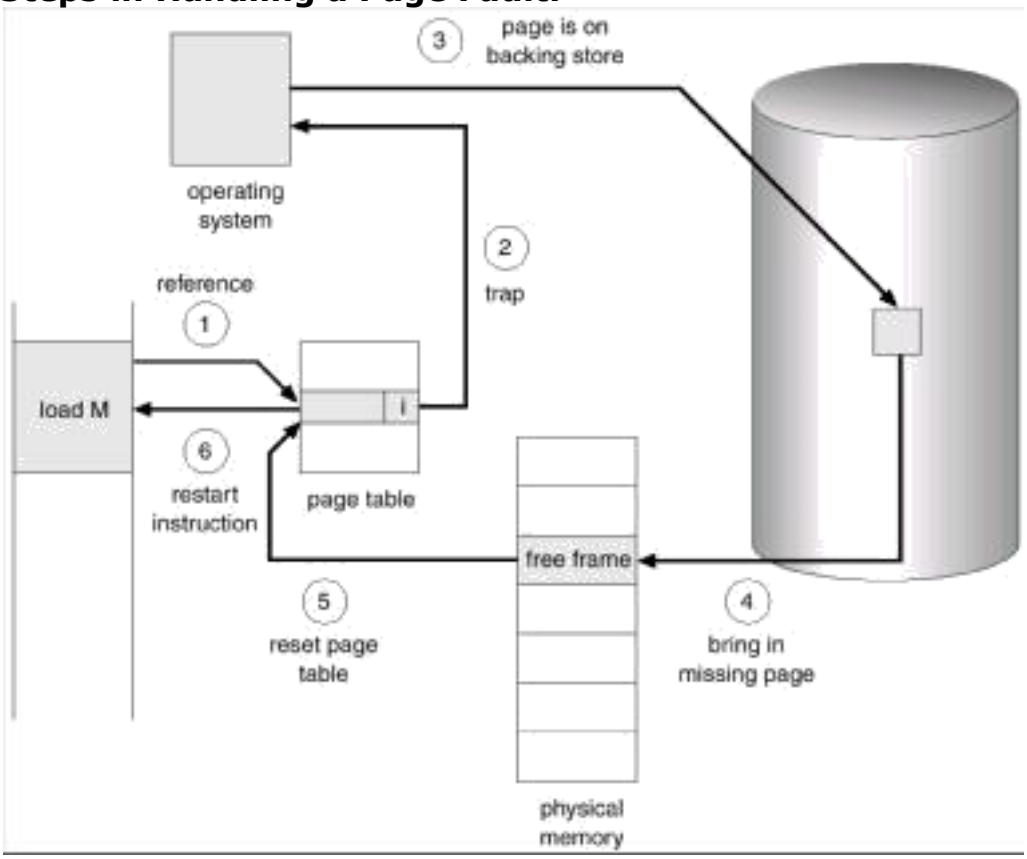
Q) Write about Demand Paging:
A) Demand Paging is similar to a paging with swapping. When we want to execute a process, we swap it into memory. Rather than swapping the entire process into memory, however we use ***lazy swapper***. The lazy swapper never swaps a page into memory unless that page will be needed. Instead of swapping whole pages, the required pages brings into the memory.
In demand paging each page table entry contains two fields: One is page index and valid /invalid bit. The bit is set to valid, this indicates that the associated page is both legal and in memory. If the bit is invalid, this indicates that the page is either not valid, but is currently on the disk.
Example of a page table snapshot:



During address translation, if valid– invalid bit in page table entry is 0 □□ page fault.
But what happens if the process this to use page that was not brought into memory.Access to a invalid page causes a ***page fault***.
"*In demand paging memory management scheme, when a required page is not in the main storage, the hardware raises a missing page interrupt, is popularly known as a* **page fault**".
Page Table When Some Pages Are Not in Main Memory:



Steps in Handling a Page Fault:



The procedure for handling this page fault is simple:

1) We check an internal table for this process to determine whether the reference was valid or invalid memory access.

2) If the reference was invalid, we terminate the process. If is valid, but we not brought into memory.

3) We find a free frame.

4) We schedule a disk operation to read the desired page into the newly allocated frame.

5) When the disk read is complete, we modify the internal table kept with the process and page table to indicate that the page is now in memory.

6) We restart the instruction that was interrupted by the illegal address trap.

Q) Explain about Performance of Demand Paging?

A)Demand paging is effect on the performance of a computer system. Let us that 'ma' is the memory access time for a computer systems range from 10 to 200 nano seconds. As long as we no page faults, the effective access time is equal to the memory access time. Let P be probability of a page fault. The effective access time = (1-P) * ma * P * page fault time. We must know how much time is needed to serve a page fault. A page fault causes the following sequence occur:

1) Trap to the operating system.
2) Save the user register and process state.
3) Determine the interrupt was page fault.
4) Check that the page reference was legal and determine the location of the page on the disk.
5) Issue a read from the disk to free frame.
a) Wait in a queue for this device until the load request is served.
b) Wait for the device seek / latency time.
c) Begin the transfer of a page to a free frame.
6) While waiting, allocate the CPU to some other user.
7) Interrupt from the disk.
8) Save the registers and process state for other user.
9) Determine that the interrupt was from the disk.
10) Correct the page table and other tables to show that the desired page is now in memory.
11) Wait for the CPU to be allocated to this process again.
12) Restore the user registers, process state and new page table then resume the interrupted instruction.

Q) Explain about page replacement algorithms in detail?

A) **Page Replacement Algorithms:**

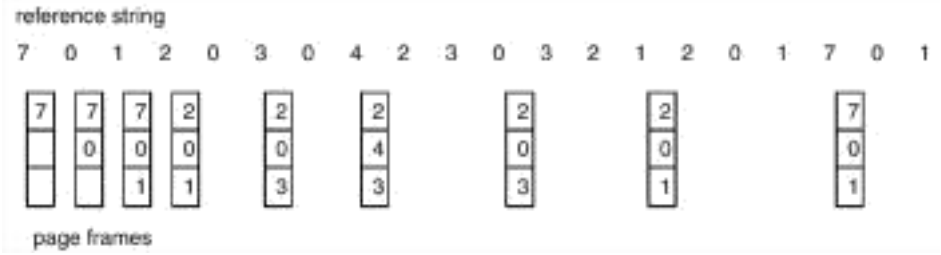
The page replacement algorithms are used to replace a page. The page replacement algorithms are

1) FIFO
2) Optimal page replacement
3) LRU

Let us assume that the reference string 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 2, 1, 2, 0, 1, 7, 0, 1 for a memory with 3 frames.

1) FIFO (First in First Out): The simplest page replacement algorithm is FIFO. When a page must be replaced, the oldest page is chosen. In FIFO, we replace the page at the head of the queue. When a page is brought into memory, we insert it at the tail of the queue.

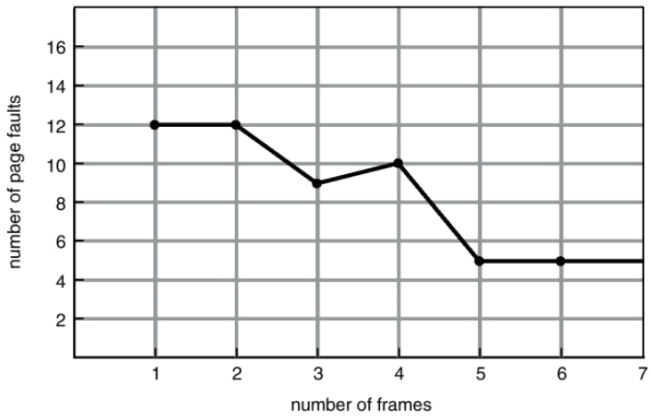
Ex:



The total number of page faults for the given reference string is 15.

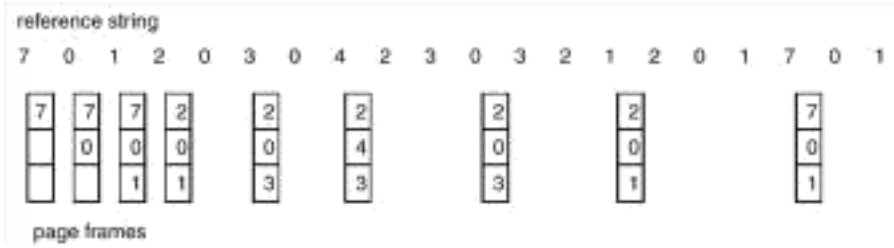
The FIFO algorithm is easy to understand. However the performance is not always good.

The FIFO algorithm suffers from Belady's anomaly i.e., the page faults rate increases as the number of allocated frames increases. This is explained in the following graph:



2) Optimal Algorithm: An optimal page replacement algorithm is the lowest page fault rate of all algorithms. This algorithm never suffers from Belady's anomaly. In optimal page replacement algorithm, *we replace a page that will not be used for the longest period of time*.

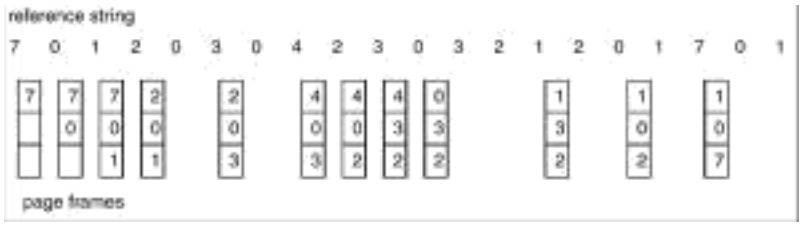
Ex:



The total number of page faults for the given reference string is 9.
Unfortunately, the optimal page replacement algorithm is difficult to implement because it future knowledge.

3) LRU (Least Recently Used) Algorithm: LRU algorithm replaces a page that is last use. LRU chooses a page that has not been used for the longest period of time. This algorithm looks backward rather than forward.

Ex:



The total number of page faults for the given reference string is 12.

UNIT-IV

FILE SYSTEM INTERFACE

1Q) Explain the concept of a file?

- **File:**"A file is a space on the disk where logically related information will be stored."**or**"A file is a named collection of related information that is recorded on secondary storage device."
- **File types:**A file has a certain defined structure according to its type. Different types of file are

a) Text File:A text file is a sequence of characters organized into lines (and possibly pages).

b) Source File: A source file is a sequence of subroutines and functions, each of which is further organized as declarations followed by executable statements.

c) Object File: An object file is a sequence of bytes organized into blocks understandable by the system's linker.

d) Executable File: An executable file is a series of code sections that the loader can bring into memory
and execute.

- **File Attributes:**A file is named, for the convenience of its human users and is referred to by its name.
- A name is usually a string of characters such as "example.c".
- Some systems differentiate between lower and upper case characters.
- A file has certain attributes, which are vary from one operating system to another. They are

Name – The symbolic file name is the only information kept in human-readable form

Type – The information is needed for systems that support different types

Location – This information is a pointer to file location on device

Size – The current file size (in bytes, words or blocks)

Protection – Access-control information controls who can do reading, writing, executing

- **Time**, **date**, **and user identification** – The can be useful data for protection, security, and usage monitoring Information about files are kept in the directory structure, which is maintained on the disk

File Operations: A file is an abstract data type.

- The operating system provides system calls to create, write, read, reposition, delete and truncate files.

1. **Create** – Two steps are required to create a file. First, space for the file system must be found for the file. Second, an entry for the new file must be made in the directory.

2. **Write** – To write the data into a file, a system call is required for both name and information to be written into the file. The system must keep a write a pointer to the location in the file where the next write is to take place. The write pointer must be updated whenever a write occurs.

3. **Read** – To read from a file, we use a system call that specifies both name of the file and where the next block of the file should be put. The system keeps a read pointer to the location in the file where the next read is to take place

4. **file seek** – Repositioning within file, is also known as a file seek

5. **Delete** – To delete a file, we search the directory for the named file.

Truncate – When the user wants the attributes of a file to remain the same, but wants to erase the contents of a file. This function can do this job.

6. **Open(Fi)** – search the directory structure on disk for entry Fi, and move the content of entry to memory
7. **Close (Fi)** – move the content of entry Fi in memory to directory structure on disk

- To open files, Several pieces of data are needed to manage open files:
- File pointer: pointer to last read/write location, per process that has the file open
- File-open count: counter of number of times a file is open – to allow removal of data from open-file table when last processes closes it.
- Disk location of the file: cache of data access information Access rights: per-process access mode information
- **Common File Types:**The following table explains common file types:

file type	usual extension	function
executable	exe, com, bin or none	read to run machine-language program
object	obj, o	compiled, machine language, not linked
source code	c, cc, java, pas, asm, a	source code in various languages
batch	bat, sh	commands to the command interpreter
text	txt, doc	textual data, documents
word processor	wp, tex, rrf, doc	various word-processor formats
library	lib, a, so, dll, mpeg, mov, rm	libraries of routines for programmers
print or view	arc, zip, tar	ASCII or binary file in a format for printing or viewing
archive	arc, zip, tar	related files grouped into one file, sometimes compressed, for archiving or storage
multimedia	mpeg, mov, rm	binary file containing audio or A/V information

2Q) Write about file access methods?

- **File Access Methods:** File access mechanism refers to the manner in which the records of a file may be accessed.
- There are several ways to access files –

a. Sequential access
b. Direct/Random access
c. Indexed sequential access

a. Sequential access:

- ✓ A sequential access is that in which the records are accessed in some sequence, i.e., the information in the file is processed in order, one record after the other.
- ✓ This access method is the most primitive one.
- ✓ Example: Compilers usually access files in this fashion.

b. Direct/Random access:

- Random access file organization provides, accessing the records directly.
- Each record has its own address on the file with by the help of which it can be directly accessed for reading or writing.
- The records need not be in any sequence within the file and they need not be in adjacent locations on the storage medium.

c. Indexed sequential access:

- This mechanism is built up on base of sequential access.
- An index is created for each file which contains pointers to various blocks.
- Index is searched sequentially and its pointer is used to access the file directly.

3Q) Write about Directory Structure?

A) Directory Structure:

➔ **Single-Level Directory:** In a single-level directory system, all the files are placed in one directory.

➔ This is very common on single-user OS's.

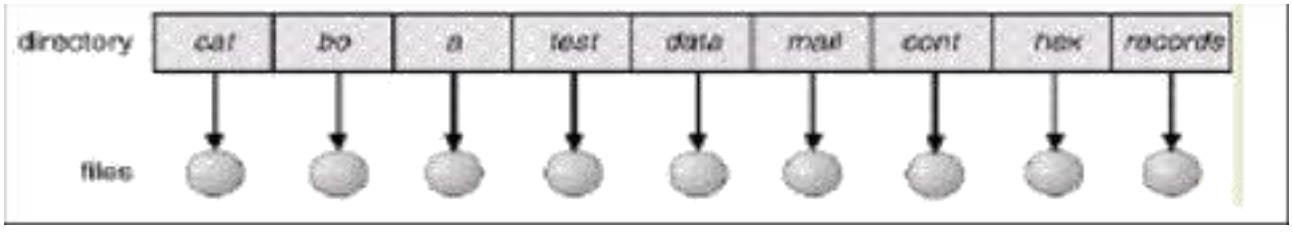
➔ A single-level directory has significant limitations, however, when the number of files increases or when there is more than one user.

➔ Since all files are in the same directory, they must have unique names.

➔ If there are two users who call their data file "test", then the unique name rule is violated.

➔ Although file names are generally selected to reflect the content of the file, they are often quite limited in length.

➔ Even with a single-user, as the number of files increases, it becomes difficult to remember the names of all the files in order to create only files with unique names.



Two-Level Directory:

➔ In the two-level directory system, the system maintains a master block that has one entry for each user.

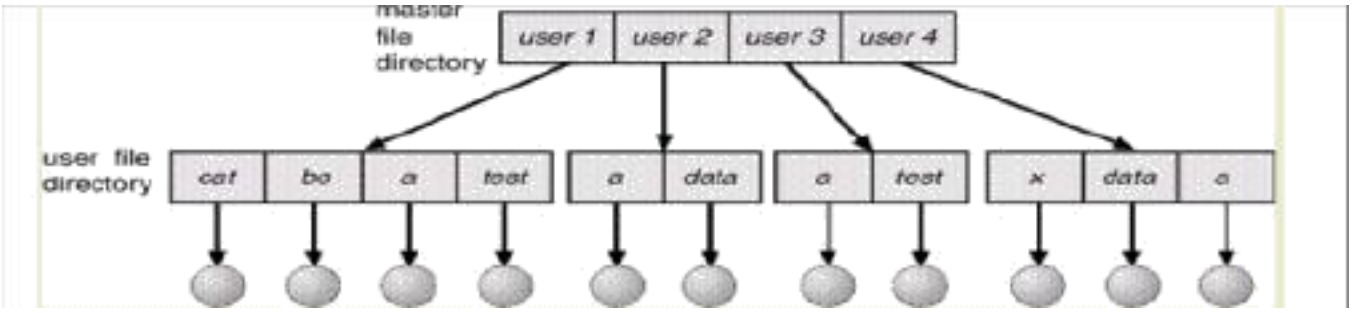
➔ This master block contains the addresses of the directory of the users.

➔ There are still problems with two-level directory structure.

➔ This structure effectively isolates one user from another.

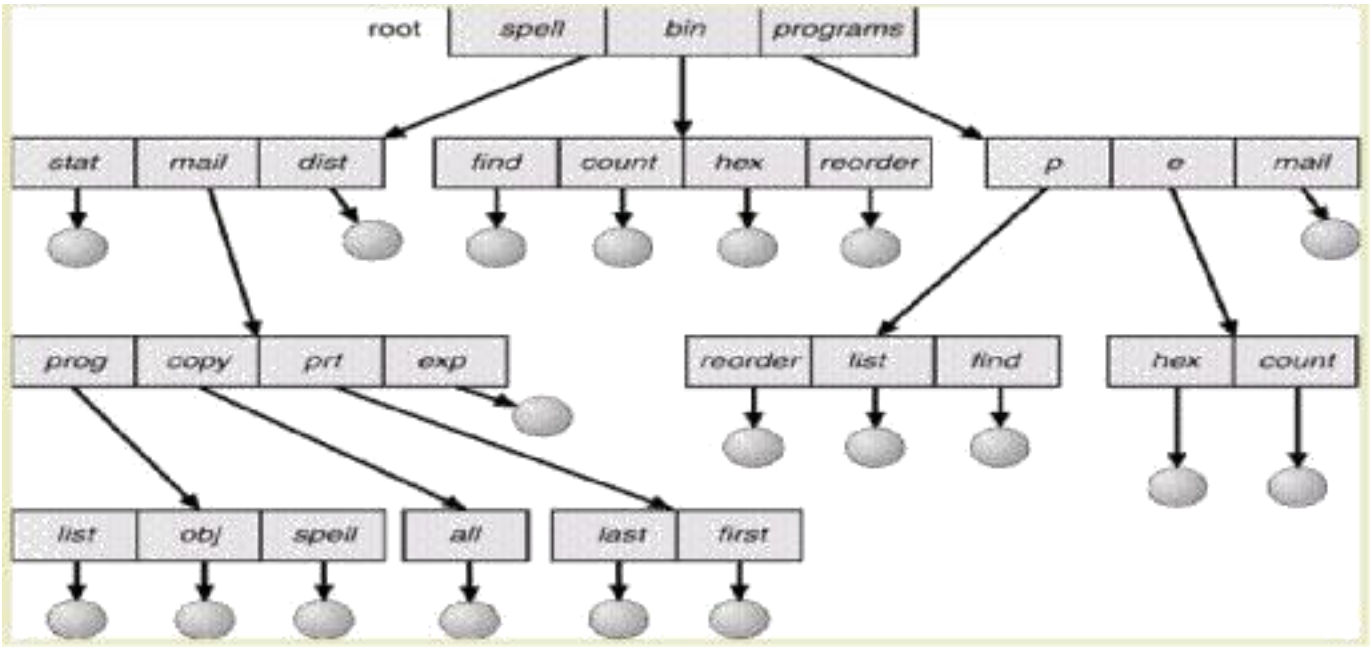
➔ This is an advantage when the users are completely independent, but a disadvantage when the users want to cooperate on some task and access files of other users.

Some systems simply do not allow local files to be accessed by other users.



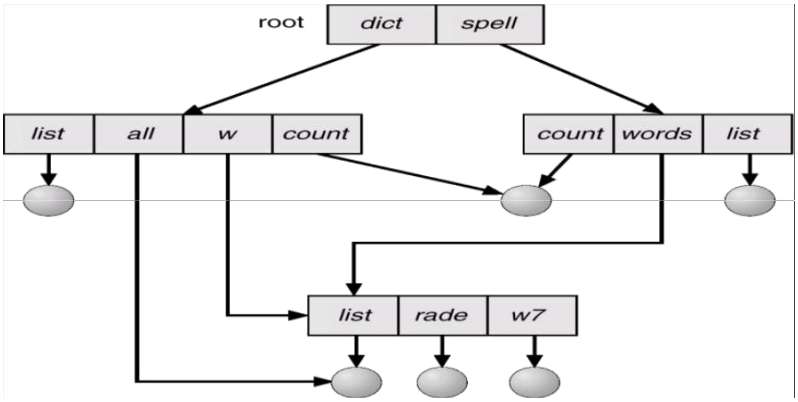
Tree-Structured Directories:

- ➔ In the tree-structured directory, the directory themselves are files.
- ➔ This leads to the possibility of having sub-directories that can contain files and sub-subdirectories.
- ➔ When a request is made to delete a directory, all of that directory's files and subdirectories are also to be deleted.



Acyclic-Graph Directories:

- ➔ A tree structure prohibits the sharing of files or directories.
- ➔ An acyclic graph, which is a graph with no cycles, allows directories to have shared subdirectories and files.
- ➔ This is explained in the following diagram:

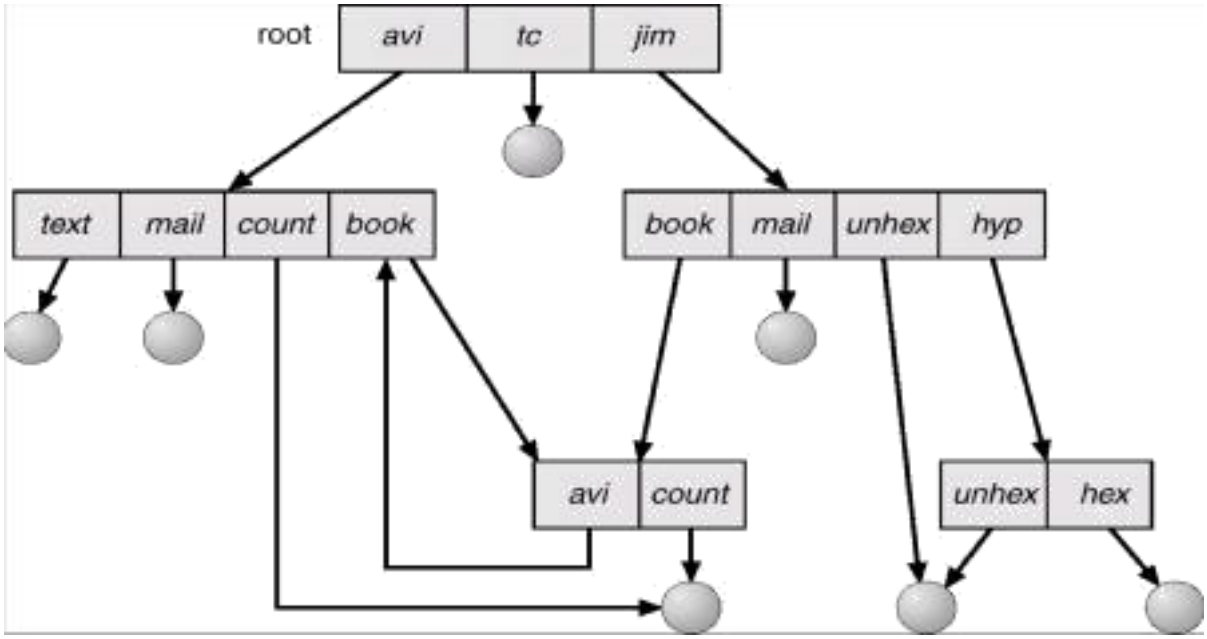


- ➔ In the above diagram count is shared.

➔ The primary advantage of acyclic graph is the relative simplicity of the algorithms to traverse the graph and determine when there are no more references to file.

General Graph Directory:

➔ This allows only links to file not subdirectories.
➔ Garbage-collection is necessary only because of possible cycles in the graph.
➔ The difficulty is to avoid cycles as new links are added to the structure. This is explained in the following diagram:



4Q) Explain about disk space Allocation Methods in file system implementation?

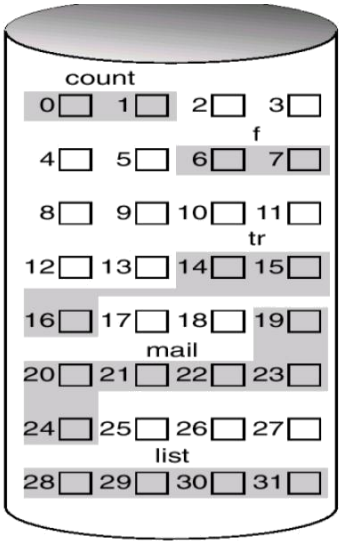
Allocation Methods:

An allocation method refers to how disk blocks are allocated for files. Three major methods of allocating disk space are:

a) Contiguous allocation
b) Linked allocation
c) Indexed allocation

a) Contiguous allocation:

➔ The contiguous allocation method requires each file to occupy a set of contiguous blocks on the disk.
➔ This method is simple – only starting location (block # (no.)) and length (number of blocks) are required.
➔ Both sequential and random access can be supported by contiguous allocation.
➔ One difficulty with contiguous allocation is finding space for a new file.
➔ The contiguous disk-space-allocation problem can be seen to be particular application of the general dynamic storage-allocation problem (Wasteful of space).
➔ Stored files cannot grow in future.
➔ The contiguous allocation of disk space is explained in the following diagram.



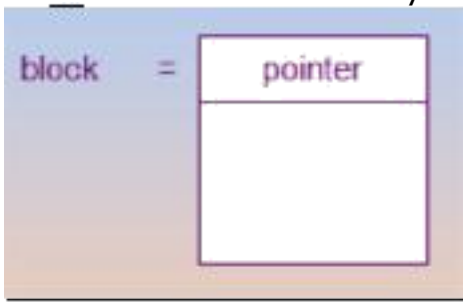
directory

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

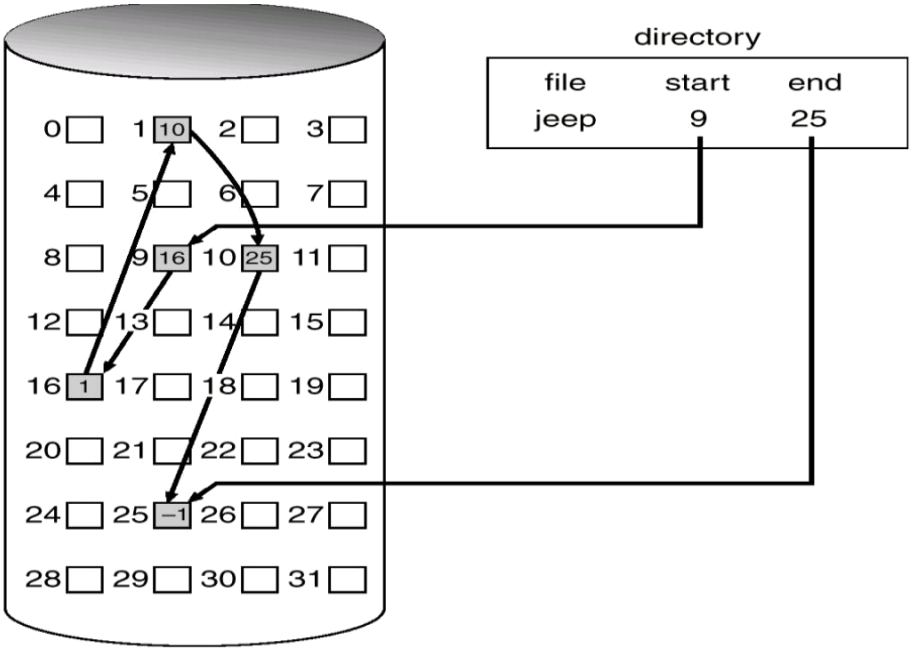
➔ This algorithm suffers from the external fragmentation.
➔ As files are allocated and deleted, the free space is broken into little pieces.
➔ A file that grows slowly over a long period (months or years) must be allocated enough space for its final size, even though much of that space may be unused for a long time.
➔ The file, therefore, has a large amount of internal fragmentation.
➔ To avoid several drawbacks of contiguous allocation method, many newer file systems use a modified contiguous allocation scheme.
➔ Extent-based file systems allocate disk blocks in **extents**.
➔ An **extent** is a contiguous block of disks.
➔ Extents are allocated for file allocation.
➔ A file consists of one or more extents.

b) Linked allocation:

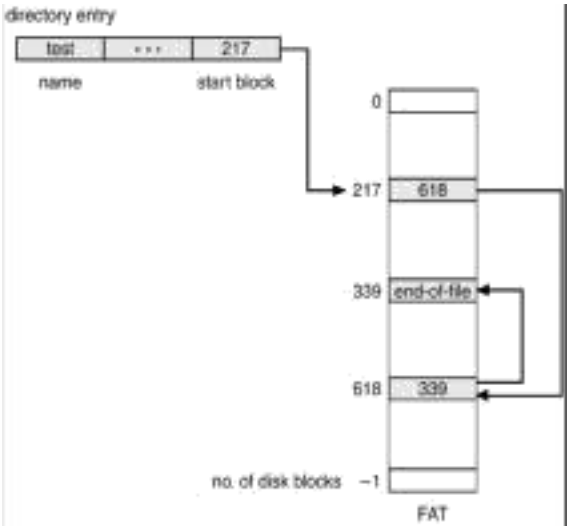
➔ Linked allocation solves all problems of contiguous allocation.
➔ With linked allocation, each file is a linked list of disk blocks.
➔ The disk blocks may be scattered anywhere on the disk.



➔ This scheme is simple.
➔ It needs only starting address.
➔ For example, a file of five blocks might start at block 9, continue at block 16, then block 1, block 10 and finally block 25.
➔ This is explained in the following diagram:



➔ Free-space management system – no waste of space
No random access
Mapping

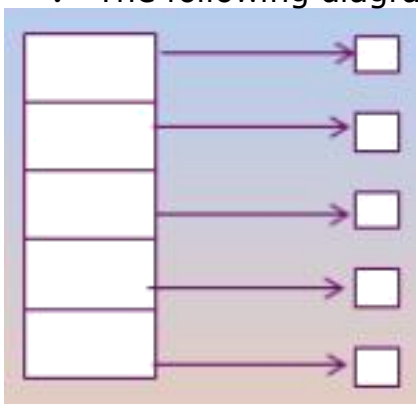


- ➢ The table has one entry for each disk block and is indexed by block number.
- ➢ The FAT is used much as is a linked list.
- ➢ The directory contains the block number of the first block of the file.
- ➢ The table entry indexed by that block number then contains the block number of the next block in the file.
- ➢ This chain continues until the last block, which has a special end-of-file value as the table entry.
- ➢ Unused blocks are indicated by a 0 (zero) table value.

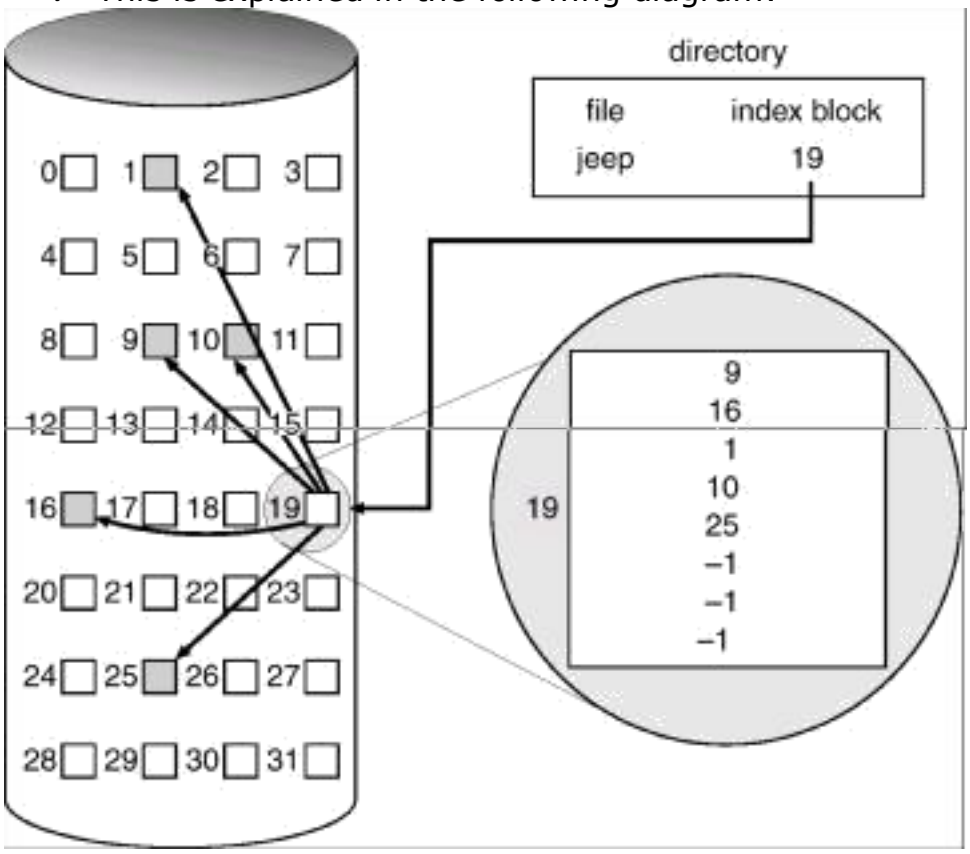
c) Indexed Allocation:

- ✓ Linked allocation solves the external fragmentation problem and size declaration problems of contiguous allocation.
- ✓ But in the absence of FAT, linked allocation can not support efficient direct access because the pointers to the blocks are scattered with the blocks themselves all over the disk and need to be retrieved in order.
- ✓ Indexed allocation solves the problem in linked allocation by bringing all pointers together into one location is known as **index block**.

✓ The following diagram shows the logical view of the index table:



✓ Each file has its own index block, which is an array of disk-block addresses.
✓ The i th entry in the index block points to the i th block of the file.
✓ The directory entry contains the address of the index block.
✓ This is explained in the following diagram:



Indexed allocation supports direct access without suffering from external fragmentation. This method suffers from wasted space.

5Q) Write about protection and security?
A)
Protection is one of the major role in the OS and file system interface. The protection can be provided in many ways. For example, these are represented as follows.
Types of access: it provides a LIMITEED FILE ACCESS TO THE USER. Several types of operations may be controlled by file system interface is as follows

- Read: read from the file
- Write: write or rewrite the file
- Execute: load the file into memory and execute
- Append: write the new information at the end of the file
- Delete: delete the file and releases the memory
- List: list the name and attributes of the file

Access list and groups:

In the file system interface, every file may requires 3 persons. They are 1.owner 2.group 3.others

Owner – the user who created the file is the owner

Group – a set of users who are sharing the file with in the work group

Universe/others – all other users in the file system interface

In the MS-Dos operating system the access rights are applied though "attriB command is as follows

C:\\ Attrib +751<fIle name>

In the UNIX operating system the access rightes are applied though "chmod" command is as follows

Chmod +751<file name>

➔ The attributes read contains 4,write contains 2,and execute contains 1

➔ Hence in the above example the owner contains read and execute permission, group contains read and execute permissions and others contain execute permission only**.**

Security:

Security requires not only an adequate protection system, but also consideration of the external environment within which the system operates. The system protect it from

i) unauthorized access.
ii) Malicious (harmful) modification or destruction
iii) accidental introduction of inconsistency.

It is easier to protect against accidental than malicious misuse.

Authentication:

➔ A major security problem for operating systems is **authentication**. Generally, authentication is based on one or more of three items: user possession (a key or card), user knowledge (a user_id and password) or user attributes (fingerprint, retina pattern or signature).

➔ ¨ Authentication means Verifying the identity of another entity

– Computer authenticating to another computer
– Person authenticating to a local computer
– Person authenticating to a remote computer

Passwords:

➔ The most common authentication of a user's identity is via a user **password**.

➔ In password authentication, users are authenticated by the password.

- User has a secret password.
- System checks it to authenticate the user.
- Vulnerable to eavesdropping when password is communicated from user to system.
- The big problem with password-based authentication is eavesdropping.

Vulnerabilities :

1) External Disclosure – Password becomes known to an unauthorized person by a means outside normal network or system operation. Includes storing passwords in unprotected files or posting it in an unsecured place.

2) Password Guessing – Results from very short passwords, not changing passwords often, allowing passwords which are common words or proper names, and using default passwords.
3) Live Eavesdropping – Results from allowing passwords to transmit in an unprotected manner.

4)Verifier Compromise – Occurs when the verifier's password file is compromised via an internal attack.

5) Replay – Recording and later replaying of a legitimate authentication exchange.

There are various **Techniques** to handle the passwords safely. They are

1) One-Time Passwords
2) Challenge-Response
3) On-Line Cryptographic Servers
4) Off-Line Cryptographic Servers
5) Use of Non-Repeating Values
6) Mutual Authentication Protocols

6Q) Write about free space management?

In the file system, it is necessary to reuse the space from deleted files for new files. In the file system the disk maintains the free space list. The free space list is implemented in 4 ways. They are

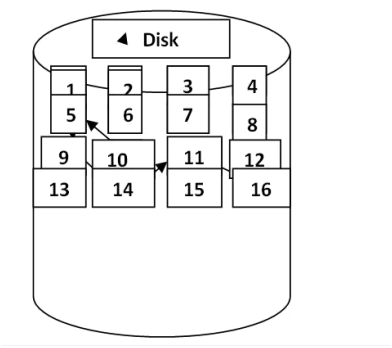
1. Bit Vector
2.Linked List
3. Grouping
4. Counting

Bit vector:

- ➔ It is also called as Bit map.In this representation ,each block is represented by either '1' or 'o'.
- ➔ If the block is free then it is representd by '1'.other wise Zero
- ➔ For example conside a disk where the blocks are 2,3,4,8,9,13 are free.
- ➔ then the bit vector is represented as follows.00111000110001
- ➔ The main advantage of this representation is relatively simple and efficient.
- ➔ In this, the calculation of the block no.is (no of Bits per word)×(no of zero values words)+off set of first one bit.

Linked List:

- ➔ In this representation ,to link together all the free disk blocks with the help of a pointer the linked list is implemented.In the linked list,the first block contains a pointer to the next n disk block and so on.
- ➔ For example,It can be representes as follows



Grouping:

➔ In this representation ,every block contains n free blocks.
➔ the last block contains the address of another n free blocks.the importance of this implementation is used in the large no of free blocks

➔ In this representation ,several contiguous blocks may be allocated or free simultaneously .
➔ In this representation a disk block maintains the list of n free contiguous blocks.
➔ Each entry requires more space than a simple disk address and the count is generally greater than 1.

7Q) Write about file system structure?

- The functionality of the file is used to store large amount of information.
- In the file system implementation, the layered approach is represented as follows.

Application programs
|
Logical file system
|
File organisation module
|
Basic file system
|
I/O control
|
Devices

➢ The lowest level contains the I/O control consists of device drivers and interrupt handlers to transfer the information between the memory and disk system.

➢ The file organisation module defines the information about the files, logical blocks as well as the physical blocks. It also include free space manager.
➢ The logical file system uses the directory structure to provide the file organisation module. It is also responsible for protection and security.
➢ Application programs are written by the user with the help of languages like c ,c++,java,.., e.t.c. in the application programs, they are various types of files. For example, they are text files, binary files, indexed files, random files, executable files, e.t.c..

8Q) Explain about Thrashing?

Thrashing:

✓ It is defined as "a situation in which a program cause page faults for every few instructions.
✓ In multiprogramming CPU utilization increases in such a case the CPU utilization is drops sharply.
✓ It can be represented as follows

9Q) Explain disk scheduling with its algorithms?

1) The operating system is responsible for using hardware efficiently — for the disk drives, this means having a fast access time and disk bandwidth.

2) Access time has two major components

a) Seek time is the time for the disk are to move the heads to the cylinder containingthedesiredsector.

b) Rotational latency is the additional time waiting for the disk to rotate the desiredsectortothediskhead.

3)Minimizeseektime

4)Seektime□ seekdistance

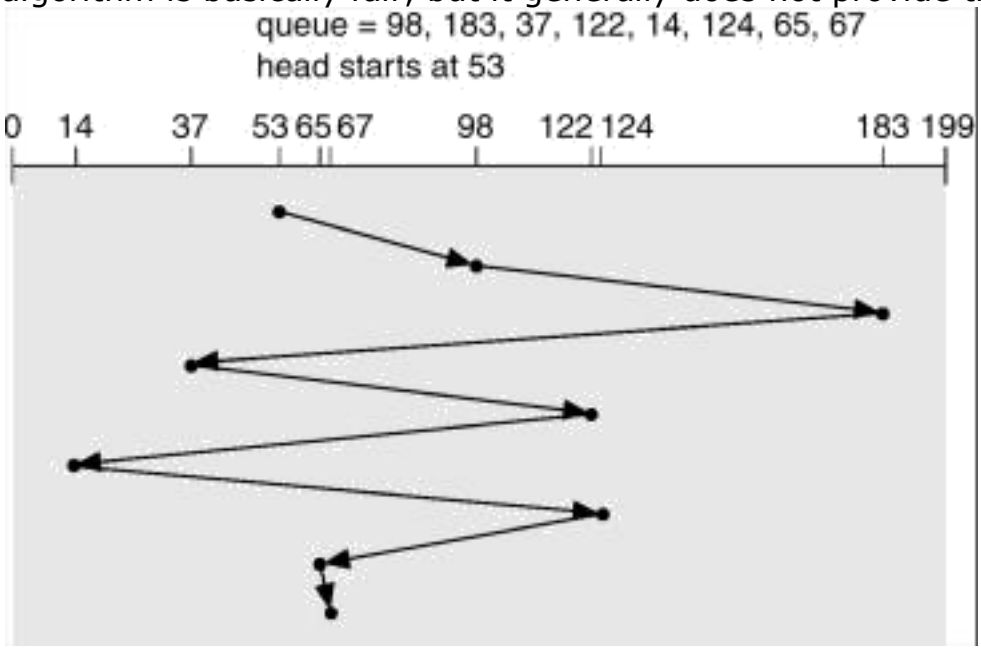
5) Disk bandwidth is the total number of bytes transferred, divided by the total time between the first request for service and the completion of the last transfer.

6) Several algorithms exist to schedule the servicing of disk I/O requests.

7) We illustrate them with a request queue (0-199).
98, 183, 37, 122, 14, 124, 65, 67
Head pointer 53

1. FCFS Scheduling:

The simplest form of disk scheduling is First-come, First-served (FCFS). This algorithm is basically fair, but it generally does not provide the fastest service.



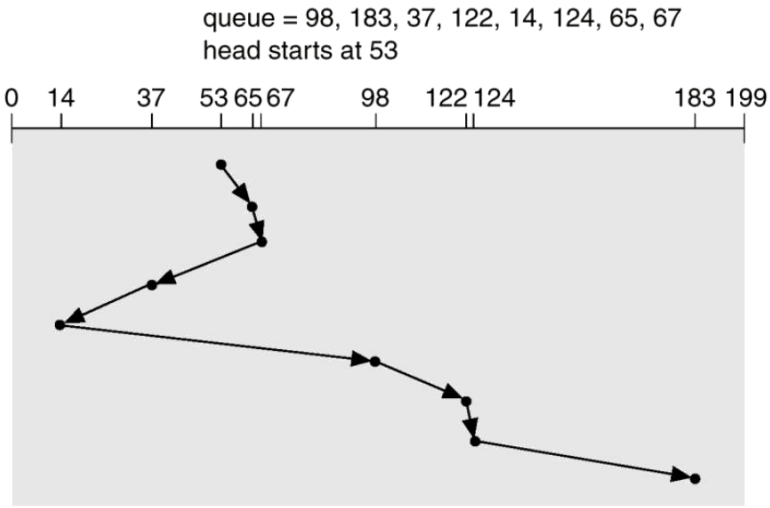
To the above example, total head movement of 640 cylinders.

2. SSTF (Shortest Seek Time First) Scheduling:

a) SSTF selects the request with the minimum seek time from the current head position.

b) SSTF scheduling is a form of SJF scheduling; may cause starvation of some requests.

Ex: This shows total head movement of 236 cylinders



3) SCAN Scheduling:

a) In the SCAN algorithm, the disk arm starts at one end of the disk, and moves toward the other end, servicing requests until it gets to the other end of the disk, where the head movement is reversed and servicing continues.

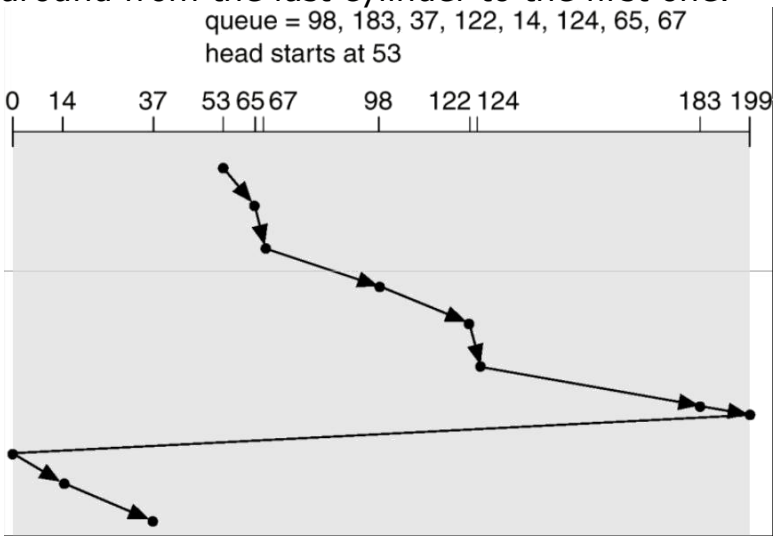
b) Sometimes this algorithm is also called the elevator algorithm because the disk arm behaves just like elevator in a building, first servicing all the requests going up and then reversing to service requests the other way.
Ex: This algorithm shows total head movement of 208 cylinders.

4) C-SCAN Scheduling:

a) This scheduling algorithm provides a more uniform wait time than SCAN.

b) The head moves from one end of the disk to the other, servicing requests along the way. When it reaches the other end, however, it immediately returns to the beginning of the disk, without servicing any requests on the return trip.

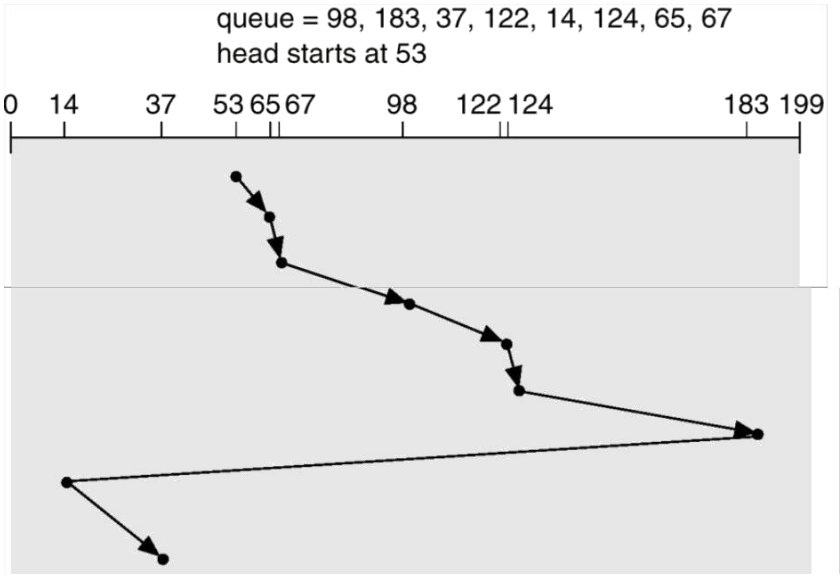
c) The C-SCAN scheduling algorithm treats the cylinders as a circular list that wraps around from the last cylinder to the first one.



5) C-LOOK Scheduling:

a) This C-LOOK is the version of C-SCAN

b) More commonly, the arm only goes as far as the last request in each direction, then reverses direction immediately, without first going all the way to the end of the disk.
Ex:



UNIT-V
DEADLOCKS

1Q)Explain about deadlock in detail?

Deadlock:

- When several processes compete for a finite number of resources, a situation may arise where a process requests a resource and the resource is not available at that time, in that time the process enters a wait state.
- It may happen that waiting processes will never again change state because the resources that they have requested are held by other waiting processes.
- This situation is called a **deadlock**.
- Deadlock can arise if **four** conditions hold simultaneously.

1) Mutual exclusion: only one process at a time can use a resource.

2) Hold and wait: a process holding at least one resource is waiting to acquire additional resources held by other processes.

3) No preemption: a resource can be released only voluntarily by the process holding it, after that process has completed its task.

4) Circular wait: there exists a set {P0, P1, …, P0} of waiting processes such that P0 is waiting for a resource that is held by P1, P1 is waiting for a resource that is held by P2, …, Pn–1 is waiting for a resource that is held by Pn, and P0 is waiting for a resource that is held by P0.

2Q) Write about Resource-Allocation Graph ?

Resource-Allocation Graph:

- Deadlocks can be described in terms of a directed graph called a system resourceallocation graph.
- This graph consists of a set of vertices V and a set of edges E.

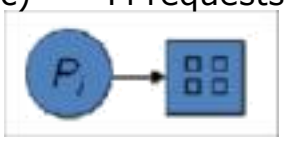
1) V is partitioned into two types:
a) P = {P1, P2, …, Pn}, the set consisting of all the processes in the system.
b) R = {R1, R2, …, Rm}, the set consisting of all resource types in the system.
2) request edge – directed edge P1 ® Rj
3) assignment edge – directed edge Rj ® Pi

The following symbols are used in the resource allocation graph :

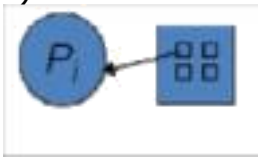
a) Process

b) Resource Type with 4 instances

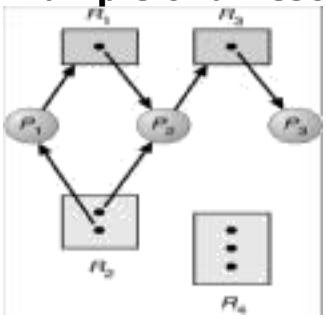
c) Pi requests instance of Rj



d) Pi is holding an instance of Rj



Example of a Resource Allocation Graph :



The above graph is describing the following situation :
1) The sets P, R and E are
: P = { P1, P2, P3 }
R = { R1, R2, R3 }
E = { P1 -> R1, P2 -> R3, R1 -> P2, R2 -> P2, R2 -> P1, R3 -> P3 }

2) Resource instances :
- One instance of resource type R1
- Two instances of resources type R2
- One instance of resource type R3
- Three instances of resource type R4

3) Process states :
- Process P1 is holding an instance of resource type R2 and is waiting for an instance of resource type R1.
- Process P2 is holding an instance of R1 and R2 and is waiting for an instance of resource type R3.
- Process P3 is holding an instance of R3.
If the graph consists **no cycles**, then no process in the system is deadlocked. If the graph **contains a cycle**, then a deadlock may exist.

3Q) Explain about Deadlock Prevention?

- ✓ Deadlock prevention is a set of methods for ensuring that at least one of the necessary conditions cannot hold.

a) Mutual Exclusion:
- ✓ The mutual exclusion condition must hold for non sharable resources. For example, a printer can not be accessed by more than one process at a time.
- ✓ But Read-only files are good example for sharable resources and can not be involved in deadlock.
- ✓ A process never needs to wait for sharable resources.
- ✓ Hence we can not prevent deadlocks by denying the mutual exclusion condition because some resources are basically non sharable.

b) Hold and Wait:
- ✓ To ensure that the hold and wait condition never occurs in the system. There are two protocols to break the hold and wait.

Protocol 1 – Each process to request and be allocated all its resources before it begins execution.

Protocol 2 – A process request some resources only when the process has none. A process may request some resources and use them. Before it can request any additional resources, it must release all the resources which are hold by it.

- ✓ By using an example we differentiate between these two protocols: A process that copies data from a tape drive to a disk file, sorts the disk file and then prints the results to a printer.
- ✓ According to protocol 1, the process must request all the resources such as tape drive, disk drive and printer before staring its execution.
- ✓ If they are available, they will be allocated to it. Otherwise, the process has to wait to start its execution until getting the all resources.
- ✓ Though all resources are allocated initially, the process can use the printer at the end.
- ✓ According to protocol 2, the process can request initially only the tape drive and disk file.
- ✓ It copies from the tape drive from disk and release both. Again the process request the disk file and printer.
- ✓ After copying the disk file to the printer, it releases two resources and terminates

Disadvantages:
These two protocols have two main disadvantages.

1) Resource utilization may be low – many of the resources may be allocated but unused for a long period. In the example, the process release the tape drive and disk file, and then again request the disk file printer. If they are available at that time we can allocate them. Otherwise the process must wait.

2) Starvation is possible – A process that needs several popular resources may have to wait indefinitely because at least one of the resources that it needs is always allocated to some other process.

c) No Preemption: The third necessary condition is that there is no preemption of resources that have already been allocated. To ensure that this condition does not hold, we can use the following two protocols:

Protocol 1 –

- ✓ If a process is holding some resources and requests another resource that cannot be immediately allocated to it.
- ✓ All the resources currently being are preempted. The preempted resources are added to the list of resources for which the process is waiting.
- ✓ The process will be restarted when it regain its old resources as well as the new ones.

Protocol 2 –

- ✓ A process requests some resources, we first check whether they are available. If they are available, we can allocate them.
- ✓ If they are not available, first we check whether they are allocated to some other process that is waiting for additional resources.
- ✓ If so, preempt them from that process and allocated to the requesting process.

- ✓ If they are not available, the requesting process must wait.
- ✓ It may be happen that while it is waiting, some of its existing resources may be preempted due to the requesting of some other process.

d) Circular Wait: The fourth and final condition for deadlocks is the circular-wait. To ensure that this condition never holds in the system. To do so, each process requests resources in an increasing order of enumeration. Let R = { R1, R2, R3, …….., Rm } be the set of resource types. We assign to each resource type unique integer number, which allows us to compare two resources and to determine whether one precedes another in our ordering.

We define a one-to-one function F: R->N, Where N is the set of natural numbers.
For example, F (tape drive) =1
F (disk drive) =5
F (printer) =12

- ✓ To prevent the deadlocks we can follow the following protocols:

Protocol 1-

- ✓ Each process can request resources in an increasing order only. A process can request the resources if and only if F (Rj) > F (Ri).
- ✓ If several instances of the same resource type are needed, as single request for all of them must be issued.

Protocol 2 –

- ✓ Whenever a process requests an instance of resource type Rj , it has released any resources Ri such that F (Ri) ³ F (Rj).
- ✓ If these two protocols are used, then the circular wait condition cannot be hold.
- ✓ Let the set of processes { P0, P1, ……., Pn }where Pi is waiting for a resource Ri , which is held by Pi+1.
- ✓ Process Pi+1 is holding resource Ri, while requesting resource Ri+1, we must have F (Ri) < F (Ri+1), for all i.
- ✓ But this condition means that F (R0) < F (R1) < …….. F (Rn) < F (R0). By transitivity, F (R0) < F (R0), which is impossible.
- ✓ Therefore, there can be no circular wait.

If we follow these protocols to fail one of the 4 conditions for the deadlock, we can easily prevent deadlock.

4Q) Explain about Deadlock Avoidance?

- ✓ It requires that the operating system be given in advance additional information concerning which resources a process will request.
- ✓ And use during its lifetime.
- ✓ With this knowledge we can make a decision whether the current process request can be satisfied or delayed.
- ✓ A deadlock avoidance algorithm dynamically examines the resource-allocation state to ensure that a circular wait condition can never exist.
- ✓ The number of available and allocated resources and the maximum demands of the processes define the resource allocation state.

- ✓ A state is safe if the system can allocate the resources to each process (up its maximum) in some order and still avoid a deadlock.

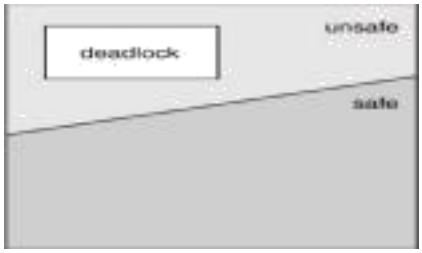
- ✓ A system is in a safe state only there exists a **safe sequence**.
- ✓ A sequence of processes < P1, P2 ……., Pn> is a safe sequence for the current allocation state if, for each Pi, the resources that Pi can still request can be satisfied by the currently available resources plus the resources held by all the Pj, with j < i. In this situation,

a) If Pi resource needs are not immediately available, then Pi can wait until all Pj have finished.

b) When Pj is finished, Pi can obtain needed resources, execute, and return allocated resources, and terminate.

c) When Pi terminates, Pi+1 can obtain its needed resources, and so on.
If there is no safe sequence, the system is said to be **unsafe**.

- ✓ **A safe sate is not a deadlock state. Conversely, a deadlock state is an unsafe state.**
- ✓ **Not all unsafe states are deadlocks**. In an unsafe state, the operating system cannot prevent processes from requesting resources such that a deadlock occurs.
- ✓ The behavior of the processes controls unsafe state.



Ex: Suppose a system with 12 magnetic tape drives and 3 processes: P0, P1 and P2 . Process

P0 requires 10 tape drives, process P1 requires 4 and process P2 requires 9 tape drives.Suppose at time t0, Process P0 is holding 5 tape drives, process P1 is holding 2 tape drives and process P2 is holding 2 tape drives. Thus there are 3 free tape drives.

Process	Maximum Needs	Current Needs
P0	10	5
P1	4	2
P2	9	2

At time t0, the system is in safe state. The sequence <P1, P0, P2> satisfies the safety condition.

Safety Algorithm:

This algorithm is used to find out whether or not the system is in a safe state.
This algorithm consists the following steps:

1. Let Work and Finish be vectors of length m and n, respectively. Initialize:
Work = Available
Finish [i] = false for i - 1,3, …, n.2.

2. Find and i such that both:

(a) Finish [i] = false (b) Needi Work
If no such i exists, go to step 4.

3. Work = Work +
Allocationi Finish[i] = true
go to step 2.

4. If Finish [i] == true for all i, then the system is in a safe state.

5Q) Write about Deadlock Detection?

- ✓ Sometimes we have not available the protocols which will ensure "NO Deadlock " in our system.
- ✓ In this case, deadlock may arrive into our system.
- ✓ Then we have to use deadlock detection & recovery procedures.
- ✓ Deadlock is an algorithm which is invoked periodically & it determines whether deadlock has occurred in our system or not.
- ✓ If the algorithm detects a deadlock in the system, we need to use the recovery procedure.
- ✓ The following is the deadlock detection algorithm.
- ✓ Detection algorithm uses information about processes, resource allocation & outstanding requests from each process and availability of resources.
- ✓ Deadlock detection algorithm uses the following data structures:

Available:
A vector of length m indicates the number of available resources of each type.

Allocation:
An n x m matrix defines the number of resources of each type currently allocated to each process.

Request:
An n x m matrix indicates the current request of each process. If Request [i, j] = k, then process Pi is requesting k more instances of resource type. Rj.

Algorithm:

1. Let Work and Finish be vectors of length m and n, respectively Initialize:
(a) Work = Available
(b) For i = 1,2, …, n, if Allocationi ¹ 0, then
Finish[i] = false; otherwise, Finish[i] = true.

2. Find an index i such that both:
(a) Finish[i] == false
(b) Requesti £ Work
If no such i exists, go to step 4.

3. Work = Work +
Allocationi Finish[i] = true
go to step 2.

4. If Finish[i] == false, for some i, 1 £ i £ n, then the system is in deadlock state. Moreover, if Finish[i] == false, then Pi is deadlocked.

This algorithm requires an order of O(m x n2) operations to detect whether the system is in deadlocked state.
Ex: Consider a system with Five processes P0 through P4; three resource types A (7 instances), B (2 instances), and C (6 instances). Snapshot at time T0:

The sequence <P0, P2, P3, P1, P4> will result in Finish[i] = true for all i.
Suppose P2 requests an additional instance of type C.

	Request
	A B C
P_0	0 0 0
P_1	2 0 1
P_2	0 0 1
P_3	1 0 0
P_4	0 0 2

The Deadlock exists, consisting of processes P1, P2, P3, and P4.

6Q) Explain how to Recovery from Deadlock?

- After the deadlock detection algorithm determines that a deadlock exists, and then two possibilities exist.
- One is that a deadlock has occurred and the user needs to deal with deadlock.
- The other is to let the system recover from the deadlock.
- There are two options for breaking a deadlock.
- One solution is to abort one or more processes to break the circular wait (i.e., to kill one of its processes).
- The second option is to preempt some resources from one or more of the deadlock processes.

1) Process Termination: To eliminate the deadlocks by aborting a process, we use one of the two methods. In both methods, the system reclaims all resources allocated to the terminated process.

a) Abort all deadlock processes: This method will break the deadlock cycle. But at a great expense, these processes may have computed for a long of time and the results of these partial computations must be discarded and may be recommended later.

b) Abort one process at a time until the dead cycle is eliminated: This method incurs considerable overhead , since after each process is aborted, a deadlock-detection algorithm must be invoked to determine whether any processes are still deadlocked.

2) Resume Preemption: In this method, we successively preempt some resources from processes and give these resources to other processes until the deadlock cycle is broken.
Here, we need to consider 3 things:

a) Selection of a victim: In this, we need to find out which resources and processes are to be preempted. In process termination, we determine the order of preemption to minimize the cost. Cost factors may like number of resources a

deadlock process is holding and amount of time a deadlock process has consumed during its execution.

b) Rollback: We must rollback the process to some safe state and restart it from that state. It is difficult to determine the safe state. So, the solution is a rollback.

c) Starvation: When victim selection is made usually on cost factors, it may happen that the same process is always picked as a victim. As a result, starvation occurs. So we must ensure that a victim is selected for a small / finite number of times. The common solution is to include the number of rollbacks in the cost factor.