Lecture Notes

On

OPERATING SYSTEM AND SYSTEM PROGRAMMING

PREPARED BY:

SUBHASISH DAS MOHAPATRA, ASST.PROF., (CSE)CVRP

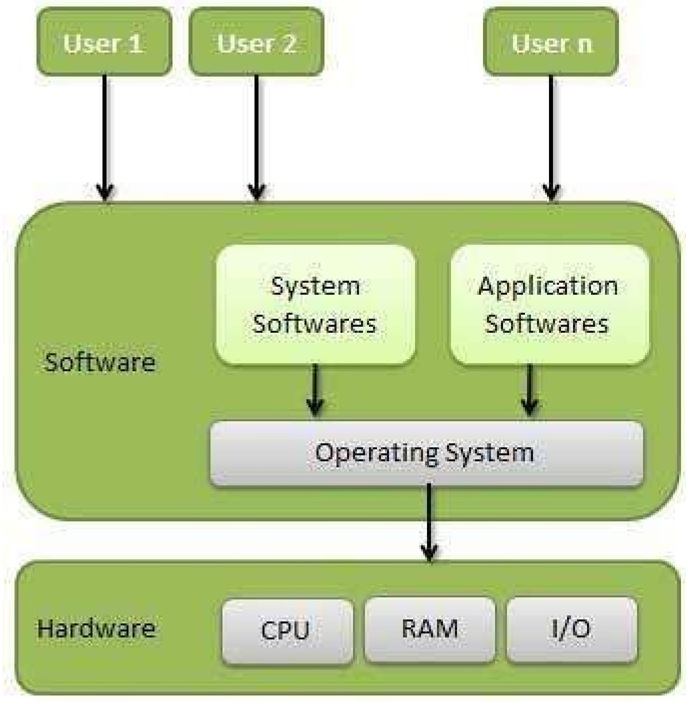
INTRODUCTION
CHAPTER-1

An Operating System (OS) is an interface between a computer user and computer hardware. An operating system is a software which performs all the basic tasks like file management, memory management, process management, handling input and output, and controlling peripheral devices such as disk drives and printers.

Some popular Operating Systems include Linux Operating System, Windows Operating System, VMS, OS/400, AIX, z/OS, etc.

Definition

An operating system is a program that acts as an interface between the user and the computer hardware and controls the execution of all kinds of programs.

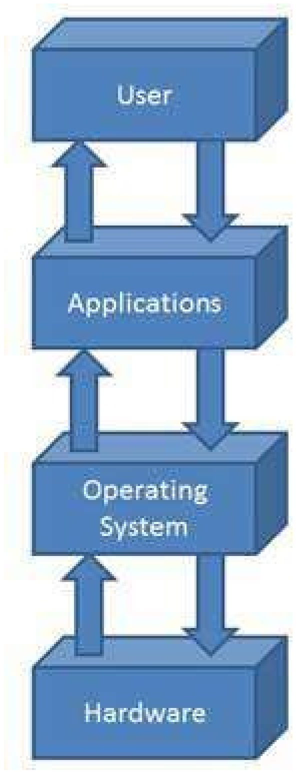


The Operating System is a program with the following features –

 An operating system is a program that acts as an interface between the software and the computer hardware.

 It is an integrated set of specialized programs used to manage overall resources and operations of the computer.

 It is a specialized software that controls and monitors the execution of all other programs that reside in the computer, including application programs and other system software.



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.

Characteristics of Operating System

Here is a list of some of the most prominent characteristic features of Operating Systems –

 Memory Management – Keeps track of the primary memory, i.e. what part of it is in use by whom, what part is not in use, etc. and allocates the memory when a process or program requests it.

- **Processor Management** – Allocates the processor (CPU) to a process and deallocates the processor when it is no longer required.
- **Device Management** – Keeps track of all the devices. This is also called I/O controller that decides which process gets the device, when, and for how much time.
- **File Management** – Allocates and de-allocates the resources and decides who gets the resources.
- **Security** – Prevents unauthorized access to programs and data by means of passwords and other similar techniques.
- **Job Accounting** – Keeps track of time and resources used by various jobs and/or users.
- **Control over System Performance** – Records delays between the request for a service and from the system.
- **Interaction with the Operators** – Interaction may take place via the console of the computer in the form of instructions. The Operating System acknowledges the same, does the corresponding action, and informs the operation by a display screen.
- **Error-detecting Aids** – Production of dumps, traces, error messages, and other debugging and error-detecting methods.
- **Coordination between Other Software and Users** – Coordination and assignment of compilers, interpreters, assemblers, and other software to the various users of the computer systems.

Function of Operating System

Following are some of important functions of an operating System.

- Memory Management
- Processor Management
- Device Management
- File Management
- Security
- Control over system performance
- Job accounting
- Error detecting aids
- Coordination between other software and users

Memory Management

Memory management refers to management of Primary Memory or Main Memory. Main memory is a large array of words or bytes where each word or byte has its own address.

Main memory provides a fast storage that can be accessed directly by the CPU. For a program to be executed, it must in the main memory. An Operating System does the following activities for memory management –

Keeps tracks of primary memory, i.e., what part of it are in use by whom, what part are not in use.

In multiprogramming, the OS decides which process will get memory when and how much.

Allocates the memory when a process requests it to do so.

De-allocates the memory when a process no longer needs it or has been terminated.

Processor Management

In multiprogramming environment, the OS decides which process gets the processor when and for how much time. This function is called **process scheduling**. An Operating System does the following activities for processor management –

Keeps tracks of processor and status of process. The program responsible for this task is known as **traffic controller**.

Allocates the processor (CPU) to a process.

De-allocates processor when a process is no longer required.

Device Management

An Operating System manages device communication via their respective drivers. It does the following activities for device management –

Keeps tracks of all devices. Program responsible for this task is known as the **I/O controller**.

Decides which process gets the device when and for how much time.

Allocates the device in the efficient way.

De-allocates devices.

File Management

A file system is normally organized into directories for easy navigation and usage. These directories may contain files and other directions.

An Operating System does the following activities for file management –

Keeps track of information, location, uses, status etc. The collective facilities are often known as **file system**.

Decides who gets the resources.

Allocates the resources.

De-allocates the resources.

Other Important Activities

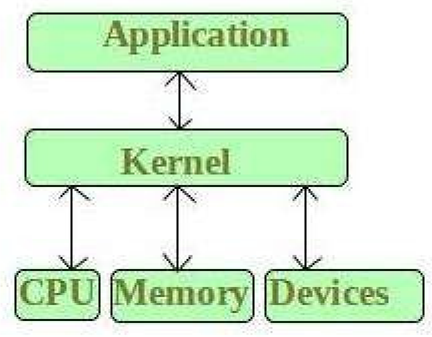
Following are some of the important activities that an Operating System performs –

- **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.

STRUCTURE OF OPERATING SYSTEM

Microkernel in Operating Systems

Kernel is the core part of an operating system which manages system resources. It also acts like a bridge between application and hardware of the computer. It is one of the first programs loaded on start-up (after the Bootloader).

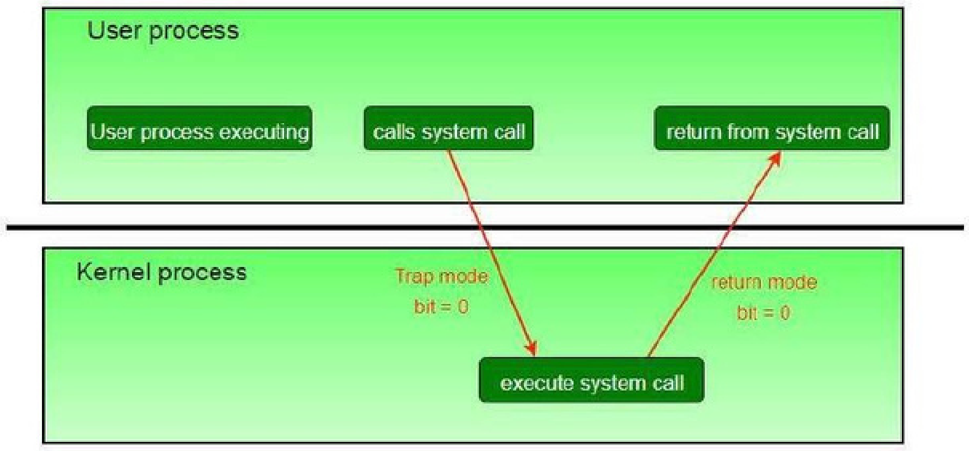


Kernel mode and User mode of CPU operation

The CPU can execute certain instruction only when it is in the kernel mode. These instruction are called privilege instruction. They allow implementation of special operation whose execution by the user program could interface with the functioning of operating system or activity of another user program. For example, instruction for managing memory protection.

- The operating system puts the CPU in kernel mode when it is executing in the kernel so, that kernel can execute some special operation.
- The operating system puts the CPU in user mode when a user program is in execution so, that user program cannot interface with the operating system program.
- User-level instruction does not require special privilege. Example are ADD,PUSH,etc.

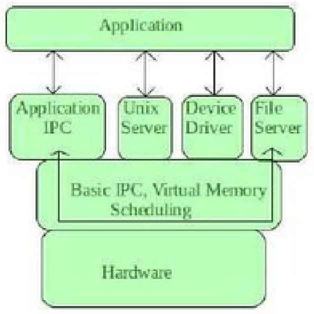
The concept of modes can be extended beyond two, requiring more than a single mode bit CPUs that support virtualization use one of these extra bits to indicate when the virtual machine manager, VMM, is in control of the system. The VMM has more privileges than ordinary user programs, but not so many as the full kernel.



System calls are typically implemented in the form of software interrupts, which causes the hardware's interrupt handler to transfer control over to an appropriate interrupt handler, which is part of the operating system, switching the mode bit to kernel mode in the process. The interrupt handler checks exactly which interrupt was generated, checks additional parameters (generally passed through registers) if appropriate, and then calls the appropriate kernel service routine to handle the service requested by the system call.

User programs' attempts to execute illegal instructions (privileged or non-existent instructions), or to access forbidden memory areas, also generate software interrupts, which are trapped by the interrupt handler and control is transferred to the OS, which issues an appropriate error message, possibly dumps data to a log (core) file for later analysis, and then terminates the offending program.

What is Microkernel?
Microkernel is one of the classification of the kernel. Being a kernel it manages all system resources. But in a microkernel, the **user services** and **kernel services** are implemented in different address space. The user services are kept in **user address space**, and kernel services are kept under **kernel address space**, thus also reduces the size of kernel and size of operating system as well.



It provides minimal services of process and memory management. The communication between client program/application and services running in user address space is established through message passing, reducing the speed of execution microkernel. The Operating System **remains unaffected** as user services and kernel services are isolated so if any user service fails it does not affect kernel service. Thus it adds to one of the
advantages in a microkernel. It is easily **extendable** i.e. if any new services are to be added they are added to user address space and hence requires no modification in kernel space.
It is also portable, secure and reliable.

Microkernel Architecture –

Since kernel is the core part of the operating system, so it is meant for handling the most important services only. Thus in this architecture only the most important services are inside kernel and rest of the OS services are present inside system application program. Thus users are able to interact with those not-so important services within the system application. And the microkernel is solely responsible for the most important services of operating system they are named as follows:
 Inter process-Communication
 Memory Management
 CPU-Scheduling
Advantages of Microkernel –
 The architecture of this kernel is small and isolated hence it can function better.
 Expansion of the system is easier, it is simply added in the system application without disturbing the kernel.

Kernel I/O Subsystem in Operating System

The kernel provides many services related to I/O. Several services such as scheduling, caching, spooling, device reservation, and error handling – are provided by the kernel, s I/O subsystem built on the hardware and device-driver infrastructure. The I/O subsystem is also responsible for protecting itself from errant processes and malicious users.

1. **I/O Scheduling –**
To schedule a set of I/O requests means to determine a good order in which to execute them. The order in which the application issues the system call is the best choice.
Scheduling can improve the overall performance of the system, can share device access permission fairly to all the processes, reduce the average waiting time, response time, turnaround time for I/O to complete.
OS developers implement schedules by maintaining a wait queue of the request for each device. When an application issue a blocking I/O system call, The request is placed in the queue for that device. The I/O scheduler rearranges the order to improve the

efficiency of the system.

2. **Buffering –**

A *buffer* is a memory area that stores data being transferred between two devices or between a device and an application. Buffering is done for three reasons.

1. The first is to cope with a speed mismatch between producer and consumer of a data stream.

2. The second use of buffering is to provide adaptation for data that have different data-transfer sizes.

3. The third use of buffering is to support copy semantics for the application I/O, "copy semantic " means, suppose that an application wants to write data on a disk that is stored in its buffer. it calls the **write()** system's call, providing a pointer to the buffer and the integer specifying the number of bytes to write.

3. **Caching –**

A *cache* is a region of fast memory that holds a copy of data. Access to the cached copy is much easier than the original file. For instance, the instruction of the currently running process is stored on the disk, cached in physical memory, and copied again in the CPU's secondary and primary cache.

The main difference between a buffer and a cache is that a buffer may hold only the existing copy of a data item, while a cache, by definition, holds a copy on faster storage of an item that resides elsewhere.

4. **Spooling and Device Reservation –**

A *spool* is a buffer that holds the output of a device, such as a printer that cannot accept interleaved data streams. Although a printer can serve only one job at a time, several applications may wish to print their output concurrently, without having their output mixes together.

The OS solves this problem by preventing all output from continuing to the printer. The output of all applications is spooled in a separate disk file. When an application finishes printing then the spooling system queues the corresponding spool file for output to the printer.

5. **Error Handling –**

An Os that uses protected memory can guard against many kinds of hardware and application errors so that a complete system failure is not the usual result of each minor mechanical glitch, Devices, and I/O transfers can fail in many ways, either for transient reasons, as when a network becomes overloaded or for permanent reasons, as when a disk controller becomes defective.

6. **I/O Protection –**

Errors and the issue of protection are closely related. A user process may attempt to issue illegal I/O instructions to disrupt the normal function of a system. We can use the various mechanisms to ensure that such disruption cannot take place in the system.

To prevent illegal I/O access, we define all I/O instructions to be privileged instructions. The user cannot issue I/O instruction directly.

CHAPTER-2

PROCESS MANAGEMENT

PROCESS SCHEDULING

- **The process scheduling is the activity of the process manager that handles the removal of the running process from the CPU and the selection of another process on the basis of a particular strategy.**
- **Process scheduling is an essential part of a Multiprogramming operating systems. Such operating systems allow more than one process to be loaded into the executable memory at a time and the loaded process shares the CPU using time multiplexing.**

The prime aim of the process scheduling system is to keep the CPU busy all the time and to deliver minimum response time for all programs.

For achieving this, the scheduler must apply appropriate rules for swapping

processes IN and OUT of CPU.

Scheduling fell into one of the two general categories:

- **Non Pre-emptive Scheduling:** When the currently executing process gives up the CPU voluntarily.
- **Pre-emptive Scheduling:** When the operating system decides to favor another process, pre-empting the currently executing process.

The OS maintains all PCBs in Process Scheduling Queues. The OS maintains a separate queue for each of the process states and PCBs of all processes in the same execution state are placed in the same queue. When the state of a process is changed, its PCB is unlinked from its current queue and moved to its new state queue.

The Operating System maintains the following important process scheduling queues –

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

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

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

MEASURING FACTOR FOR CPU SCHEDULING ALGORITHM

● THROUGHPUT : The performance of the CPU is measured by this unit.

Throughput means the number of jobs are completed per unit time

● BURST TIME : The time required by the processor to execute a job is termed as burst time. It is generally expressed in milliseconds(MS)

● WAITING TIME: It is the sum of the periods spent by a process in the ready queue.

Waiting time=starting time – receiving time

Least waiting time shows efficiency of an algorithm

● TURN AROUND TIME: The time interval between summation of the process and time of completion, is known as TAT

TAT= finish time – arrival time

TYPES OF CPU SCHEDULING ALGORITHM

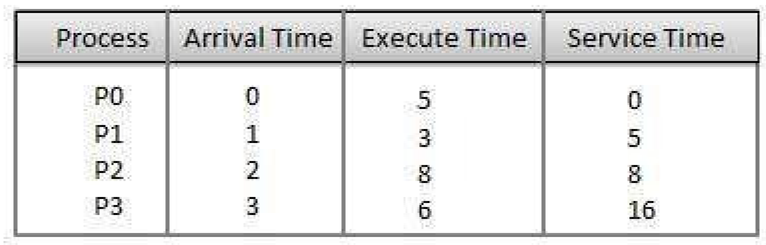
A Process Scheduler schedules different processes to be assigned to the CPU based on particular scheduling algorithms. There are six popular process scheduling algorithms which we are going to discuss in this chapter –

- First-Come, First-Served (FCFS) Scheduling
- Shortest-Job-Next (SJN) Scheduling
- Priority Scheduling
- Shortest Remaining Time
- Round Robin(RR) Scheduling
- Multiple-Level Queues Scheduling

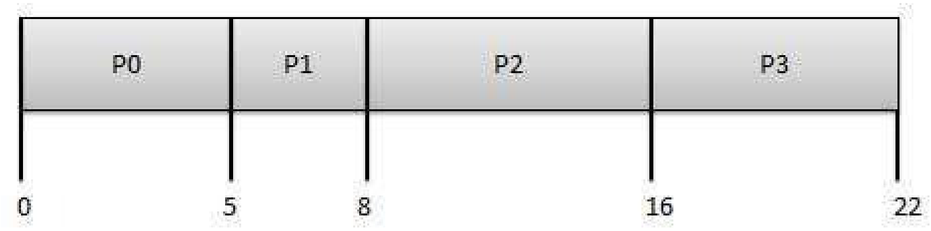
These algorithms are either **non-preemptive or preemptive**. Non-preemptive algorithms are designed so that once a process enters the running state, it cannot be preempted until it completes its allotted time, whereas the preemptive scheduling is based on priority where a scheduler may preempt a low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

First Come First Serve (FCFS)

- Jobs are executed on first come, first serve basis.
- It is a non-preemptive, pre-emptive 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, pre-emptive 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 required CPU time is not known.

 The processer should know in advance how much time process will take.

Given: Table of processes, and their Arrival time, Execution time

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

Waiting time of each process is as follows –

Process	Waiting Time
P0	0 - 0 = 0
P1	5 - 1 = 4
P2	14 - 2 = 12
P3	8 - 3 = 5

Average Wait Time: (0 + 4 + 12 + 5)/4 = 21 / 4 = 5.25

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.

Given: Table of processes, and their Arrival time, Execution time, and priority. Here we are considering 1 is the lowest priority.

Process	Arrival Time	Execution Time	Priority	Service Time
P0	0	5	1	0
P1	1	3	2	11
P2	2	8	1	14
P3	3	6	3	5

Waiting time of each process is as follows –

Process	Waiting Time
P0	0 - 0 = 0
P1	11 - 1 = 10
P2	14 - 2 = 12
P3	5 - 3 = 2

Average Wait Time: (0 + 10 + 12 + 2)/4 = 24 / 4 = 6

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 preempted 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 give preference.

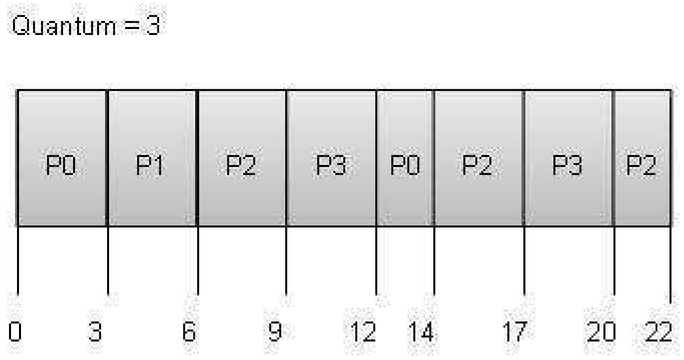
Round Robin Scheduling

- Round Robin is the 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.



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

Multiple-Level Queues Scheduling

Multiple-level queues are not an independent scheduling algorithm. They make use of other existing algorithms to group and schedule jobs with common characteristics.

- Multiple queues are maintained for processes with common characteristics.
- Each queue can have its own scheduling algorithms.
- Priorities are assigned to each queue.

For example, CPU-bound jobs can be scheduled in one queue and all I/O-bound jobs in another queue. The Process Scheduler then alternately selects jobs from each queue and assigns them to the CPU based on the algorithm assigned to the queue.

PROCESS MANAGEMENT

What is a Process?

PROGRAM UNDER EXECUTION IS CALLED PROCESS.

- **Process** is the execution of a program that performs the actions specified in that program. It can be defined as an execution unit where a program runs. The OS helps you to create, schedule, and terminates the processes which is used by CPU. A process created by the main process is called a child process.

- Process operations can be easily controlled with the help of PCB(Process Control Block). You can consider it as the brain of the process, which contains all the crucial information related to processing like process id, priority, state, CPU registers, etc.

What is Process Management?

Process management involves various tasks like creation, scheduling, termination of processes, and a dead lock. Process is a program that is under execution, which is an important part of modern-day operating systems. The OS must allocate resources that enable processes to share and exchange information. It also protects the resources of each process from other methods and allows synchronization among processes.

It is the job of OS to manage all the running processes of the system. It handles operations by performing tasks like process scheduling and such as resource allocation.

Process Architecture

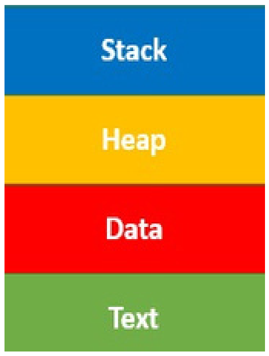
Here, is an Architecture diagram of the Process

Stack: The Stack stores temporary data like function parameters, returns addresses, and local variables.

Heap Allocates memory, which may be processed during its run time.

Data: It contains the variable.

Text: Text Section includes the current activity, which is represented by the value of the Program Counter.



Process Control Blocks(PCB)

The PCB is a full form of Process Control Block. It is a data structure that is maintained by the Operating System for every process. The PCB should be identified by an integer Process ID

(PID). It helps you to store all the information required to keep track of all the running processes.

It is also accountable for storing the contents of processor registers. These are saved when the process moves from the running state and then returns back to it. The information is quickly updated in the PCB by the OS as soon as the process makes the state transition.

Process States

A process state is a condition of the process at a specific instant of time. It also defines the current position of the process.

There are mainly seven stages of a process which are:

• New: The new process is created when a specific program calls from secondary memory/ hard disk to primary memory/ RAM a

• Ready: In a ready state, the process should be loaded into the primary memory, which is ready for execution.

• Waiting: The process is waiting for the allocation of CPU time and other resources for execution.

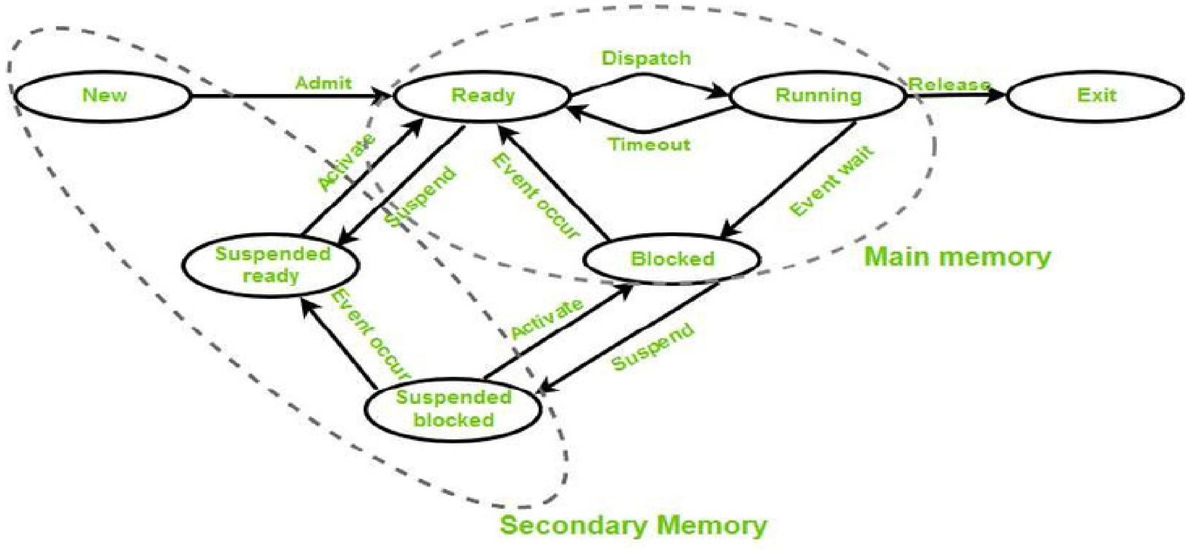
• Executing: The process is an execution state.

• Blocked: It is a time interval when a process is waiting for an event like I/O operations to complete.

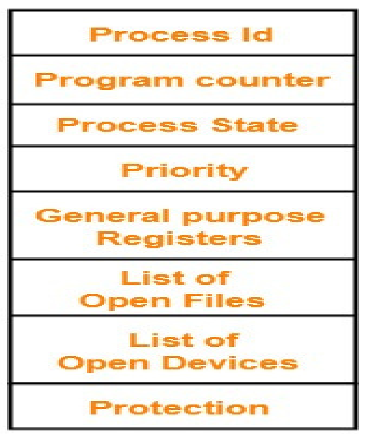
• Suspended: Suspended state defines the time when a process is ready for execution but has not been placed in the ready queue by OS.

• Terminated: Terminated state specifies the time when a process is terminated

After completing every step, all the resources are used by a process, and memory becomes free.



Process Control Block(PCB)



Process Control Block (PCB)

• **Pointer –** It is a stack pointer which is required to be saved when the process is switched from one state to another to retain the current position of the process.

• **Process state –** It stores the respective state of the process.

• **Process number –** Every process is assigned with a unique id known as process ID or PID which stores the process identifier.

• **Program counter –** It stores the counter which contains the address of the next instruction that is to be executed for the process.

• **Register –** These are the CPU registers which includes: accumulator, base, registers and general purpose registers.

• **Memory limits –** This field contains the information about memory management system used by operating system. This may include the page tables, segment tables etc.

• **Open files list –** This information includes the list of files opened for a process.

Every process is represented in the operating system by a process control block, which is also called a task control block.

Here, are important components of PCB

• **Process state:** A process can be new, ready, running, waiting, etc.

• **Program counter:** The program counter lets you know the address of the next instruction, which should be executed for that process.

• **CPU registers:** This component includes accumulators, index and general-purpose registers, and information of condition code.

• **CPU scheduling information:** This component includes a process priority, pointers for scheduling queues, and various other scheduling parameters.

• **Accounting and business information:** It includes the amount of CPU and time utilities like real time used, job or process numbers, etc.

• **Memory-management information:** This information includes the value of the base and limit registers, the page, or segment tables. This depends on the memory system, which is used by the operating system.

• **I/O status information:** This block includes a list of open files, the list of I/O devices that are allocated to the process, etc.

INTERPROCESS COMMUNICATION(IPC)

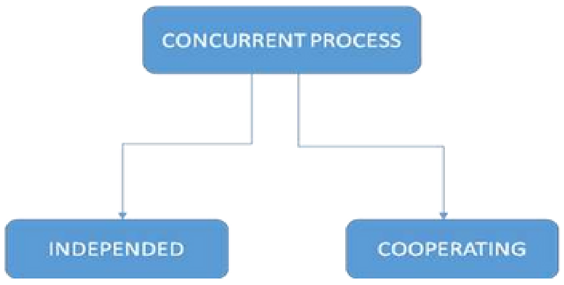
Concurrent Processes in Operating System

Concurrent processing is a computing model in which multiple processors execute instructions simultaneously for better performance. Concurrent means, which occurs when something else happens. The tasks are broken into sub-types, which are then assigned to different processors to perform simultaneously, sequentially instead, as they would have to be performed by one processor. Concurrent processing is sometimes synonymous with parallel processing.

EXAMPLE

A simple **example** of a task that can be performed more efficiently by **concurrent processing** is a program to calculate the sum of a large list of numbers. Several **processes** can simultaneously compute the sum of a subset of the list, after which these sums are added to produce the final total.

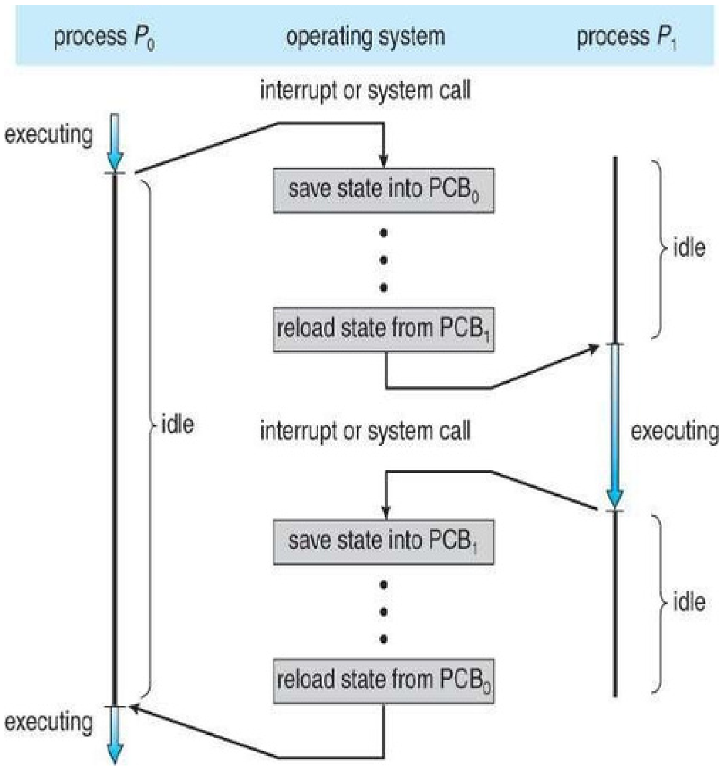
INTERPROCESS COMMUNICATION(IPC)



Independent Processes operating concurrently on a systems are those that can neither affect other processes or be affected by other processes.

Cooperating processes are those that can affect or are affected by other **processes** running on the system. **Cooperating processes** may share data with each other.

CPU SWITCH FROM PROCESS TO PROCESS(CONTEXT SWITCH/PROCESS SWITCH)



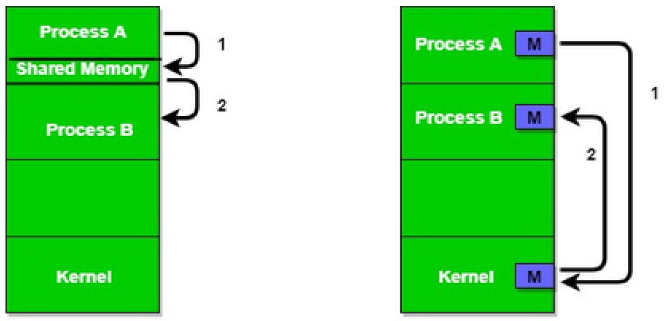
ADVANTAGES OF CO-OPERATING PROCESS

- **Information Sharing**
- **Computation Speedup**
- **Modularity**
- **Convenience**

IPC MODEL

1. SHARED MEMORY SYSTEM

2. MESSAGE PASSING SYSTEM



PROCESS SYNCHRONIZATION

● It is the task phenomenon of coordinating the execution of processes in such a way that no two processes can have access to the same shared data and resources.

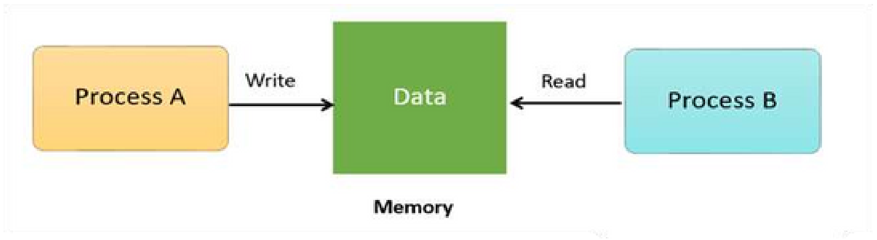
● It is a procedure that is involved in order to preserve the appropriate order of execution of cooperative processes.

● In order to synchronize the processes, there are various synchronization mechanisms.

● Process Synchronization is mainly needed in a multi-process system when multiple processes are running together, and more than one processes try to gain access to the same shared resource or any data at the same time.

How Process Synchronization Works?

For Example, process A changing the data in a memory location while another process B is trying to read the data from the same memory location. There is a high probability that data read by the second process will be erroneous.



Sections of a Program

Here, are four essential elements of the critical section:

● **Entry Section:** It is part of the process which decides the entry of a particular process.

● **Critical Section:** This part allows one process to enter and modify the shared variable.

● **Exit Section:** Exit section allows the other process that are waiting in the Entry Section, to enter into the Critical Sections. It also checks that a process that finished its execution should be removed through this Section.

● **Remainder Section:** All other parts of the Code, which is not in Critical, Entry, and Exit Section, are known as the Remainder Section.

Race Condition

● At the time when more than one process is either executing the same code or accessing the same memory or any shared variable; In that condition, there is a possibility that the output or the value of the shared variable is wrong so for that purpose all the processes are doing the race to say that my output is correct. This condition is commonly known as **a race condition.** As several processes access and process the manipulations on the same data in a concurrent manner and due to which the outcome depends on the particular order in which the access of data takes place.

● Mainly this condition is a situation that may occur inside the **critical section**. Race condition in the critical section happens when the result of multiple thread execution differs according to the order in which the threads execute. But this condition is critical sections can be avoided if the critical section is treated as an atomic instruction. Proper thread synchronization using locks or atomic variables can also prevent race conditions.

What is Critical Section Problem?

● A critical section is a segment of code which can be accessed by a signal process at a specific point of time. The section consists of shared data resources that required to be accessed by other processes.

● The entry to the critical section is handled by the wait() function, and it is represented as P().

● The exit from a critical section is controlled by the signal() function, represented as V().

● In the critical section, only a single process can be executed. Other processes, waiting to execute their critical section, need to wait until the current process completes its execution.

Rules for Critical Section

The critical section need to must enforce all three rules:

● **Mutual Exclusion:** Mutual Exclusion is a special type of binary semaphore which is used for controlling access to the shared resource. It includes a priority inheritance mechanism to avoid extended priority inversion problems. Not more than one process can execute in its critical section at one time.

● **Progress:** This solution is used when no one is in the critical section, and someone wants in. Then those processes not in their reminder section should decide who should go in, in a finite time.

● **Bound Waiting:** When a process makes a request for getting into critical section, there is a specific limit about number of processes can get into their critical section. So, when the limit is reached, the system must allow request to the process to get into its critical section.

Solutions To The Critical Section

In Process Synchronization, critical section plays the main role so that the problem must be solved.

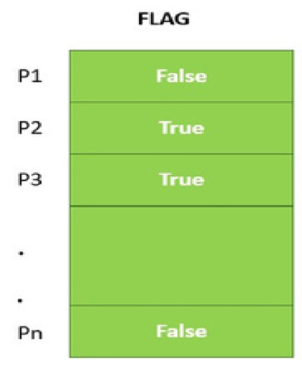
Here are some widely used methods to solve the critical section problem.

Peterson Solution

● Peterson's solution is widely used solution to critical section problems. This algorithm was developed by a computer scientist Peterson that's why it is named as a Peterson's solution.

● In this solution, when a process is executing in a critical state, then the other process only executes the rest of the code, and the opposite can happen. This method also helps to make sure that only a single process runs in the critical section at a specific time.

Example



PROCESS Pi

FLAG[i] = true

while((turn != i) AND (CS is !free))

{ wait; }

CRITICAL SECTION FLAG[i] = false

turn = j; //choose another process to go to CS

● Assume there are N processes (P1, P2, ... PN) and every process at some point of time requires to enter the Critical Section

● A FLAG[] array of size N is maintained which is by default false. So, whenever a process requires to enter the critical section, it has to set its flag as true. For example, If Pi wants to enter it will set FLAG[i]=TRUE.

● Another variable called TURN indicates the process number which is currently waiting to enter into the CS.

● The process which enters into the critical section while exiting would change the TURN to another number from the list of ready processes.

● Example: turn is 2 then P2 enters the Critical section and while exiting turn=3 and therefore P3 breaks out of wait loop.

Synchronization Hardware

● Some times the problems of the Critical Section are also resolved by hardware. Some operating system offers a lock functionality where a Process acquires a lock when entering the Critical section and releases the lock after leaving it. So when another process is trying to enter the critical section, it will not be able to enter as it is locked. It can only do so if it is free by acquiring the lock itself.

Mutex Locks

● Synchronization hardware not simple method to implement for everyone, so strict software method known as Mutex Locks was also introduced.

● In this approach, in the entry section of code, a LOCK is obtained over the critical resources used inside the critical section. In the exit section that lock is released.

Semaphore Solution

● Semaphore is simply a variable that is non-negative and shared between threads. It is another algorithm or solution to the critical section problem. It is a signaling mechanism and a thread that is waiting on a semaphore, which can be signaled by another thread.

It uses two atomic operations, 1)wait, and 2) signal for the process synchronization.

Example

```
WAIT ( S ):
while ( S <= 0 );
S = S - 1;
SIGNAL ( S ):
S = S + 1;
```

There are two types of semaphores:

● **Binary Semaphores**

● **Counting Semaphores**

- Binary Semaphores: They can only be either 0 or 1. They are also known as mutex locks, as the locks can provide mutual exclusion. All the processes can share the same mutex semaphore that is initialized to 1. Then, a process has to wait until the lock becomes 0. Then, the process can make the mutex semaphore 1 and start its critical section. When it completes its critical section, it can reset the value of mutex semaphore to 0 and some other process can enter its critical section.
- Counting Semaphores: They can have any value and are not restricted over a certain domain. They can be used to control access to a resource that has a limitation on the number of simultaneous accesses. The semaphore can be initialized to the number of instances of the resource. Whenever a process wants to use that resource, it checks if the number of remaining instances is more than zero, i.e., the process has an instance available. Then, the process can enter its critical section thereby decreasing the value of the counting semaphore by 1. After the process is over with the use of the instance of the resource, it can leave the critical section thereby adding 1 to the number of available instances of the resource.

CHAPTER-3

MEMORY MANAGEMENT

- **Memory Management** is the process of controlling and coordinating computer memory, assigning portions known as blocks to various running programs to optimize the overall performance of the system.
- It is the most important function of an operating system that manages primary memory. It helps processes to move back and forward between the main memory and execution disk. It helps OS to keep track of every memory location, irrespective of whether it is allocated to some process or it remains free.

Why Use Memory Management?

- It allows you to check how much memory needs to be allocated to processes that decide which processor should get memory at what time.
- Tracks whenever inventory gets freed or unallocated. According to it will update the status.
- It allocates the space to application routines.
- It also make sure that these applications do not interfere with each other.
- Helps protect different processes from each other
- It places the programs in memory so that memory is utilized to its full extent.

The memory management techniques is divided into two parts.

– **Uniprogramming:**

In the uniprogramming technique, the RAM is divided into two parts one part is for the resigning the operating system and other portion is for the user process. Here the fence register is used which contain the last address of the operating system parts. The operating system will compare the user data addresses with the fence register and if it is different that means the user is not entering in the OS area. Fence register is also called boundary register and is used to prevent a user from entering in the operating system area. Here the CPU utilization is very poor and hence multiprogramming is used.

– **Multiprogramming:**

In the multiprogramming, the multiple users can share the memory simultaneously. By multiprogramming we mean there will be more than one process in the main memory and if the running process wants to wait for an event like I/O then instead of sitting ideal CPU will make a context switch and will pick another process.

– Contiguous memory allocation

– Non-contiguous memory allocation

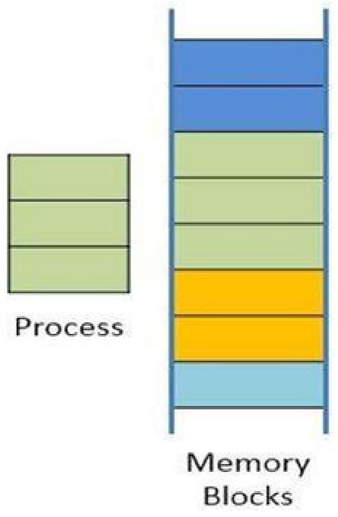
Contiguous memory allocation

– In **contiguous memory allocation**, all the available memory space remain together in one place. It means freely available memory partitions are not scattered here and there across the whole memory space.

– In the **contiguous memory allocation**, both the operating system and the user must reside in the main memory. The main memory is divided into two portions one portion is for the operating and other is for the user program.

– In the **contiguous memory allocation** when any user process request for the memory a single section of the contiguous memory block is given to that process according to its need. We can achieve contiguous memory allocation by dividing memory into the fixed-sized partition.

A single process is allocated in that fixed sized single partition. But this will increase the degree of multiprogramming means more than one process in the main memory that bounds the number of fixed partition done in memory. Internal fragmentation increases because of the contiguous memory allocation.



Contiguous Memory Allocation

• **Fixed sized partition**

In the fixed sized partition the system divides memory into fixed size partition (may or may not be of the same size) here entire partition is allowed to a process and if there is some wastage inside the partition is allocated to a process and if there is some wastage inside the partition then it is called internal fragmentation.

Advantage: Management or book keeping is easy.

Disadvantage: Internal fragmentation

• **Variable size partition**

In the variable size partition, the memory is treated as one unit and space allocated to a process is exactly the same as required and the leftover space can be reused again.

Advantage: There is no internal fragmentation.

Disadvantage: Management is very difficult as memory is becoming purely fragmented after some time.

Non-contiguous memory allocation

• In the **non-contiguous memory allocation** the available free memory space are scattered here and there and all the free memory space is not at one place. So this is time-consuming.

• In the **non-contiguous memory allocation**, a process will acquire the memory space but it is not at one place it is at the different locations according to the process requirement.

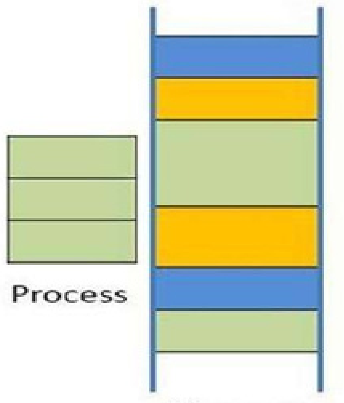
• This technique of **non-contiguous memory allocation** reduces the wastage of memory which leads to internal and external fragmentation. This utilizes all the free memory space which is created by a different process.

• **Non-contiguous memory allocation** is of different types,

• Paging

• Segmentation

• Segmentation with paging



Noncontiguous Memory Allocation

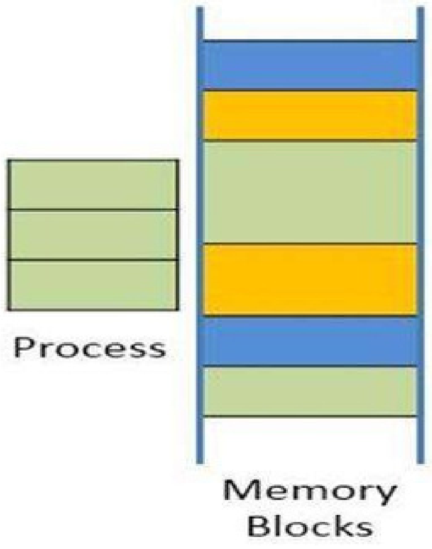
Difference between Contiguous and Non-contiguous Memory Allocation :

Contiguous Memory Allocation	Non-Contiguous Memory Allocation
Contiguous memory allocation allocates consecutive blocks of memory to a file/process.	Non-Contiguous memory allocation allocates separate blocks of memory to a file/process.

Faster in Execution. Slower in Execution.	
It is easier for the OS to control. It is difficult for the OS to control.	
Overhead is minimum as not much address translations are there while executing a address translations. process.	More Overheads are there as there are more
Internal fragmentation occurs in External fragmentation occurs in Non- Contiguous memory allocation method. Contiguous memory allocation method.	
It includes single partition allocation and It includes paging and segmentation. multi-partition allocation.	
Wastage of memory is there. No memory wastage is there.	
In contiguous memory allocation, In non-contiguous memory allocation, swapped-in processes are arranged in the swapped-in processes can be arranged in any originally allocated space.place in the memory.	

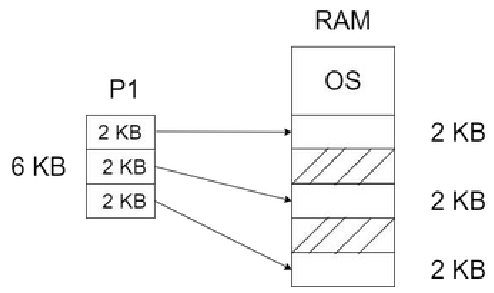
NON CONTIGUOUS MEMORY ALLOCATION

In the **non-contiguous memory allocation** the available free memory space are scattered here and there and all the free memory space is not at one place. So this is time-consuming. In the **non-contiguous memory allocation**, a process will acquire the memory space but it is not at one place it is at the different locations according to the process requirement. This technique of **non-contiguous memory allocation** reduces the wastage of memory which leads to internal and external fragmentation. This utilizes all the free memory space which is created by a different process.



- In contiguous memory allocation a process cannot span at different memory location, but here we can divide the process and allocate different parts of memory to it.

- External fragmentation is a problem which is present in contiguous memory allocation, can be removed by using the concept of non-contiguous memory allocation.



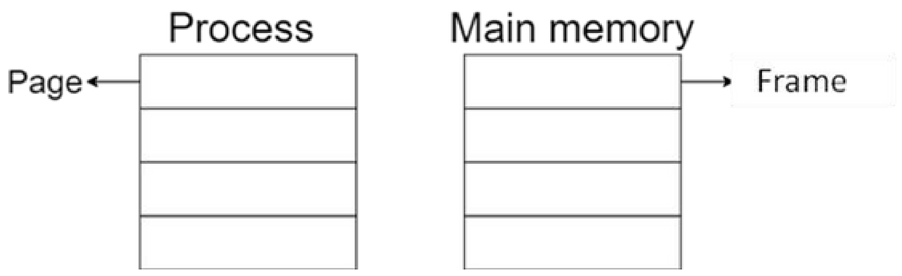
• Here we are dividing a process into small fragments *(parts)* and then allocating them to different parts of RAM, by this we are removing the problem known as external fragmentation.

• We have a problem with this approach, that the holes in the RAM are created dynamically which means their value; size and location keep on changing. So we have to analysis the RAM and divide the process, only then we can place/ put process inside memory *(RAM)*, but this is the problem because this whole process require time and it is not time efficient.

• So to overcome this problem we divide the processes into multiple parts before it comes into the main memory *(RAM)*, and this divided section is known as **page**.

• We are doing partition in secondary memory.

• On the other hand we also divide the main memory *(RAM)* and these section is known as **frame**



• Process will be divided into **pages** and main memory *(RAM)* will be divided into **frames**.

• This is because than page will fit in frame easily.

• Page size = Frame size

SWAPPING

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

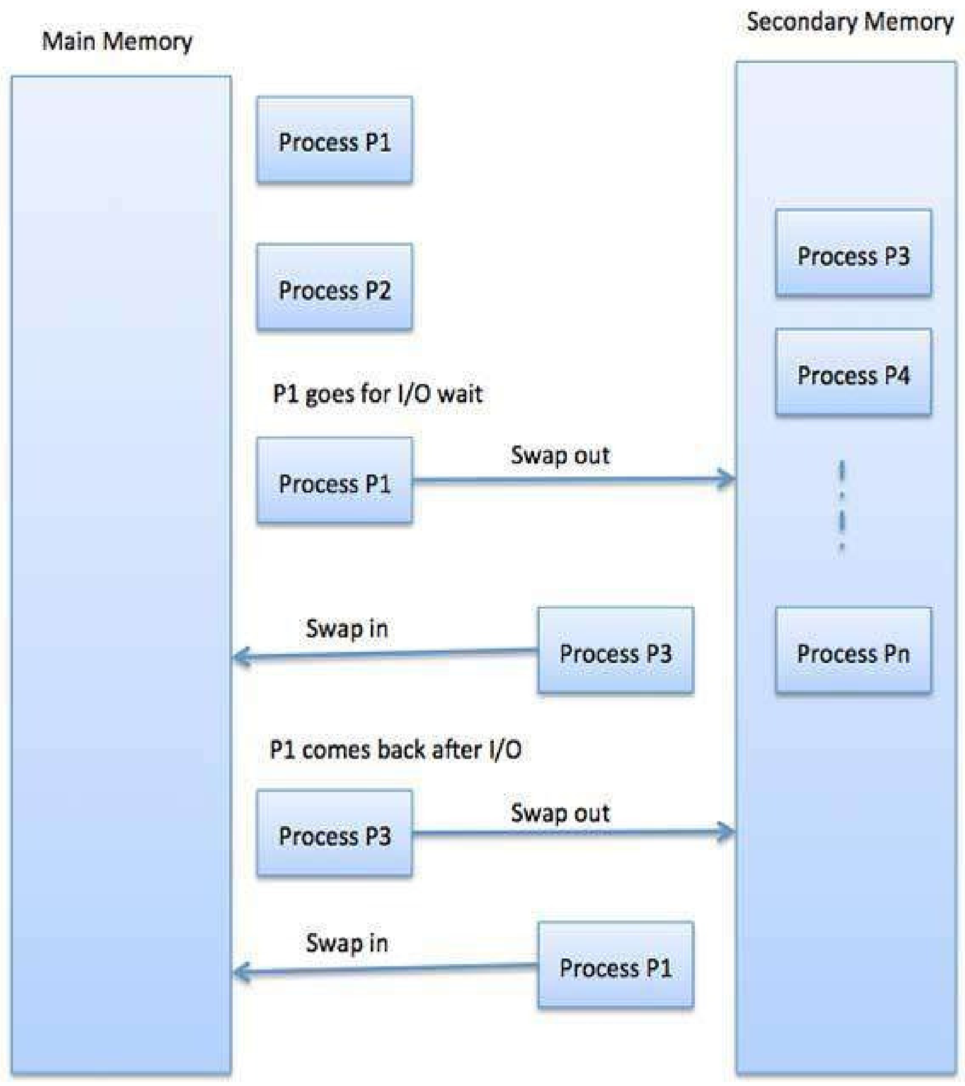
The purpose of the swapping in operating system is to access the data present in the hard disk and bring it to RAM so that the application programs can use it. The thing to remember is that swapping is used only when data is not present in RAM.

Although the process of swapping affects the performance of the system, it helps to run larger and more than one process. This is the reason why swapping is also referred to as memory compaction.

The concept of swapping has divided into two more concepts: Swap-in and Swap-out.

• **Swap-out** is a method of removing a process from RAM and adding it to the hard disk.

• **Swap-in** is a method of removing a program from a hard disk and putting it back into the main memory or RAM.



Example: Suppose the user process's size is 2048KB and is a standard hard disk where swapping has a data transfer rate of 1Mbps. Now we will calculate how long it will take to transfer from main memory to secondary memory.

User process size is 2048Kb

Data transfer rate is 1Mbps = 1024 kbps

Time = process size / transfer rate

= 2048 / 1024

= 2 seconds

= 2000 milliseconds

Now taking swap-in and swap-out time, the process will take 4000 milliseconds.

Advantages of Swapping

1. It helps the CPU to manage multiple processes within a single main memory.

2. It helps to create and use virtual memory.

3. Swapping allows the CPU to perform multiple tasks simultaneously. Therefore, processes do not have to wait very long before they are executed.

4. It improves the main memory utilization.

Disadvantages of Swapping

1. If the computer system loses power, the user may lose all information related to the program in case of substantial swapping activity.

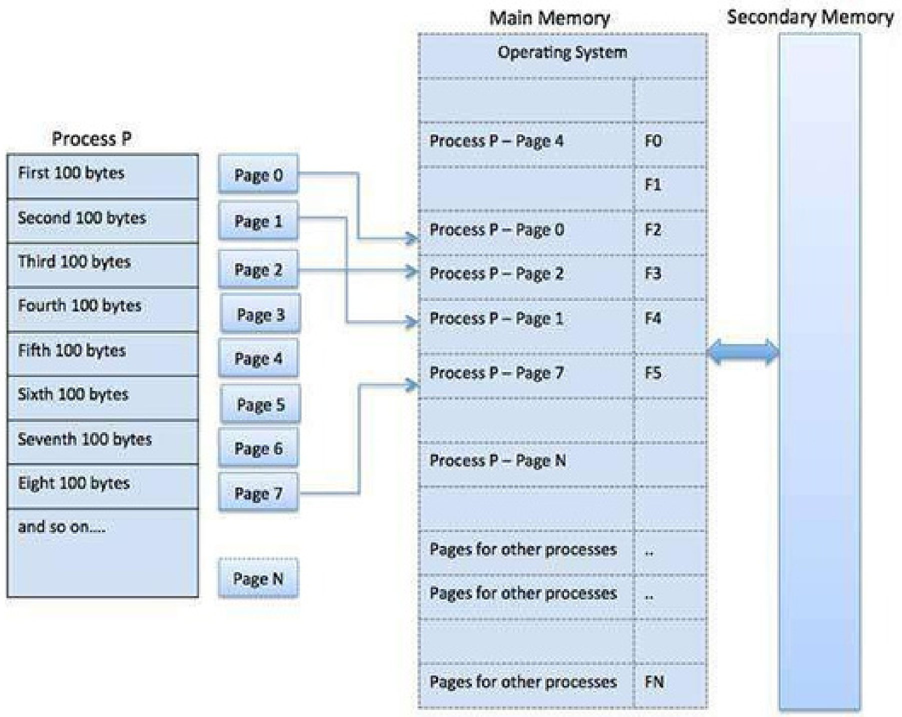
2. If the swapping algorithm is not good, the composite method can increase the number of Page Fault and decrease the overall processing performance.

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

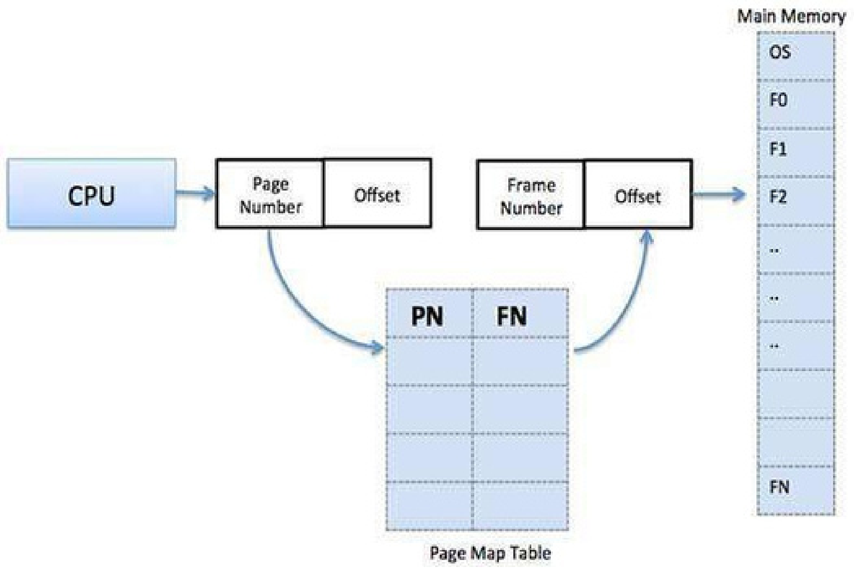
Page address is called **logical address** and represented by **page number** and the **offset**.

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.

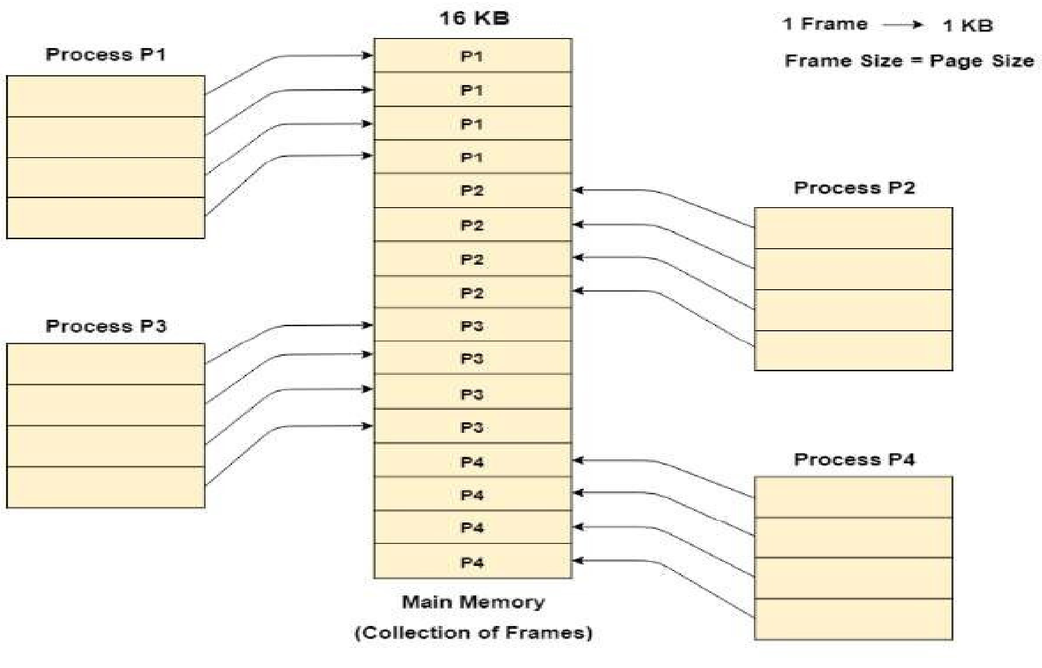
Example

• Let us consider the main memory size 16 Kb and Frame size is 1 KB therefore the main memory will be divided into the collection of 16 frames of 1 KB each.

• There are 4 processes in the system that is P1, P2, P3 and P4 of 4 KB each. Each process is divided into pages of 1 KB each so that one page can be stored in one frame.

• Initially, all the frames are empty therefore pages of the processes will get stored in the contiguous way.

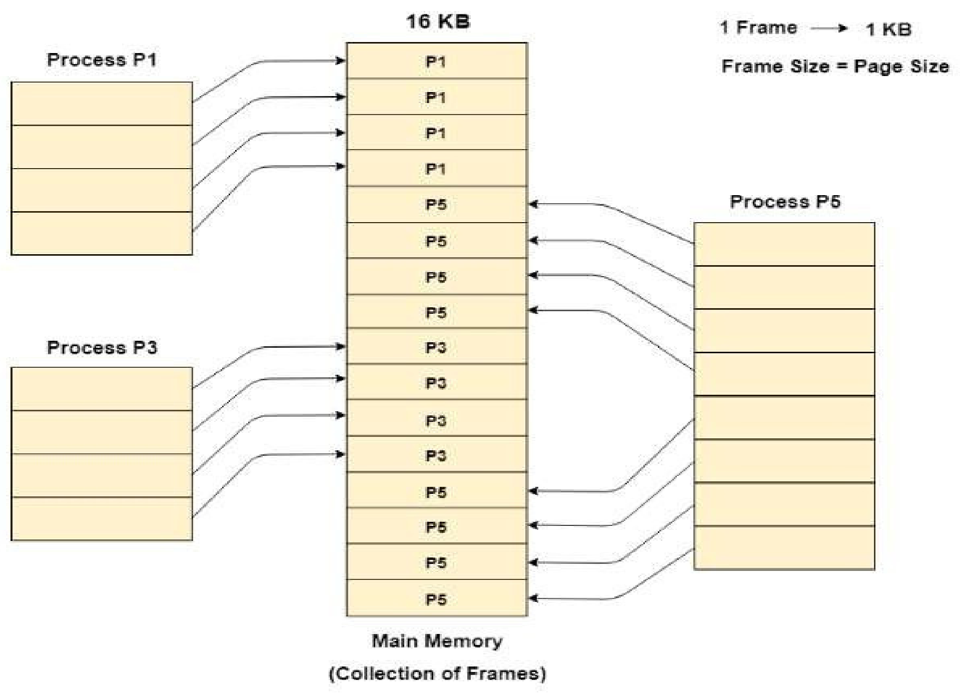
• Frames, pages and the mapping between the two is shown in the image below.



Paging

Let us consider that, P2 and P4 are moved to waiting state after some time. Now, 8 frames become empty and therefore other pages can be loaded in that empty place. The process P5 of size 8 KB (8 pages) is waiting inside the ready queue.

Given the fact that, we have 8 non contiguous frames available in the memory and paging provides the flexibility of storing the process at the different places. Therefore, we can load the pages of process P5 in the place of P2 and P4.



Paging

Physical and Logical Address Space

Physical Address Space

Physical address space in a system can be defined as the size of the main memory. It is really important to compare the process size with the physical address space. The process size must be less than the physical address space.

Physical Address Space = Size of the Main Memory

If, physical address space = 64 KB = 2 ^ 6 KB = 2 ^ 6 X 2 ^ 10 Bytes = 2 ^ 16 bytes

Let us consider,
word size = 8 Bytes = 2 ^ 3 Bytes

Hence,
Physical address space (in words) = (2 ^ 16) / (2 ^ 3) = 2 ^ 13 Words

Therefore,
Physical Address = 13 bits

In General,
If, Physical Address Space = N Words

then, Physical Address = $\log_2 N$

Logical Address Space

Logical address space can be defined as the size of the process. The size of the process should be less enough so that it can reside in the main memory.
Let's say,

Logical Address Space = 128 MB = (2 ^ 7 X 2 ^ 20) Bytes = 2 ^ 27 Bytes
Word size = 4 Bytes = 2 ^ 2 Bytes

Logical Address Space (in words) = (2 ^ 27) / (2 ^ 2) = 2 ^ 25 Words
Logical Address = 25 Bits

In general,
If, logical address space = L words
Then, Logical Address = $\text{Log}_2 L$ bits

What is a Word?

The Word is the smallest unit of the memory. It is the collection of bytes. Every operating system defines different word sizes after analyzing the n-bit address that is inputted to the decoder and the 2
^n memory locations that are produced from the decoder.

Page Table

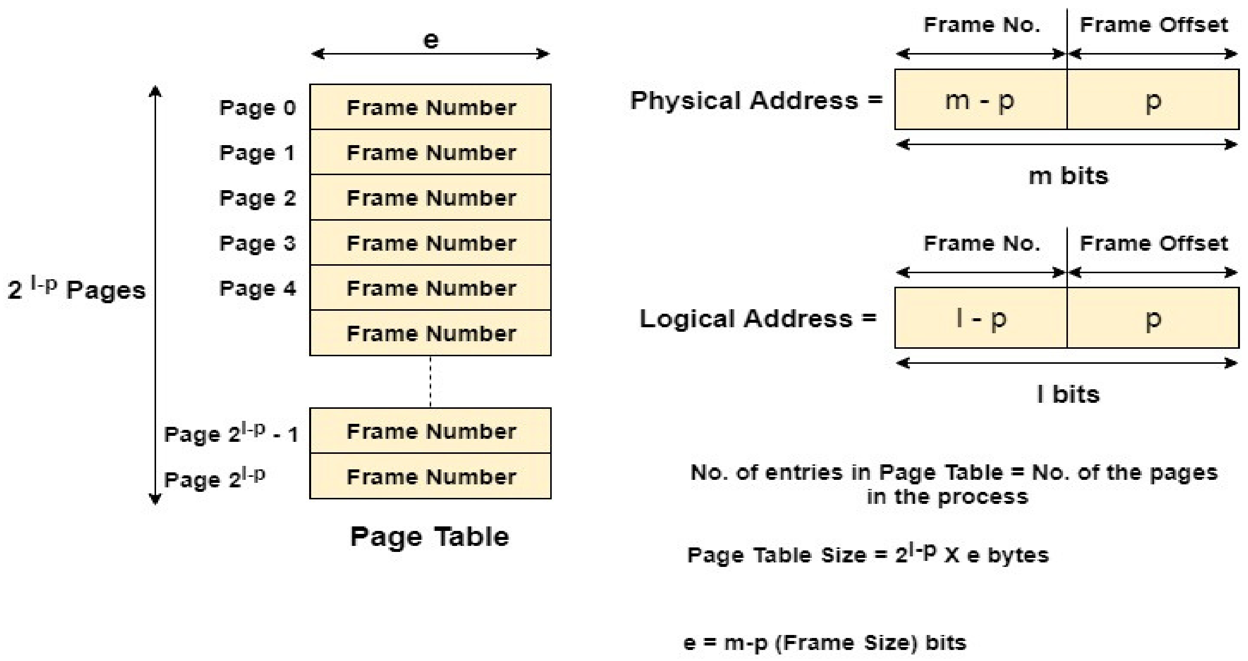
Page Table is a data structure used by the virtual memory system to store the mapping between logical addresses and physical addresses.

Logical addresses are generated by the CPU for the pages of the processes therefore they are generally used by the processes.
Physical addresses are the actual frame address of the memory. They are generally used by the hardware or more specifically by RAM subsystems.

The image given below considers,

Physical Address Space = M words
Logical Address Space = L words
Page Size = P words

Physical Address = $\log_2 M = m$ bits
Logical Address = $\log_2 L = l$ bits
page offset = $\log_2 P = p$ bits



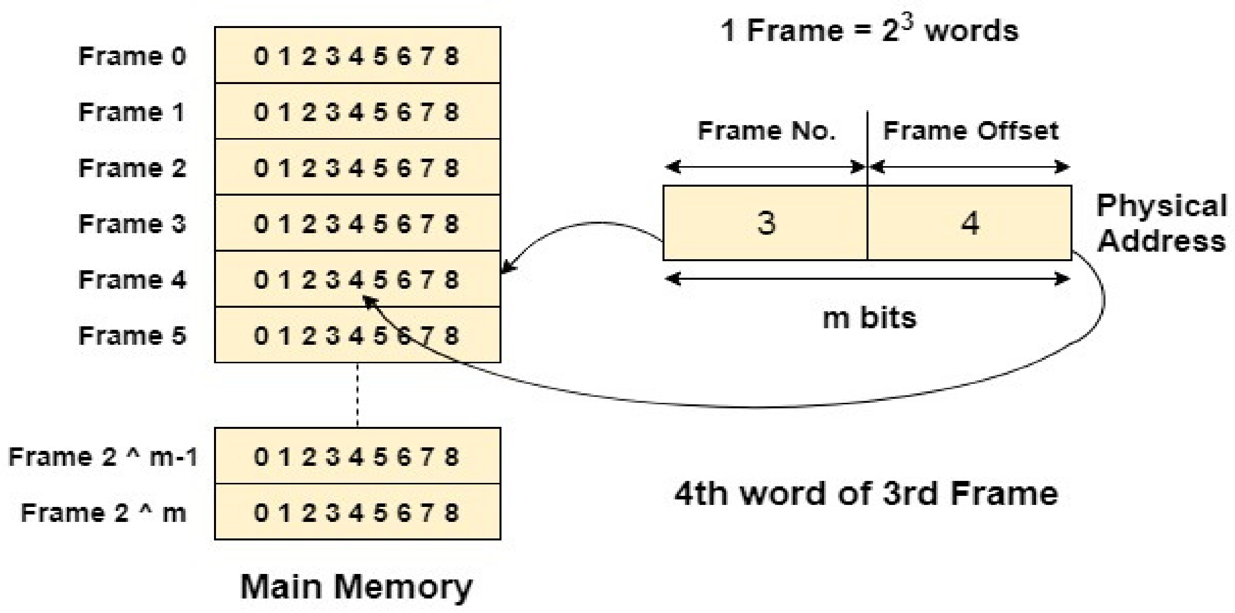
The CPU always accesses the processes through their logical addresses. However, the main memory recognizes physical address only.

In this situation, a unit named as Memory Management Unit comes into the picture. It converts the page number of the logical address to the frame number of the physical address. The offset remains same in both the addresses.

To perform this task, Memory Management unit needs a special kind of mapping which is done by page table. The page table stores all the Frame numbers corresponding to the page numbers of the page table.

In other words, the page table maps the page number to its actual location (frame number) in the memory.

In the image given below shows, how the required word of the frame is accessed with the help of offset.



Mapping from page table to main memory

In operating systems, there is always a requirement of mapping from logical address to the physical address. However, this process involves various steps which are defined as follows.

1. Generation of logical address

CPU generates logical address for each page of the process. This contains two parts: page number and offset.

2. Scaling

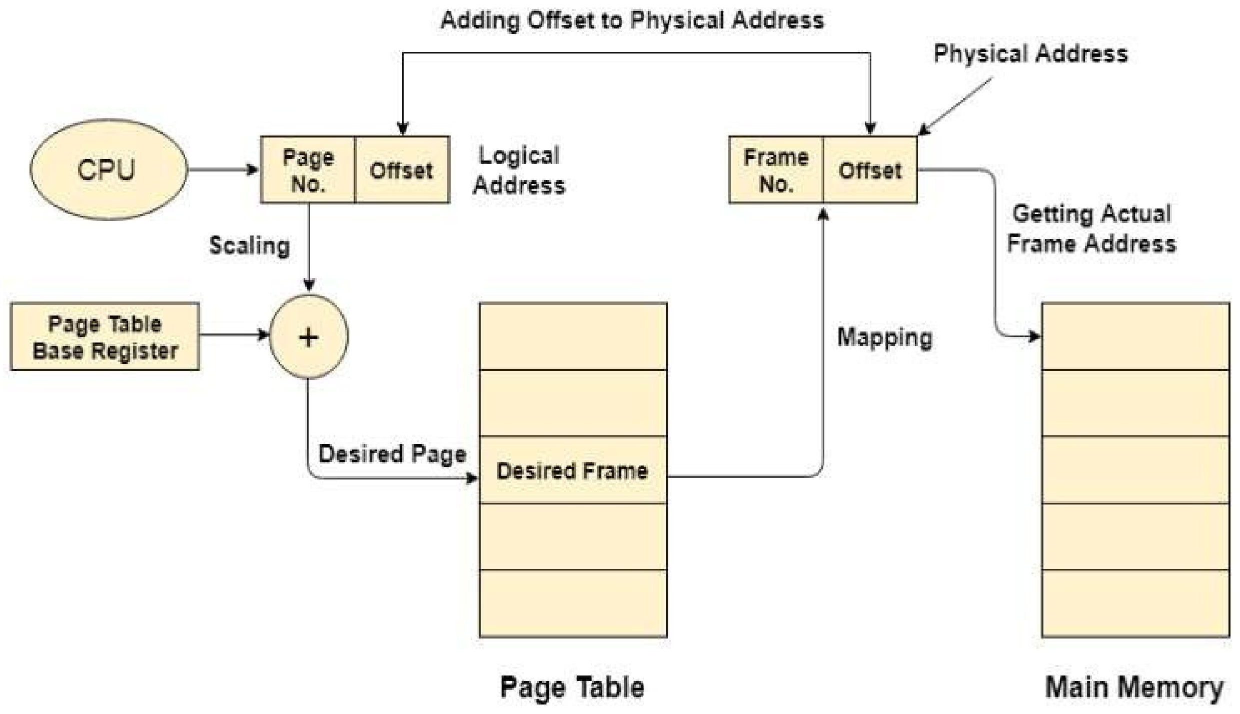
To determine the actual page number of the process, CPU stores the page table base in a special register. Each time the address is generated, the value of the page table base is added to the page number to get the actual location of the page entry in the table. This process is called scaling.

3. Generation of physical Address

The frame number of the desired page is determined by its entry in the page table. A physical address is generated which also contains two parts : frame number and offset. The Offset will be similar to the offset of the logical address therefore it will be copied from the logical address.

4. Getting Actual Frame Number

The frame number and the offset from the physical address is mapped to the main memory in order to get the actual word address.



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.

QUESTION

Given
LAS=4GB
PAS=64MB
PAGE SIZE=4KB
CALCULATE
PAGE SIZE=?
TOTAL NO OF PAGES=?
FRAME SIZE=?
TOTAL NO OF FRAMES=?
NO OF ENTRIES IN PT=?
SIZE OF PT=?

SEGMENTATION

▹ In Operating Systems, Segmentation is a memory management technique in which, the memory is divided into the variable size parts. Each part is known as segment which can be allocated to a process.

▹ The details about each segment are stored in a table called as segment table. Segment table is stored in one (or many) of the segments.

▹ Segment table contains mainly two information about segment:

▸ Base: It is the base address of the segment

▸ Limit: It is the length of the segment.

Why Segmentation is required?

It is better to have segmentation which divides the process into the segments. Each segment contain same type of functions such as main function can be included in one segment and the library functions can be included in the other segment

Translation of Logical address into physical address by segment table

▹ CPU generates a logical address which contains two parts:

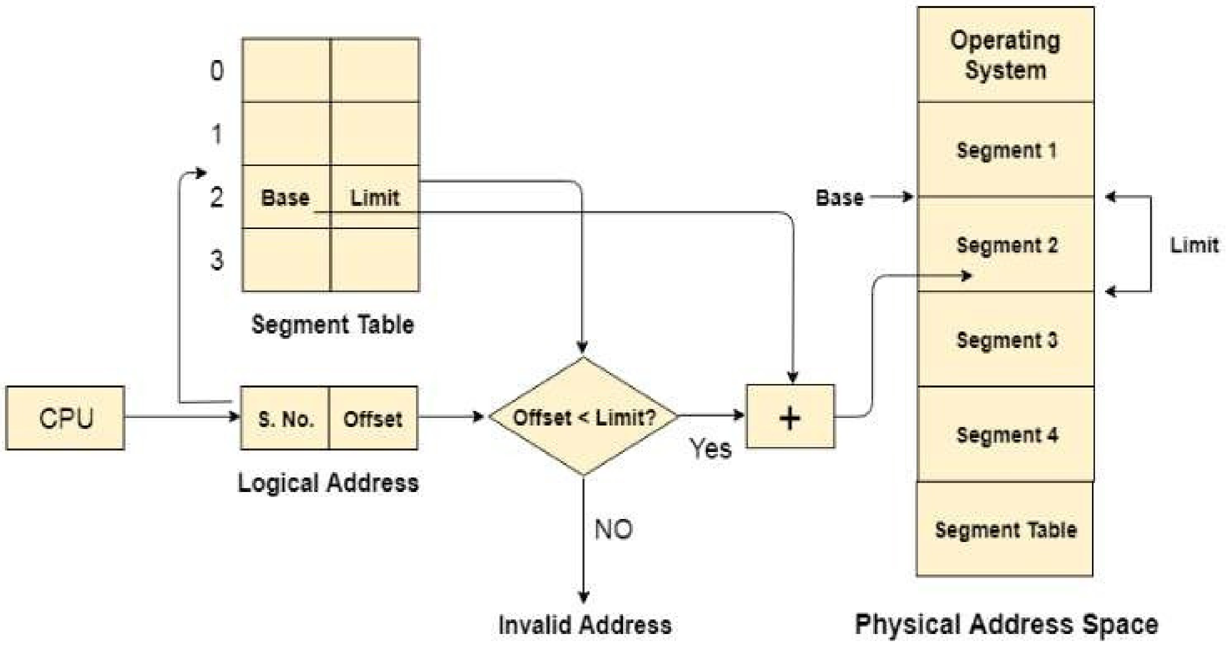
▸ Segment Number

▸ Offset

▹ The Segment number is mapped to the segment table. The limit of the respective segment is compared with the offset. If the offset is less than the limit then the address is valid otherwise it throws an error as the address is invalid.

▹ In the case of valid address, the base address of the segment is added to the offset to get the physical address of actual word in the main memory.

In the case of valid address, the base address of the segment is added to the offset to get the physical address of actual word in the main memory.



Advantages of Segmentation

1. No internal fragmentation
2. Average Segment Size is larger than the actual page size.
3. Less overhead
4. It is easier to relocate segments than entire address space.
5. The segment table is of lesser size as compare to the page table in paging.

Disadvantages

1. It can have external fragmentation.
2. it is difficult to allocate contiguous memory to variable sized partition.
3. Costly memory management algorithms.

Paging VS Segmentation

Sr No.	Paging	Segmentation
1	Non-Contiguous memory allocation	Non-contiguous memory allocation
2	Paging divides program into fixed size pages.	Segmentation divides program into variable size segments.

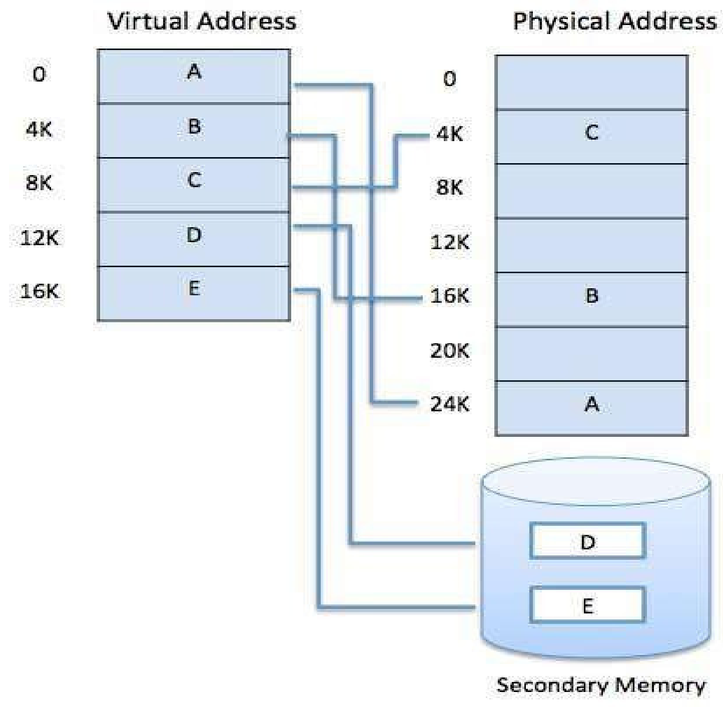
3	OS is responsible	Compiler is responsible.
4	Paging is faster than segmentation	Segmentation is slower than paging
5	Paging is closer to Operating System	Segmentation is closer to User
6	It suffers from internal fragmentation	It suffers from external fragmentation
7	There is no external fragmentation	There is no external fragmentation
8	Logical address is divided into page number and page offset	Logical address is divided into segment number and segment offset
9	Page table is used to maintain the page information.	Segment Table maintains the segment information
10	Page table entry has the frame number and some flag bits to represent details about pages.	Segment table entry has the base address of the segment and some protection bits for the segments.

VIRTUAL MEMORY USING 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 disk that's set up to emulate the computer's RAM.
- The main visible advantage of this scheme is that programs can be larger than physical memory. Virtual memory serves two purposes. First, it allows us to extend the use of physical memory by using disk. Second, it allows us to have memory protection, because each virtual address is translated to a physical address.
- It is a technique that is implemented using both hardware and software. It maps memory addresses used by a program, called virtual addresses, into physical addresses in computer memory.

1. All memory references within a process are logical addresses that are dynamically translated into physical addresses at run time. This means that a process can be swapped in and out of main memory such that it occupies different places in main memory at different times during the course of execution.
2. A process may be broken into number of pieces and these pieces need not be continuously located in the main memory during execution. The combination of dynamic run-time address translation and use of page or segment table permits this.

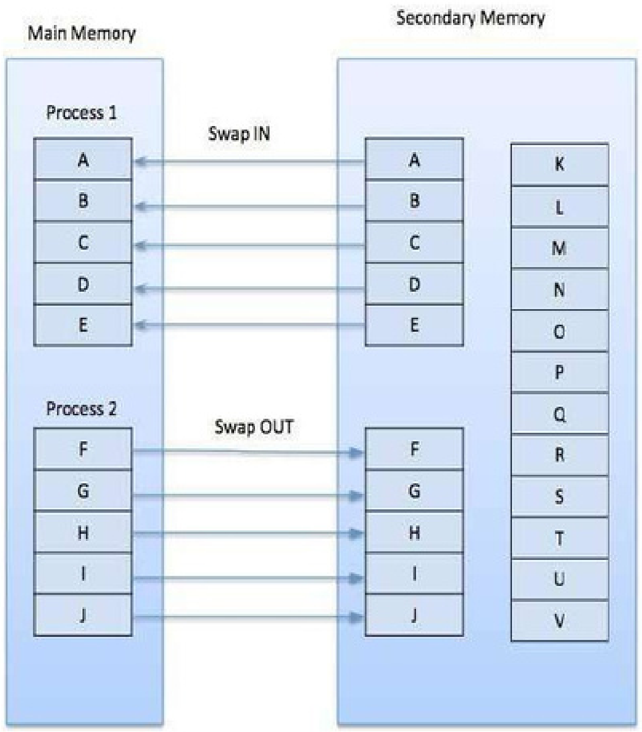
- Virtual memory is commonly implemented by demand paging.



DEMAND PAGING

- A demand paging system is quite similar to a paging system with swapping where processes reside in secondary memory and pages are loaded only on demand, not in advance. When a context switch occurs, the operating system does not copy any of the old program's pages out to the disk or any of the new program's pages into the main memory Instead, it just begins executing the new program after loading the first page and fetches that program's pages as they are referenced.

- While executing a program, if the program references a page which is not available in the main memory because it was swapped out a little ago, the processor treats this invalid memory reference as a page fault and transfers control from the program to the operating system to demand the page back into the memory.



The process of loading the page into memory on demand (whenever page fault occurs) is known as demand paging.

The process includes the following steps :

1. If CPU try to refer a page that is currently not available in the main memory, it generates an interrupt indicating memory access fault.

2. The OS puts the interrupted process in a blocking state. For the execution to proceed the OS must bring the required page into the memory.

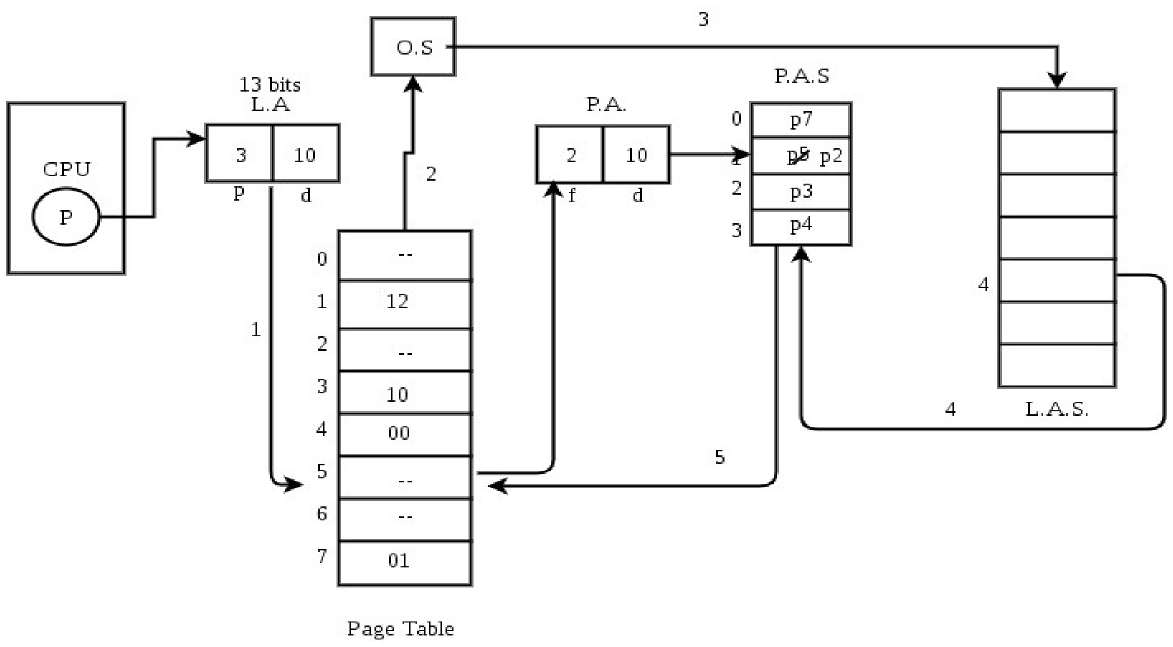
3. The OS will search for the required page in the logical address space.

4. The required page will be brought from logical address space to physical address space. The page replacement algorithms are used for the decision making of replacing the page in physical address space.

5. The page table will updated accordingly.

6. The signal will be sent to the CPU to continue the program execution and it will place the process back into ready state.

7. Hence whenever a page fault occurs these steps are followed by the operating system and the required page is brought into memory.



PAGE FAULT –

A page fault happens when a running program accesses a memory page that is mapped into the virtual address space, but not loaded in physical memory.

Since actual physical memory is much smaller than virtual memory, page faults happen. In case of page fault, Operating System might have to replace one of the existing pages with the newly needed page. Different page replacement algorithms suggest different ways to decide which page to replace. The target for all algorithms is to reduce the number of page faults.

PAGE FAULT SERVICE TIME :

The time taken to service the page fault is called as page fault service time. The page fault service time includes the time taken to perform all the above six steps.

Let Main memory access time is: m

Page fault service time is: s

Page fault rate is : p

Then, Effective memory access time = (p*s) + (1-p)*m

Steps for handling page fault

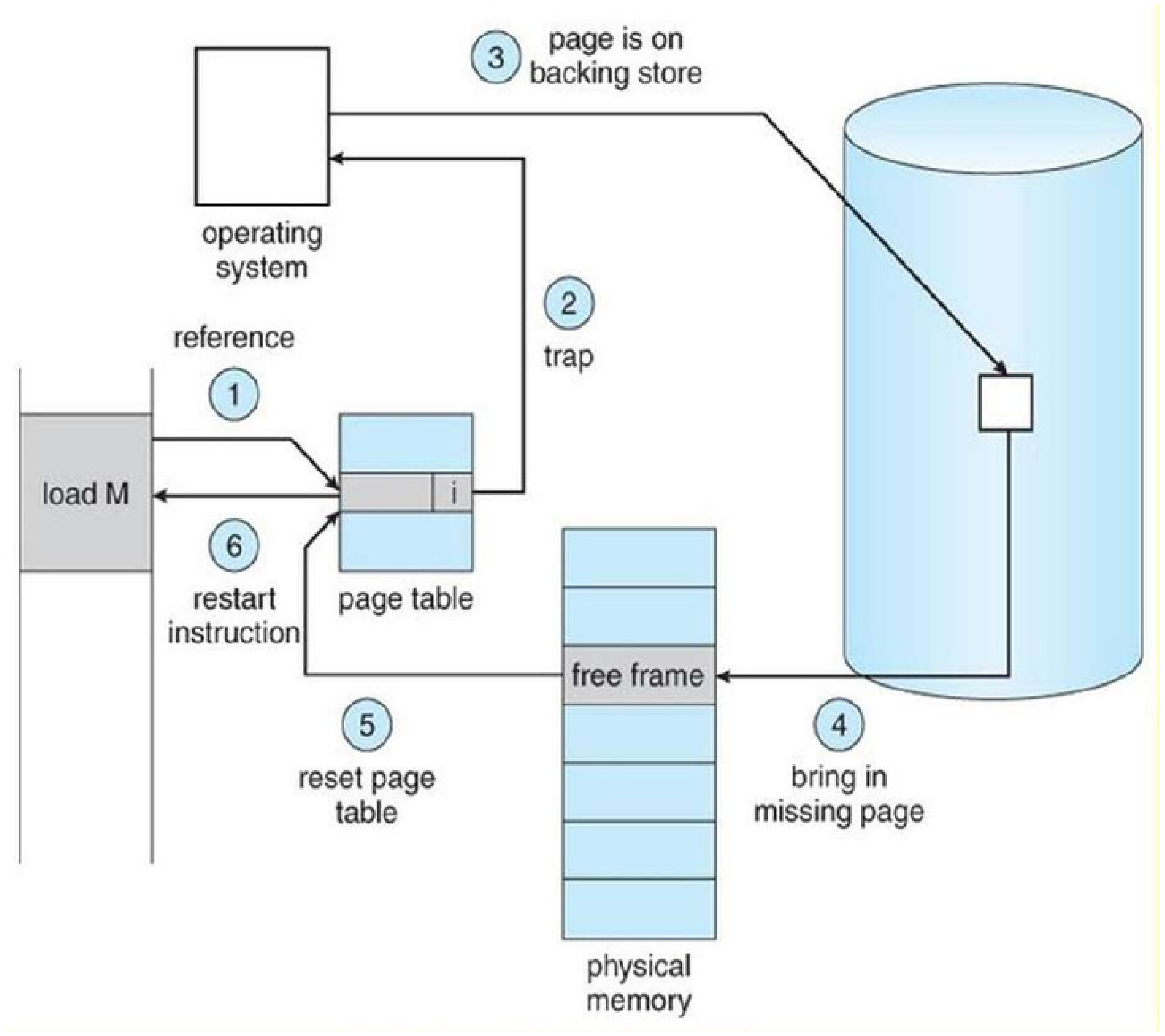
• The basic idea behind paging is that when a process is swapped in, the pager only loads into memory those pages that it expects the process to need (right away.)

• Pages that are not loaded into memory are marked as invalid in the page table, using the invalid bit. (The rest of the page table entry may either be blank or contain information about where to find the swapped-out page on the hard drive.)

• If the process only ever accesses pages that are loaded in memory (***memory resident*** pages), then the process runs exactly as if all the pages were loaded in to memory.

- On the other hand, if a page is needed that was not originally loaded up, then a ***page fault trap*** is generated, which must be handled in a series of **steps**:

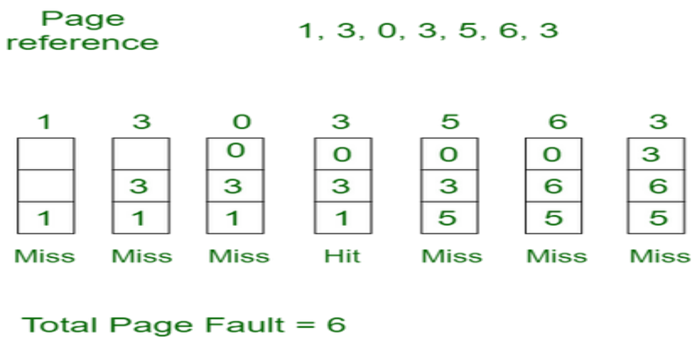
1. The memory address requested is first checked, to make sure it was a valid memory request.
2. If the reference was invalid, the process is terminated. Otherwise, the page must be paged in.
3. A free frame is located, possibly from a free-frame list.
4. A disk operation is scheduled to bring in the necessary page from disk. (This will usually block the process on an I/O wait, allowing some other process to use the CPU in the meantime.)
5. When the I/O operation is complete, the process's page table is updated with the new frame number, and the invalid bit is changed to indicate that this is now a valid page reference.
6. The instruction that caused the page fault must now be restarted from the beginning, (as soon as this process gets another turn on the CPU.)



PAGE REPLACEMENT ALGORITHMS :

First In First Out (FIFO) –
This is the simplest page replacement algorithm. In this algorithm, the operating system keeps track of all pages in the memory in a queue, the oldest page is in the front of the queue. When a page needs to be replaced page in the front of the queue is selected for removal.

Example-1 Consider page reference string 1, 3, 0, 3, 5, 6 with 3 page frames. Find number of page faults.

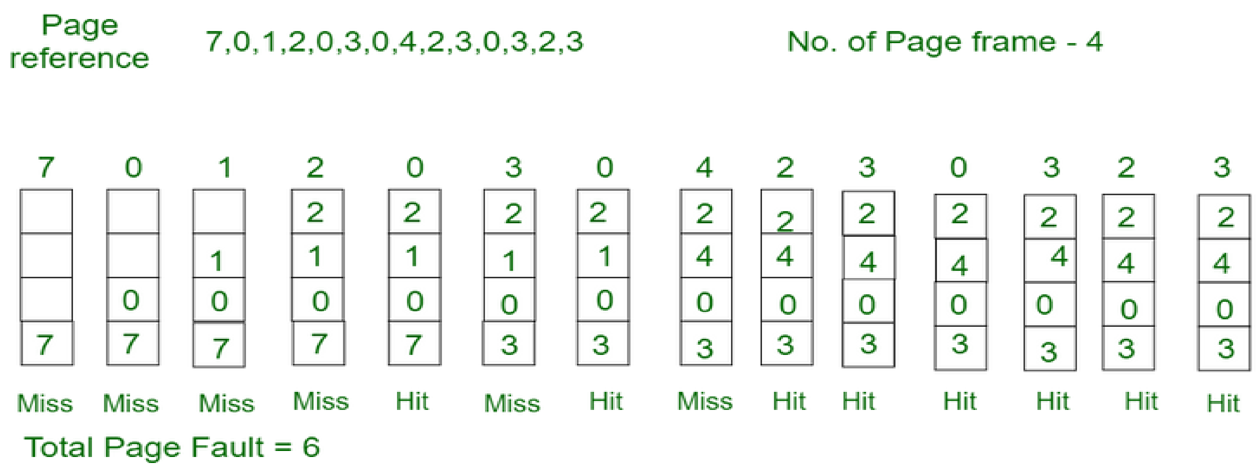


Initially all slots are empty, so when 1, 3, 0 came they are allocated to the empty slots —> **3 Page Faults.**
when 3 comes, it is already in memory so —> **0 Page Faults.**
Then 5 comes, it is not available in memory so it replaces the oldest page slot i.e 1. —>**1 Page Fault.**
6 comes, it is also not available in memory so it replaces the oldest page slot i.e 3 —>**1 Page Fault.**
Finally when 3 come it is not avilable so it replaces 0 **1 page fault**

Optimal Page replacement (OPR)–

In this algorithm, pages are replaced which would not be used for the longest duration of time in the future.

Example-2: Consider the page references 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, with 4 page frame. Find number of page fault.



Initially all slots are empty, so when 7 0 1 2 are allocated to the empty slots —> **4 Page faults**
0 is already there so —> **0 Page fault.**
when 3 came it will take the place of 7 because it is not used for the longest duration of time in the future.—>**1 Page fault.**
0 is already there so —> **0 Page fault.**.
4 will takes place of 1 —> **1 Page Fault.**

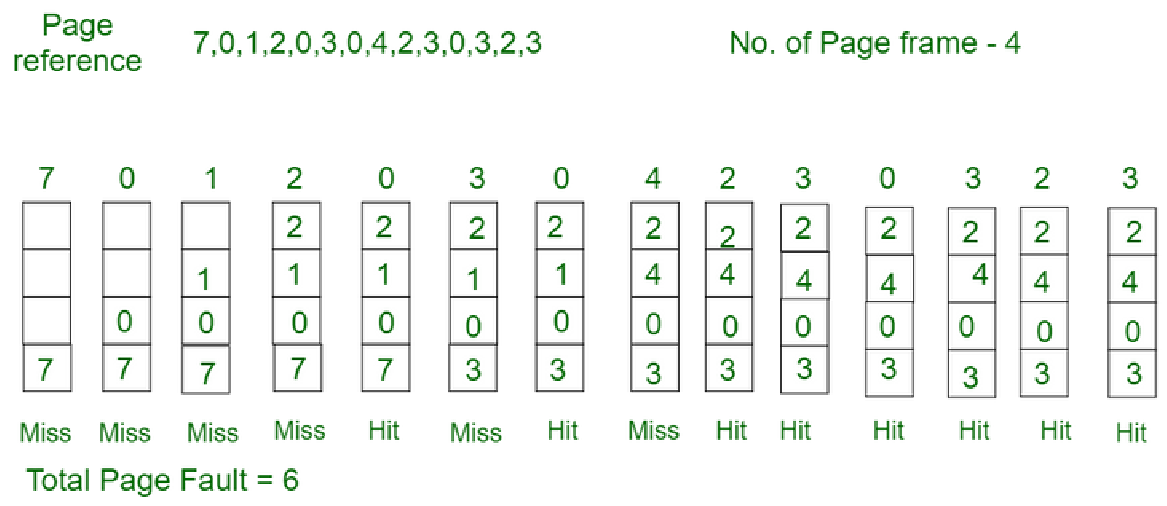
Now for the further page reference string —> **0 Page fault** because they are already available in the memory.

Optimal page replacement is perfect, but not possible in practice as the operating system cannot know future requests. The use of Optimal Page replacement is to set up a benchmark so that other replacement algorithms can be analyzed against it.

Least Recently Used (LRU) –

In this algorithm page will be replaced which is least recently used.

Example-3 Consider the page reference string 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2 with 4 page frames. Find number of page faults



Here LRU has same number of page fault as optimal but it may differ according to question.

Initially all slots are empty, so when 7 0 1 2 are allocated to the empty slots —> **4 Page faults**

0 is already their so —> **0 Page fault.**
when 3 came it will take the place of 7 because it is least recently used —>**1 Page fault**
0 is already in memory so —> **0 Page fault**.
4 will takes place of 1 —> **1 Page Fault**

Now for the further page reference string —> **0 Page fault** because they are already available in the memory.

CHAPTER-4

DEVICE MANAGEMENT

Device management in operating system known as the management of the I/O devices such as a keyboard, magnetic tape, disk, printer, microphone, USB ports, scanner, etc.as well as the supporting units like control channels.

Technique of device management in the operating system

An operating system or the OS manages communication with the devices through their respective drivers. The operating system component provides a uniform interface to access devices of varied physical attributes. For device management in operating system:

- Keep tracks of all devices and the program which is responsible to perform this is called I/O controller.
- Monitoring the status of each device such as storage drivers, printers and other peripheral devices.

- Enforcing preset policies and taking a decision which process gets the device when and for how long.
- Allocates and Deallocates the device in an efficient way. Deallocating them at two levels: at the process level when I/O command has been executed and the device is temporarily released, and at the job level, when the job is finished and the device is permanently released.
- Optimizes the performance of individual devices.

Types of devices

The OS peripheral devices can be categorized into 3:

- Dedicated
- Shared
- Virtual.

The differences among them are the functions of the characteristics of the devices as well as how they are managed by the Device Manager.

Dedicated devices:-

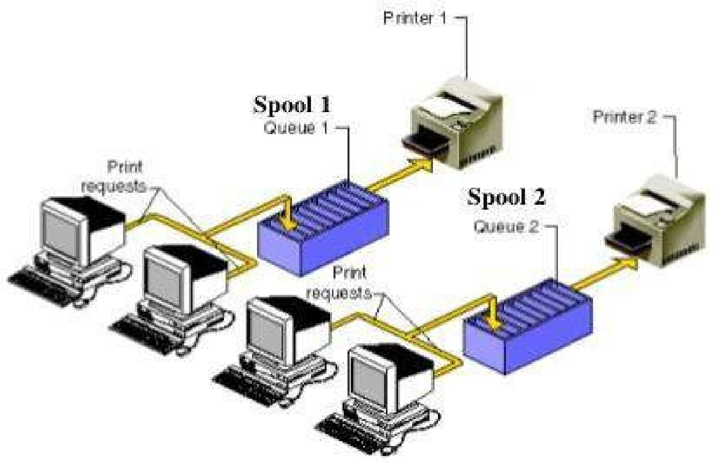
Such type of devices in the **device management in operating system** are dedicated or assigned to only one job at a time until that job releases them. Devices like printers, tape drivers, plotters etc. demand such allocation scheme since it would be awkward if several users share them at the same point of time. The disadvantages of such kind f devices s the inefficiency resulting from the allocation of the device to a single user for the entire duration of job execution even though the device is not put to use 100% of the time.

Shared devices:-

These devices can be allocated o several processes. Disk-DASD can be shared among several processes at the same time by interleaving their requests. The interleaving is carefully controlled by the Device Manager and all issues must be resolved on the basis of predetermined policies.

Virtual Devices:-

These devices are the combination of the first two types and they are dedicated devices which are transformed into shared devices. For example, a printer converted into a shareable device via spooling program which re-routes all the print requests to a disk. A print job is not sent straight to the printer, instead, it goes to the disk(spool)until it is fully prepared with all the necessary sequences and formatting, then it goes to the printers. This technique can transform one printer into several virtual printers which leads to better performance and use.



Input/Output devices:-

input/output devices are the devices that are responsible for the input/output operations in a computer system.
Basically there are following two types of input/output devices:

- Block devices
- Character devices

Block Devices

A block device stores information in block with fixed-size and own-address.

It is possible to read/write each and every block independently in case of block device.

In case of disk, it is always possible to seek another cylinder and then wait for required block to rotate under head without mattering where the arm currently is. Therefore, disk is a block addressable device.

Character Devices

A character device accepts/delivers a stream of characters without regarding to any block structure.

Character device isn't addressable.

Character device doesn't have any seek operation.

There are too many character devices present in a computer system such as printer, mice, rats, network interfaces etc. These four are the common character devices.

Input/output Devices Examples

Here are the list of some most popular and common input/output devices:

- Keyboard
- Mouse
- Monitor
- Modem
- Scanner
- Laser Printer
- Ethernet
- Disk

Storage devices

There are two types of storage devices:-

- **Volatile Storage Device –**

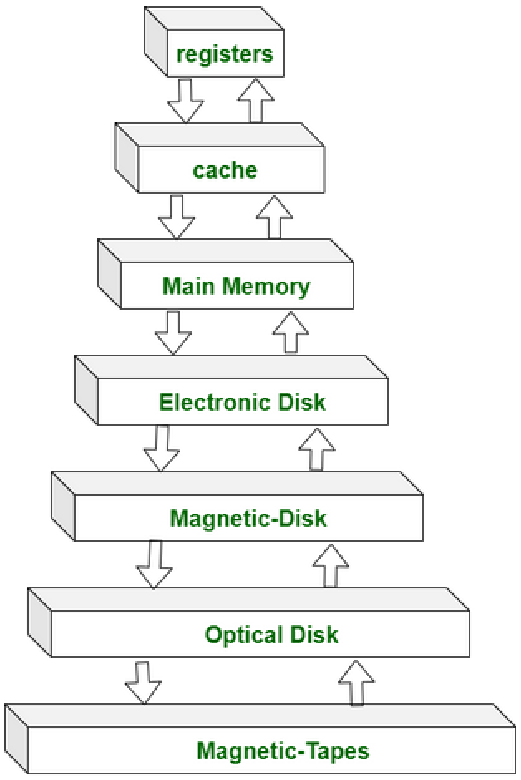
It looses its contents when the power of the device is removed.

- **Non-Volatile Storage device –**

It does not looses its contents when the power is removed. It holds all the data when the power is removed.

Secondary Storage is used as an extension of main memory. Secondary storage devices can hold the data permanently.

Storage devices consists of Registers, Cache, Main-Memory, Electronic-Disk, Magnetic-Disk, Optical-Disk, Magnetic-Tapes. Each storage system provides the basic system of storing a datum and of holding the datum until it is retrieved at a later time. All the storage devices differ in speed, cost, size and volatility. The most common Secondary-storage device is a Magnetic-disk, which provides storage for both programs and data.



Storage Device Hierarchy

Secondary storage structure

Secondary storage devices are those devices whose memory is non volatile, meaning, the stored data will be intact even if the system is turned off. Here are a few things worth noting about secondary storage.

• Secondary storage is also called auxiliary storage.

• Secondary storage is less expensive when compared to primary memory like RAMs.

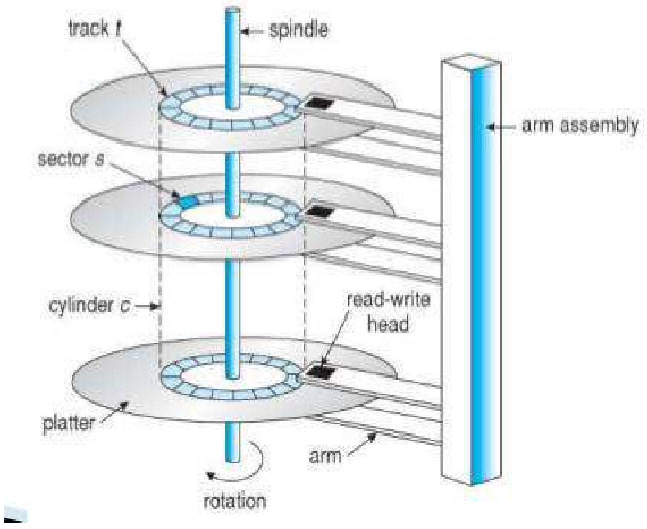
• The speed of the secondary storage is also lesser than that of primary storage.

• Hence, the data which is less frequently accessed is kept in the secondary storage.

• A few examples are magnetic disks, magnetic tapes, removable thumb drives etc

Magnetic Disk Structure

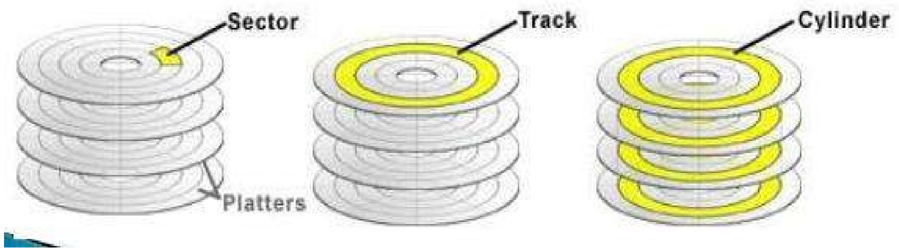
In modern computers, most of the secondary storage is in the form of magnetic disks. Hence, knowing the structure of a magnetic disk is necessary to understand how the data in the disk is accessed by the computer.



Structure of a magnetic disk

A magnetic disk contains several **platters**. Each platter is divided into circular shaped **tracks**. The length of the tracks near the center is less than the length of the tracks farther from the center. Each track is further divided into **sectors**, as shown in the figure.

Tracks of the same distance from center form a **cylinder**. A **read-write head** is used to read data from a sector of the magnetic disk.



The speed of the disk is measured as two parts:

• **Transfer rate:** This is the rate at which the data moves from disk to the computer.

• **Random access time:** It is the sum of the seek time and rotational latency.

Seek time is the time taken by the arm to move to the required track.

Rotational latency is defined as the time taken by the arm to reach the required sector in the track.

Even though the disk is arranged as sectors and tracks physically, the data is logically arranged and addressed as an array of blocks of fixed size. The size of a block can be **512** or **1024** bytes. Each logical block is mapped with a sector on the disk, sequentially. In this way, each sector in the disk will have a logical address.

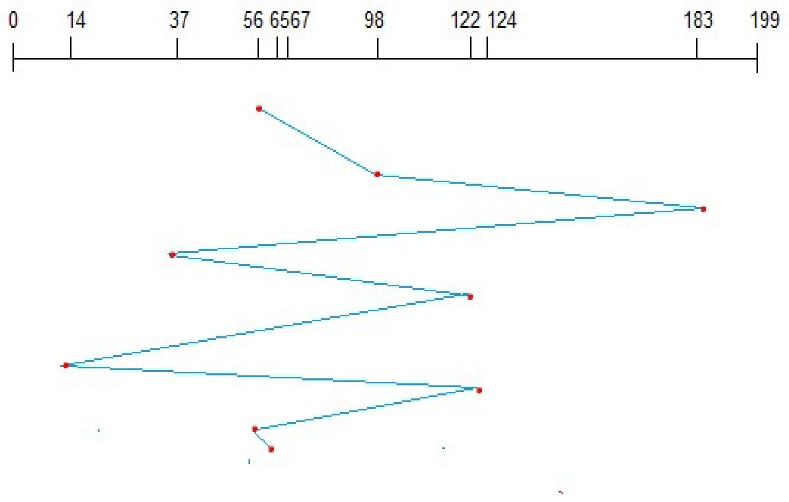
Disk Scheduling Algorithms

First Come First Serve

This algorithm performs requests in the same order asked by the system. Let's take an example where the queue has the following requests with cylinder numbers as follows:

98, 183, 37, 122, 14, 124, 65, 67

Assume the head is initially at cylinder **56**. The head moves in the given order in the queue i.e., **56→98→183→…→67**.

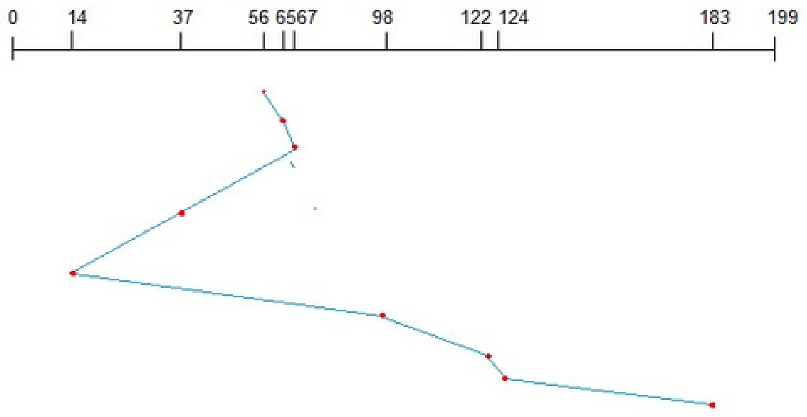


Shortest Seek Time First (SSTF)

Here the position which is closest to the current head position is chosen first. Consider the previous example where disk queue looks like,

98, 183, 37, 122, 14, 124, 65, 67

Assume the head is initially at cylinder **56**. The next closest cylinder to **56** is **65**, and then the next nearest one is **67**, then **37**, **14**, so on.

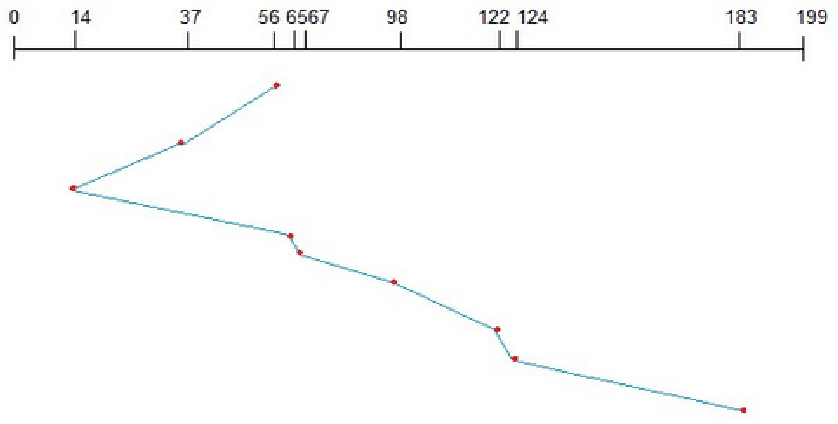


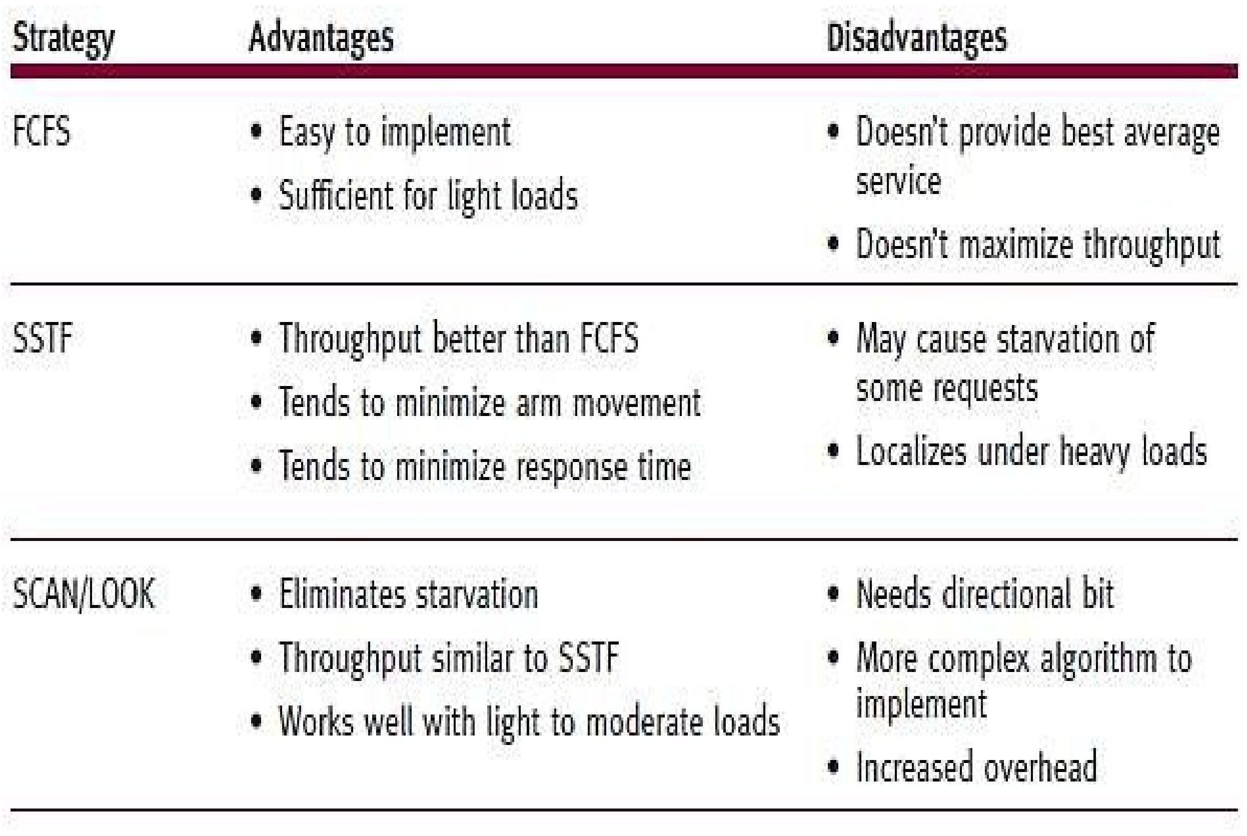
SCAN algorithm

This algorithm is also called the elevator algorithm because of it's behavior. Here, first the head moves in a direction (say backward) and covers all the requests in the path. Then it moves in the opposite direction and covers the remaining requests in the path. This behavior is similar to that of an elevator. Let's take the previous example,

98, 183, 37, 122, 14, 124, 65, 67

Assume the head is initially at cylinder **56**. The head moves in backward direction and accesses **37** and **14**. Then it goes in the opposite direction and accesses the cylinders as they come in the path.





Strategy	Advantages	Disadvantages
FCFS	• Easy to implement • Sufficient for light loads	• Doesn't provide best average service • Doesn't maximize throughput
SSTF	• Throughput better than FCFS • Tends to minimize arm movement • Tends to minimize response time	• May cause starvation of some requests • Localizes under heavy loads
SCAN/LOOK	• Eliminates starvation • Throughput similar to SSTF • Works well with light to moderate loads	• Needs directional bit • More complex algorithm to implement • Increased overhead

Management of I/O Requests

I/O traffic controller

- Watches status of devices, control units, channels

• Three main tasks

□
Determine if path available

□
If more than one path available, determine which one to select

□
If paths all busy, determine when one is available

• Maintains database containing each unit’s status and connections

I/O scheduler

o Same job as process scheduler

o Allocates devices, control units, and channels

o If requests greater than available paths

• Decides which request to satisfy first: based on different criteria

o In many systems

• I/O requests not preempted

o For some systems

• Allow preemption with I/O request subdivided

• Allow preferential treatment for high-priority requests

I/O device handler

• Performs actual data transfer

o Processes device interrupts

o Handles error conditions

o Provides detailed scheduling algorithms

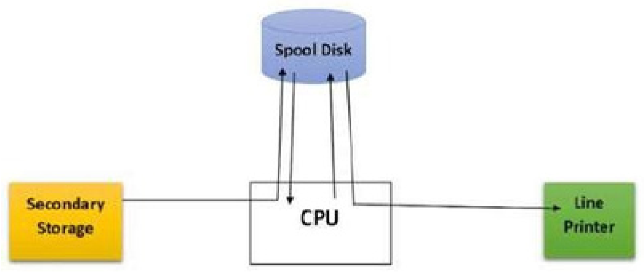
• Device dependent

~~• Each I/O device type has its own device handler algorit~~hm

SPOOLING(simultaneous peripheral operations on-line)

• It is a kind of buffering mechanism or a process in which data is temporarily held to be used and executed by a device, program or the system. Data is sent to and stored in memory or other volatile storage until the program or computer requests it for execution.

- In a computer system peripheral equipment, such as printers and punch card readers etc. (batch processing), are very slow relative to the performance of the rest of the system. Getting input and output from the system was quickly seen to be a bottleneck. Here comes the need for spool.
- Spooling works like a typical request queue where data, instructions and processes from multiple sources are accumulated for execution later on. Generally, it is maintained on computer's physical memory, buffers or the I/O device-specific interrupts. The spool is processed in FIFO manner i.e. whatever first instruction is there in the queue will be popped and executed.



<u>Applications/Implementations of Spool</u>:

The most common can be found in I/O devices like keyboard printers and mouse. For example, In printer, the documents/files that are sent to the printer are first stored in the memory or the printer spooler. Once the printer is ready, it fetches the data from the spool and prints it.

Even experienced a situation when suddenly for some seconds your mouse or keyboard stops working? Meanwhile, we usually click again and again here and there on the screen to check if its working or not. When it actually starts working, what and wherever we pressed during its hang state gets executed very fast because all the instructions got stored in the respective device's spool.

2) A batch processing system uses spooling to maintain a queue of ready-to-run jobs which can be started as soon as the system has the resources to process them.

3) Spooling is capable of overlapping I/O operation for one job with processor operations for another job. i.e. multiple processes can write documents to a print queue without waiting and resume with their work.

4) E-mail: an email is delivered by a MTA (Mail Transfer Agent) to a temporary storage area where it waits to be picked up by the MA (Mail User Agent)

5) Can also be used for generating Banner pages (these are the pages used in computerized printing in order to separate documents from each other and to identify e.g. the originator of

the print request by username, an account number or a bin for pickup. Such pages are used in office environments where many people share the small number of available resources).

Difference between Spooling and Buffering

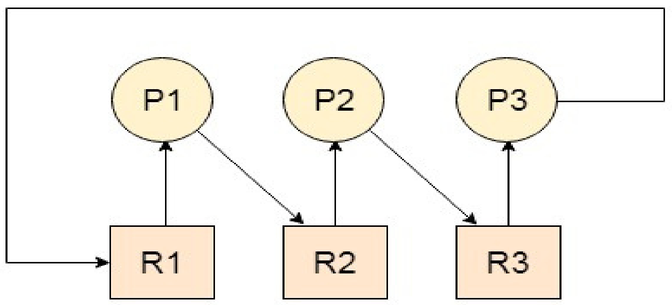
	SPOOLING	BUFFERING
Basic Difference	It overlap the input/output of one job with the execution of another job.	It overlaps the input/output of one job with the execution of the same job.
Full form (stands for)	Simultaneous peripheral operation online	No full form
Efficiency	Spooling is more efficient than buffering.	Buffering is less efficient than spooling.
Consider Size	It considers disk as a huge spool or buffer.	Buffer is a limited area in main memory.

CHAPTER-5 DEADLOCKS

- Every process needs some resources to complete its execution. However, the resource is granted in a sequential order.
- The process requests for some resource.
- OS grant the resource if it is available otherwise let the process waits.
- The process uses it and release on the completion.
- A Deadlock is a situation where each of the computer process waits for a resource which is being assigned to some another process. In this situation, none of the process gets executed since the resource it needs, is held by some other process which is also waiting for some other resource to be released.
- Let us assume that there are three processes P1, P2 and P3. There are three different resources R1, R2 and R3. R1 is assigned to P1, R2 is assigned to P2 and R3 is assigned to P3.
After some time, P1 demands for R1 which is being used by P2. P1 halts its execution since it can't complete without R2. P2 also demands for R3 which is being used by P3. P2

also stops its execution because it can't continue without R3. P3 also demands for R1 which is being used by P1 therefore P3 also stops its execution.

- In this scenario, a cycle is being formed among the three processes. None of the process is progressing and they are all waiting. The computer becomes unresponsive since all the processes got blocked.



Difference between Starvation and Deadlock

Sr.	Deadlock	Starvation
1	Deadlock is a situation where no process got blocked and no process proceeds	Starvation is a situation where the low priority process got blocked and the high priority processes proceed.
2	Deadlock is an infinite waiting.	Starvation is a long waiting but not infinite.
3	Every Deadlock is always a starvation.	Every starvation need not be deadlock.
4	The requested resource is blocked by the other process.	The requested resource is continuously be used by the higher priority processes.
5	Deadlock happens when Mutual exclusion, hold and wait, No preemption and circular wait occurs simultaneously.	It occurs due to the uncontrolled priority and resource management.

Necessary conditions for Deadlocks

- **Mutual Exclusion**
  - A resource can only be shared in mutually exclusive manner. It implies, if two process cannot use the same resource at the same time.
- **Hold and Wait**

 A process waits for some resources while holding another resource at the same time.

No pre-emption

 The process which once scheduled will be executed till the completion. No other process can be scheduled by the scheduler meanwhile.

Circular Wait

 All the processes must be waiting for the resources in a cyclic manner so that the last process is waiting for the resource which is being held by the first process.

Strategies for handling Deadlock

1. **Deadlock Ignorance**

 Deadlock Ignorance is the most widely used approach among all the mechanism. This is being used by many operating systems mainly for end user uses. In this approach, the Operating system assumes that deadlock never occurs. It simply ignores deadlock. This approach is best suitable for a single end user system where User uses the system only for browsing and all other normal stuff.

 There is always a tradeoff between Correctness and performance. The operating systems like Windows and Linux mainly focus upon performance. However, the performance of the system decreases if it uses deadlock handling mechanism all the time if deadlock happens 1 out of 100 times then it is completely unnecessary to use the deadlock handling mechanism all the time.

 In these types of systems, the user has to simply restart the computer in the case of deadlock. Windows and Linux are mainly using this approach.

2. Deadlock prevention

 Deadlock happens only when Mutual Exclusion, hold and wait, No preemption and circular wait holds simultaneously. If it is possible to violate one of the four conditions at any time then the deadlock can never occur in the system.

 The idea behind the approach is very simple that we have to fail one of the four conditions but there can be a big argument on its physical implementation in the system.

3. Deadlock avoidance

 In deadlock avoidance, the operating system checks whether the system is in safe state continues until the system is in safe state. Once the system moves to unsafe state, the OS has to backtrack one step.

 In simple words, The OS reviews each allocation so that the allocation doesn't cause the deadlock in the system.

4. Deadlock detection and recovery

This approach let the processes fall in deadlock and then periodically check whether deadlock occur in the system or not. If it occurs then it applies some of the recovery methods to the system to get rid of deadlock.

DEADLOCK PREVENTION

� If we simulate deadlock with a table which is standing on its four legs then we can also simulate four legs with the four conditions which when occurs simultaneously, cause the deadlock.

� However, if we break one of the legs of the table then the table will fall definitely. The same happens with deadlock, if we can be able to violate one of the four necessary conditions and don't let them occur together then we can prevent the deadlock.

� Let's see how we can prevent each of the conditions.

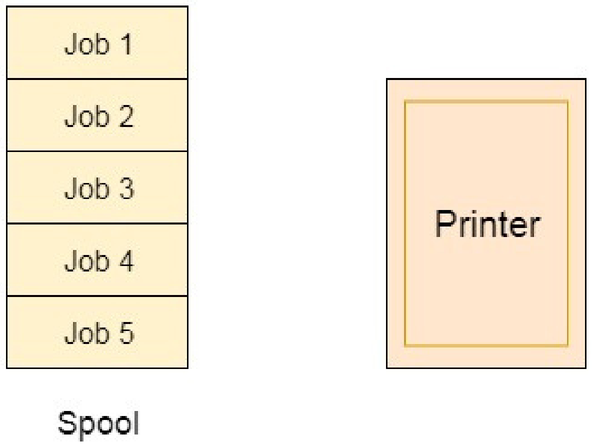
1. Mutual Exclusion

� Mutual section from the resource point of view is the fact that a resource can never be used by more than one process simultaneously which is fair enough but that is the main reason behind the deadlock. If a resource could have been used by more than one process at the same time then the process would have never been waiting for any resource.

� However, if we can be able to violate resources behaving in the mutually exclusive manner then the deadlock can be prevented.

Spooling

� For a device like printer, spooling can work. There is a memory associated with the printer which stores jobs from each of the process into it. Later, Printer collects all the jobs and print each one of them according to FCFS. By using this mechanism, the process doesn't have to wait for the printer and it can continue whatever it was doing. Later, it collects the output when it is produced.



Although, Spooling can be an effective approach to violate mutual exclusion but it suffers from two kinds of problems.

- This cannot be applied to every resource.
- After some point of time, there may arise a race condition between the processes to get space in that spool.

We cannot force a resource to be used by more than one process at the same time since it will not be fair enough and some serious problems may arise in the performance. Therefore, we cannot violate mutual exclusion for a process practically.

2. Hold and Wait

Hold and wait condition lies when a process holds a resource and waiting for some other resource to complete its task. Deadlock occurs because there can be more than one process which are holding one resource and waiting for other in the cyclic order.
However, we have to find out some mechanism by which a process either doesn't hold any resource or doesn't wait. That means, a process must be assigned all the necessary resources before the execution starts. A process must not wait for any resource once the execution has been started.
!(Hold and wait) = !hold or !wait (negation of hold and wait is, either you don't hold or you don't wait)
This can be implemented practically if a process declares all the resources initially. However, this sounds very practical but can't be done in the computer system because a process can't determine necessary resources initially.
Process is the set of instructions which are executed by the CPU. Each of the instruction may demand multiple resources at the multiple times. The need cannot be fixed by the OS.

The problem with the approach is:
• Practically not possible.
• Possibility of getting starved will be increases due to the fact that some process may hold a resource for a very long time.

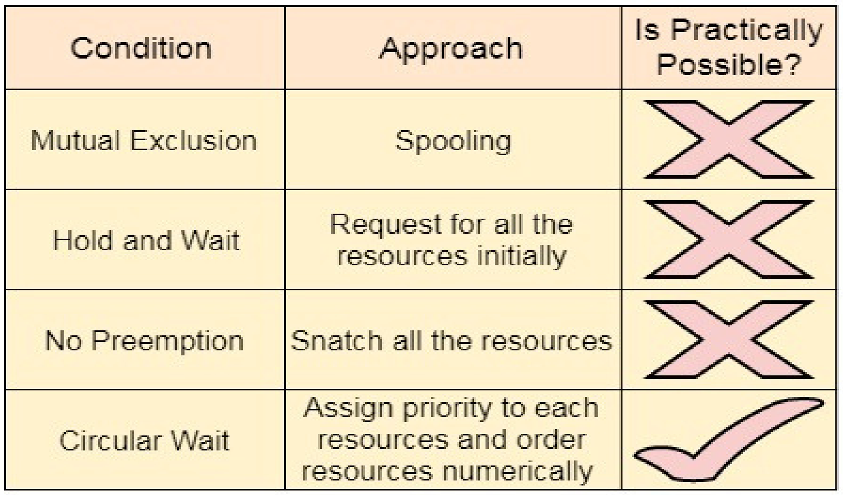
3. No Preemption

- Deadlock arises due to the fact that a process can't be stopped once it starts. However, if we take the resource away from the process which is causing deadlock then we can prevent deadlock.
- This is not a good approach at all since if we take a resource away which is being used by the process then all the work which it has done till now can become inconsistent.
- Consider a printer is being used by any process. If we take the printer away from that process and assign it to some other process then all the data which has been printed can become inconsistent and ineffective and also the fact that the process can't start printing again from where it has left which causes performance inefficiency.

4. Circular Wait

To violate circular wait, we can assign a priority number to each of the resource. A process can't request for a lesser priority resource. This ensures that not a single process can request a resource which is being utilized by some other process and no cycle will be formed.

Among all the methods, violating Circular wait is the only approach that can be implemented practically.



Condition	Approach	Is Practically Possible?
Mutual Exclusion	Spooling	✗
Hold and Wait	Request for all the resources initially	✗
No Preemption	Snatch all the resources	✗
Circular Wait	Assign priority to each resources and order resources numerically	✓

BANKER'S ALGORITHM

◦ Banker's algorithm is a **deadlock avoidance algorithm**. It is named so because this algorithm is used in banking systems to determine whether a loan can be granted or not.

◦ Consider there are n account holders in a bank and the sum of the money in all of their accounts is S. Every time a loan has to be granted by the bank, it subtracts the **loan amount** from the **total money** the bank has. Then it checks if that difference is greater than S. It is done because, only then, the bank would have enough money even if all the n account holders draw all their money at once.

◦ Banker's algorithm works in a similar way in computers.

◦ Whenever a new process is created, it must specify the maximum instances of each resource type that it needs, exactly.

Characteristics of Banker's Algorithm

◦ The characteristics of Banker's algorithm are as follows:

◦ If any process requests for a resource, then it has to wait.

◦ This algorithm consists of advanced features for maximum resource allocation.

◦ There are limited resources in the system we have.

◦ In this algorithm, if any process gets all the needed resources, then it is that it should return the resources in a restricted period.

◦ Various resources are maintained in this algorithm that can fulfil the needs of at least one client.

◦ Let us assume that there are n processes and m resource types.

DATA STRUCTURES USED TO IMPLEMENT THE BANKER'S ALGORITHM

Some data structures that are used to implement the banker's algorithm are:

◦ 1. **Available**

◦ It is an **array** of length m. It represents the number of available resources of each type. If Available[j] = k, then there are k instances available, of resource type Rj.

◦ 2. **Max**

◦ It is an n x m matrix which represents the maximum number of instances of each resource that a process can request. If Max[i][j] = k, then the process Pi can request atmost k instances of resource type Rj.

⬨ 3. **Allocation**

⬨ It is an n x m matrix which represents the number of resources of each type currently allocated to each process. If Allocation[i][j] = k, then process Pi is currently allocated k instances of resource type Rj.

⬨ 4. **Need**

⬨ It is a two-dimensional array. It is an n x m matrix which indicates the remaining resource needs of each process. If Need[i][j] = k, then process Pi may need k more instances of resource type Rj to complete its task.

⬨ Need[i][j] = Max[i][j] - Allocation [i][j]

Banker's algorithm comprises of two algorithms:

1. Safety algorithm

2. Resource request algorithm

SAFETY ALGORITHM

A safety algorithm is an algorithm used to find whether or not a system is in its safe state. The algorithm is as follows:

1. Let Work and Finish be vectors of length **m** and **n**, respectively. Initially,Work = Available Finish[i] =false for i = 0, 1, ... , n - 1. This means, initially, no process has finished and the number of available resources is represented by the **Available** array.

2. Find an index **i** such that both Finish[i] ==false Needi <= Work If there is no such i present, then proceed to step 4.

3. It means, we need to find an unfinished process whose needs can be satisfied by the available resources. If no such process exists, just go to step 4.

4. Perform the following:Work = Work + Allocationi Finish[i] = true Go to step 2.

5. When an unfinished process is found, then the resources are allocated and the process is marked finished. And then, the loop is repeated to check the same for all other processes.

6. If Finish[i] == true for all i, then the system is in a safe state.That means if all processes are finished, then the system is in safe state.

7. This algorithm may require an order of **mxn^2 operations** in order to determine whether a state is safe or not.

Example: Let us consider the following snapshot for understanding the banker's algorithm:

Processes Allocation A B C Max A B C Available A B C

P0 1 1 2 4 3 3 2 1 0

P1 2 1 2 3 2 2

P2 4 0 1 9 0 2

P3 0 2 0 7 5 3

P4 1 1 2 1 1 2

1. calculate the content of the need matrix?

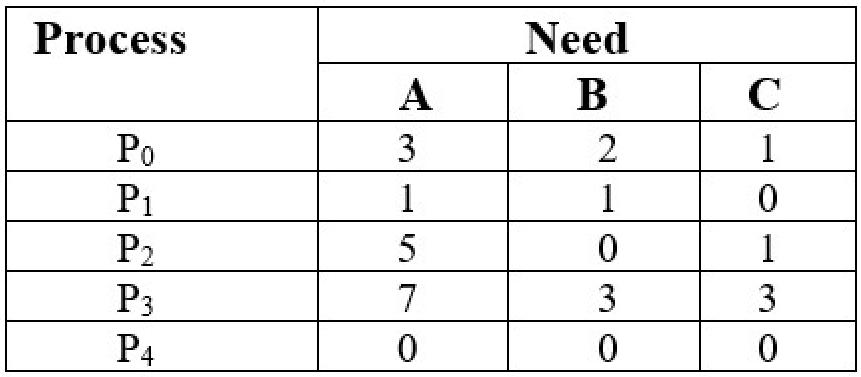
2. Check if the system is in a safe state?

3. Determine the total sum of each type of resource?

Solution:

1. The Content of the need matrix can be calculated by using the formula given below:

Need = Max – Allocation



Process	Need		
	A	B	C
P_0	3	2	1
P_1	1	1	0
P_2	5	0	1
P_3	7	3	3
P_4	0	0	0

2. Let us now check for the safe state.

Safe sequence

1.For process P0, Need = (3, 2, 1) and

Available = (2, 1, 0)

Need <=Available = False

So, the system will move to the next process.

2. For Process P1, Need = (1, 1, 0)

Available = (2, 1, 0)

Need <= Available = True

Request of P1 is granted.

Available = Available +Allocation

= (2, 1, 0) + (2, 1, 2)

= (4, 2, 2) (New Available)

3. For Process P2, Need = (5, 0, 1)

Available = (4, 2, 2)

Need <=Available = False

So, the system will move to the next process.

4. For Process P3, Need = (7, 3, 3)

Available = (4, 2, 2)

Need <=Available = False

So, the system will move to the next process.

5. For Process P4, Need = (0, 0, 0)

Available = (4, 2, 2)

Need <= Available = True

Request of P4 is granted.

Available = Available + Allocation

= (4, 2, 2) + (1, 1, 2)

= (5, 3, 4) now, (New Available)

6. Now again check for Process P2, Need = (5, 0, 1)

Available = (5, 3, 4)

Need <= Available = True

Request of P2 is granted.

Available = Available + Allocation

= (5, 3, 4) + (4, 0, 1)

= (9, 3, 5) now, (New Available)

7. Now again check for Process P3, Need = (7, 3, 3)

Available = (9, 3, 5)

Need <=Available = True

The request for P3 is granted.

Available = Available +Allocation

= (9, 3, 5) + (0, 2, 0) = (9, 5, 5)

8. Now again check for Process P0, = Need (3, 2, 1)

= Available (9, 5, 5)

Need <= Available = True

So, the request will be granted to P0.

Safe sequence: < P1, P4, P2, P3, P0>

The system allocates all the needed resources to each process. So, we can say that the system is in a safe state.

3. The total amount of resources will be calculated by the following formula:

The total amount of resources= sum of columns of allocation + Available

= [8 5 7] + [2 1 0] = [10 6 7]

CHAPTER-6

FILE MANAGEMENT

File Systems

- A file is a collection of correlated information which is recorded on secondary or non-volatile storage like magnetic disks, optical disks, and tapes. It is a method of data collection that is used as a medium for giving input and receiving output from that program.

- In general, a file is a sequence of bits, bytes, or records whose meaning is defined by the file creator and user. Every File has a logical location where they are located for storage and retrieval.
- File system is the part of the operating system which is responsible for file management. It provides a mechanism to store the data and access to the file contents including data and programs. Some Operating systems treats everything as a file for example Ubuntu.

Objective of File management System

- It provides I/O support for a variety of storage device types.
- Minimizes the chances of lost or destroyed data
- Helps OS to standardized I/O interface routines for user processes.
- It provides I/O support for multiple users in a multiuser systems environment.

Properties of a File System

- Files are stored on disk or other storage and do not disappear when a user logs off.
- Files have names and are associated with access permission that permits controlled sharing.
- Files could be arranged or more complex structures to reflect the relationship between them.

File structure

A File Structure needs to be predefined format in such a way that an operating system understands . It has an exclusively defined structure, which is based on its type.

Three types of files structure in OS:

- A text file: It is a series of characters that is organized in lines.
- An object file: It is a series of bytes that is organized into blocks.
- A source file: It is a series of functions and processes.

File Attributes

A file has a name and data. Moreover, it also stores meta information like file creation date and time, current size, last modified date, etc. All this information is called the attributes of a file system.

Here, are some important File attributes used in OS:

▹ **Name:** It is the only information stored in a human-readable form.

▹ **Identifier**: Every file is identified by a unique tag number within a file system known as an identifier.

▹ **Location:** Points to file location on device.

▹ **Type:** This attribute is required for systems that support various types of files.

▹ **Size**. Attribute used to display the current file size.

▹ **Protection**. This attribute assigns and controls the access rights of reading, writing, and executing the file.

▹ **Time, date and security:** It is used for protection, security, and also used for monitoring

File Type

It refers to the ability of the operating system to differentiate various types of files like text files, binary, and source files. However, Operating systems like MS_DOS and UNIX has the following type of files:

⊳ **Character Special File**

It is a hardware file that reads or writes data character by character, like mouse, printer, and more.

⊳ **Ordinary files**

▹ These types of files stores user information.

▹ It may be text, executable programs, and databases.

▹ It allows the user to perform operations like add, delete, and modify.

⊳ **Directory Files**

Directory contains files and other related information about those files. Its basically a folder to hold and organize multiple files.

▷ **Special Files**

These files are also called device files. It represents physical devices like printers, disks, networks, flash drive, etc.

Functions of File

▷ Create file, find space on disk, and make an entry in the directory.

▷ Write to file, requires positioning within the file

▷ Read from file involves positioning within the file

▷ Delete directory entry, regain disk space.

▷ Reposition: move read/write position.

Commonly used terms in File systems

▷ **Field:**

This element stores a single value, which can be static or variable length.

▷ **DATABASE:**

Collection of related data is called a database. Relationships among elements of data are explicit.

▷ **FILES:**

Files is the collection of similar record which is treated as a single entity.

▷ **RECORD:**

A Record type is a complex data type that allows the programmer to create a new data type with the desired column structure. Its groups one or more columns to form a new data type. These columns will have their own names and data type.

File Access Methods

~~File access is a proc~~ess that determines the way that files are accessed and read into memory. Generally, a single access method is always supported by operating systems. Though there are some operating system which also supports multiple access methods.

Three file access methods are:

▹ Sequential access

▹ Direct random access

▹ Index sequential access

Sequential Access

⊳ In this type of file access method, records are accessed in a certain pre-defined sequence. In the sequential access method, information stored in the file is also processed one by one. Most compilers access files using this access method.

Random Access

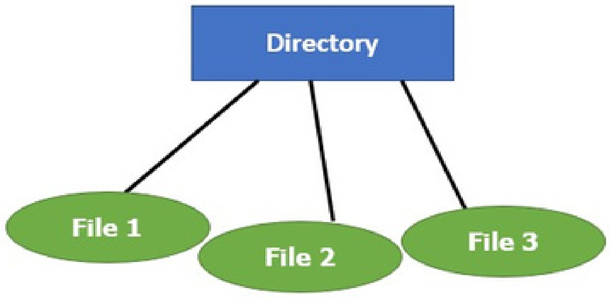
The random access method is also called direct random access. This method allow accessing the record directly. Each record has its own address on which can be directly accessed for reading and writing.

Index sequential access

⊳ This type of accessing method is based on simple sequential access. In this access method, an index is built for every file, with a direct pointer to different memory blocks. In this method, the Index is searched sequentially, and its pointer can access the file directly. Multiple levels of indexing can be used to offer greater efficiency in access. It also reduces the time needed to access a single record.

File Directories

⊳ A single directory may or may not contain multiple files. It can also have sub-directories inside the main directory. Information about files is maintained by Directories. In Windows OS, it is called folders.



Following is the information which is maintained in a directory:

▹ **Name** The name which is displayed to the user.

▹ **Type**: Type of the directory.

▹ **Position**: Current next-read/write pointers.

▹ **Location**: Location on the device where the file header is stored.

▹ **Size** : Number of bytes, block, and words in the file.

▹ **Protection**: Access control on read/write/execute/delete.

▹ **Usage**: Time of creation, access, modification

Operation performed on directory are:

- Search for a file
- Create a file
- Delete a file
- List a directory
- Rename a file
- Traverse the file system

File types- name, extension

File Type	Usual extension	Function
Executable	exe, com, bin or none	ready-to-run machine-language program
Object	obj, o	complied, machine language, not linked
Source code	c. p, pas, 177, asm, a	source code in various languages
Batch	bat, sh	Series of commands to be executed
Text	txt, doc	textual data documents
Word processor	doc,docs, tex, rrf, etc.	various word-processor formats
Library	lib, h	libraries of routines
Archive	arc, zip, tar	related files grouped into one file, sometimes compressed.

FILE PROTECTION

In computer systems, a lot of user's information is stored, the objective of the operating system is to keep safe the data of the user from the improper access to the system. Protection can be provided in number of ways. For a single laptop system, we might provide protection by locking the computer in a desk drawer or file cabinet. For multi-user systems, different mechanisms are used for the protection.

Types of Access :

The files which have direct access of the any user have the need of protection. The files which are not accessible to other users doesn't require any kind of protection. The mechanism of the protection provide the facility of the controlled access by just limiting the types of access to the file. Access can be given or not given to any user depends on several factors, one of which is the type of access required. Several different types of operations can be controlled:

▹ **Read**

Reading from a file.

▹ **Write**

Writing or rewriting the file.

▹ **Execute**

Loading the file and after loading the execution process starts.

▹ **Append**

Writing the new information to the already existing file, editing must be end at the end of the existing file.

▹ **Delete**

Deleting the file which is of no use and using its space for the another data.

▹ **List**

List the name and attributes of the file.

Operations like renaming, editing the existing file, copying; these can also be controlled. There are many protection mechanism. each of them mechanism have different advantages and disadvantages and must be appropriate for the intended application.

Access Control :

▸ There are different methods used by different users to access any file. The general way of protection is to associate *identity-dependent access* with all the files and directories an list called access-control list (ACL) which specify the names of the users and the types of access associate with each of the user. The main problem with the access list is their length. If we want to allow everyone to read a file, we must list all the users with the read access. This technique has two undesirable consequences:

- Constructing such a list may be tedious and unrewarding task, especially if we do not know in advance the list of the users in the system.

Previously, the entry of the any directory is of the fixed size but now it changes to the variable size which results in the complicates space management. These problems can be resolved by use of a condensed version of the access list. To condense the length of the access-control list, many systems recognize three classification of users in connection with each file:

Owner

Owner is the user who has created the file.

Group

A group is a set of members who has similar needs and they are sharing the same file.

Universe In the system, all other users are under the category called universe.

The most common recent approach is to combine access-control lists with the normal general owner, group, and universe access control scheme. For example: Solaris uses the three categories of access by default but allows access-control lists to be added to specific files and directories when more fine-grained access control is desired.

Other Protection Approaches:

The access to any system is also controlled by the password. If the use of password are is random and it is changed often, this may be result in limit the effective access to a file. The use of passwords has a few disadvantages:

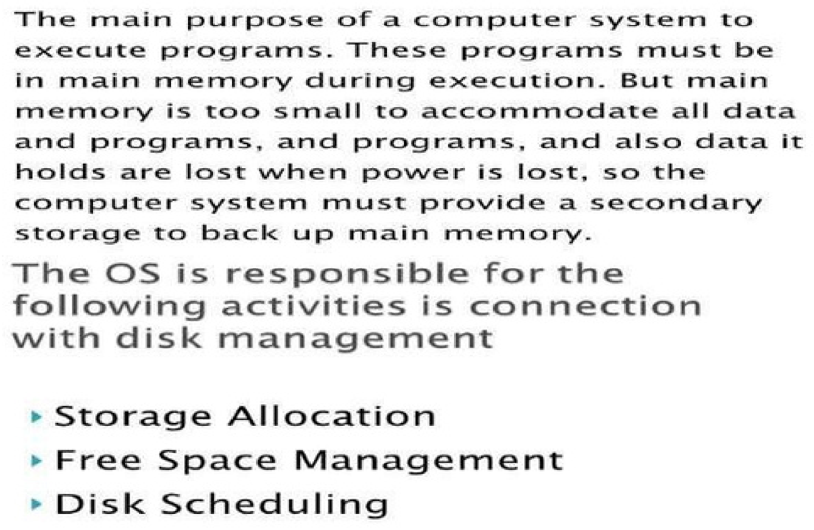
The number of passwords are very large so it is difficult to remember the large passwords.

If one password is used for all the files, then once it is discovered, all files are accessible; protection is on all-or-none basis.

SECONDARY STORAGE MANAGEMENT

Secondary storage devices are non-volatile devices where the data is stored for long-term storage. Disks are the mainly used secondary storage devices. They provide the bulk of secondary storage in operating systems today.

- The main activity that is performed in secondary storage management is disk scheduling. There are many disk scheduling algorithms. However, the important ones are FCFS scheduling, SSTF scheduling, SCAN scheduling and LOOK scheduling.



The main purpose of a computer system to execute programs. These programs must be in main memory during execution. But main memory is too small to accommodate all data and programs, and programs, and also data it holds are lost when power is lost, so the computer system must provide a secondary storage to back up main memory.

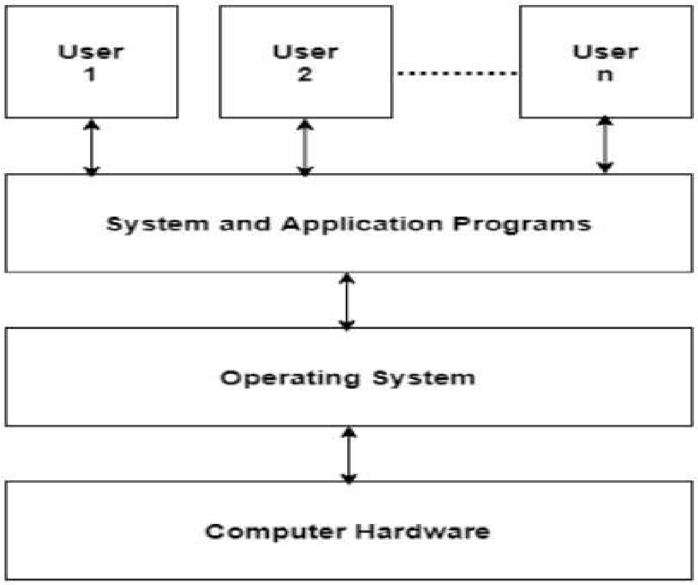
The OS is responsible for the following activities is connection with disk management

- Storage Allocation
- Free Space Management
- Disk Scheduling

CHAPTER-7

SYSTEM PROGRAMMING

There are mainly two categories of programs i.e. application programs and system programs. A diagram that demonstrates their place in the logical computer hierarchy is as follows –



Application Programs

These programs perform a particular function directly for the users. Some of the common application programs include Email, web browsers, gaming software, word processors, graphics software, media player etc.

All of these programs provide an application to the end users, so they are known as application programs. For example: a web browser is used to find information while a gaming software is used to play games.

The requests for service and application communication systems used in an application by a programmer is known as an application program interface (API).

System Programs

The system programs are used to program the operating system software. While application programs provide software that is used directly by the user, system programs provide software that are used by other systems such as SaaS applications, computational science applications etc.

The attributes of system programming are –

• Using system programming, a programmer can make assumptions about the hardware of the system that the program runs on.

• A low level programming language is used in system programming normally. This is so that the programs can operate in low resource environments easily.

• Most system programs are created to have a low runtime overhead. These programs may have small runtime library.

• Some parts of the system programs may be directly written in assembly language by the programmers.

• A debugger cannot be used on system programs mostly. This problem can be solved by running the programs in a simulated environment.

Some examples of system programs are operating system, networking system, web site server, data backup server etc.

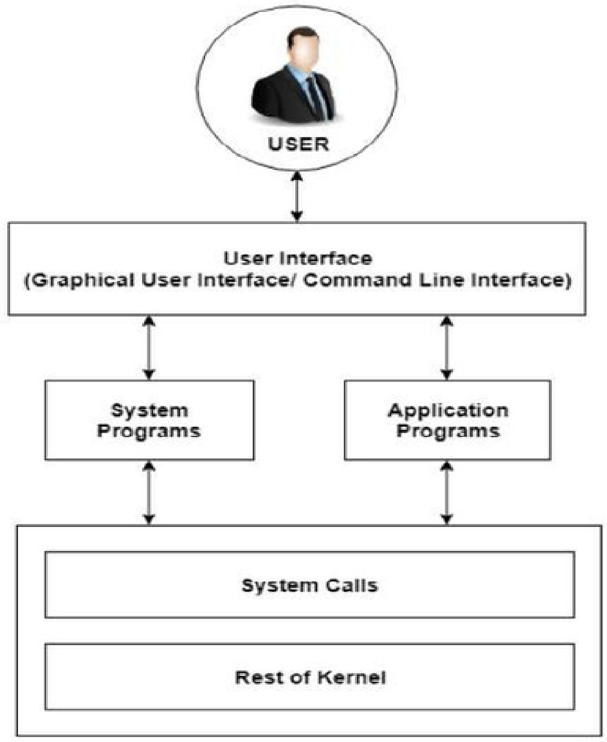
What Does System Programming Mean?

○ System programming involves designing and writing computer programs that allow the computer hardware to interface with the programmer and the user, leading to the effective execution of application software on the computer system.

○ Typical system programs include the operating system and firmware, programming tools such as compilers, assemblers, I/O routines, interpreters, scheduler, loaders and linkers as well as the runtime libraries of the computer programming languages.

○ System programs provide an environment where programs can be developed and executed. In the simplest sense, system programs also provide a bridge between the user interface and system calls. In reality, they are much more complex. For example: A compiler is a complex system program.

- The user view of the system is actually defined by system programs and not system calls because that is what they interact with and system programs are closer to the user interface.
- An image that describes system programs in the operating system hierarchy is as follows



In the above image, system programs as well as application programs form bridge between the user interface and the system calls. So, from the user view the operating system observed is actually the system programs and not the system calls.

Types of System Programs

System programs can be divided into seven parts. These are given as follows –

- **File Manipulation**

  These system programs are used to manipulate system files. This can be done using various commands like create, delete, copy, rename, print etc. These commands can create files, delete files, copy the contents of one file into another, rename files, print them etc.

- **Status Information**

  The status information system programs provide required data on the current or past status of the system. This may include the system date, system time, available memory in system, disk space, logged in users etc.

- **File Modification**

  System programs that are used for file modification basically change the data in the file or modify it in some other way. Text editors are a big example of file modification system programs.

- **Programming Language Support**

  These system programs provide additional support features for different programming languages. Some examples of these are compilers, debuggers etc. These compile a program and make sure it is error free respectively.

- **Loading and Execution**

  The system programs that deal with program loading and execution make sure that programs can be loaded into memory and executed correctly. Loaders and Linkers are a prime example of this type of system programs.

- **Communications**

  These system programs are needed for system communications such as web browsers. Web browsers allow systems to communicate and access information from the network as required.

- **Application Programs**

  Application programs can perform a wide range of services as per the needs of the users. These include programs for database systems, word processors, plotting tools, spreadsheets, games, scientific applications etc.

Explain components of system software ?

System software is used to create and to run application software. System software classified in two groups :

Program Development Environment :- Text Editor, Compiler, Assembler, Microprocessor

• **Text Editor** :- Text editor is used to edit programs. User writes application programs using text editor.

• **Compiler** :- Compiler translates programs written in a high level language to object code or machine code. C, C ++, Java are examples of high level language

- **Assembler** :- Assembler translates programs written in assembly language or law level language into object code or machine code. Assembler using input as .asm program
- **Microprocessor** :- Macro is a unit of specification for program generation through expansion. Microprocessor takes assembly or high level language program which contains macro definition

and macro call and converts it into an assembly or high level language program without macro.

Run Time Environment :- Linker , Loader, Interpreter, Operating System

- **Linker** :- Linker combines two or more separate object programs. For generation of machine code we required different files which are collected by linker.
- **Loader** :- Loader is responsible for taking machine code from secondary memory into primary memory. Thus loader is actually responsible for initiating the execution process . The loader is responsible for allocating space in main memory.
- **Interpreter** :- An interpreter is a language processor which bridges an execution gap without generating a machine language program. An interpreter code reads the statements of a program, analyzes them and then executes them on the virtual machine.
- **Operating system** :- An operating system is interface between users and the hardware of a computer system. An operating system is used to manage all resources of computer.