School of Computer Sciences and Engineering

Department of Computer Science and Engineering

B.E. – CSE

SEMESTER : 5th

SUBJECT NAME : OPERATING SYSTEM

SUBJECT CODE : 17YCT502

FACULTY : Dr. Rais Abdul Hamid Khan

Lecture Notes

Unit-1

Module-1

What is an Operating System?
Computer System = Hardware + Software
Software = Application Software + System Software(OS)

An Operating System is a system Software that acts as an intermediary/interface between a user of a computer and the computer hardware.

Operating system goals:

- Execute user programs and make solving user problems easier
- Make the computer system convenient to use
- Use the computer hardware in an efficient manner

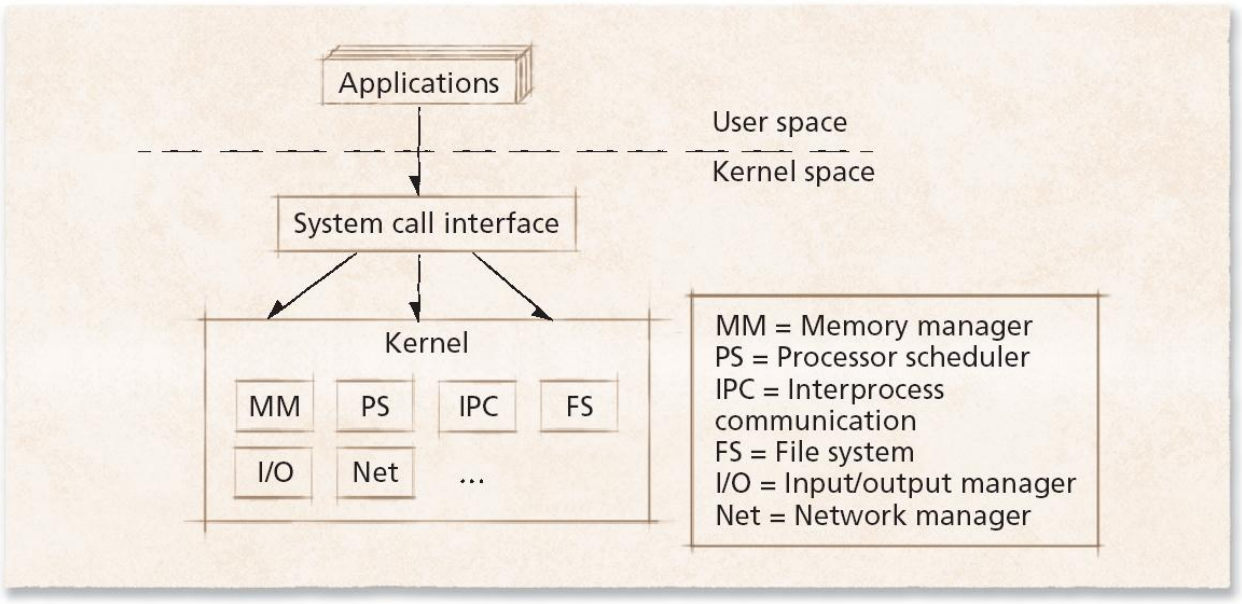
Architecture of Operating System:

Today's operating systems tend to be complex

- Provide many services
- Support variety of hardware and software
- Operating system architectures help manage this complexity
    - Organize operating system components
    - Specify privilege with which each component executes

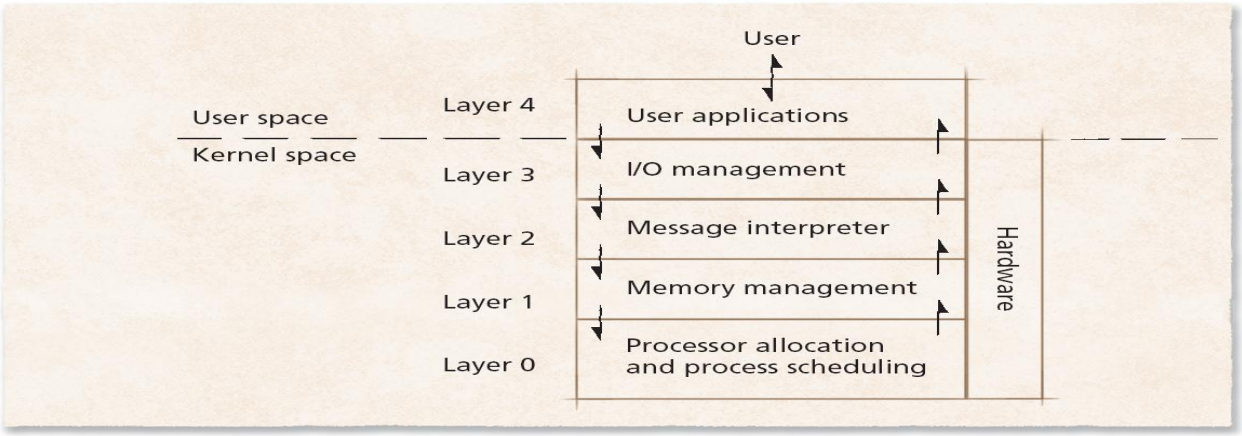
Monolithic Architecture

- Monolithic operating system
    - Every component contained in kernel
        - Any component can directly communicate with any other
    - Tend to be highly efficient
    - Disadvantage is difficulty determining source of subtle errors



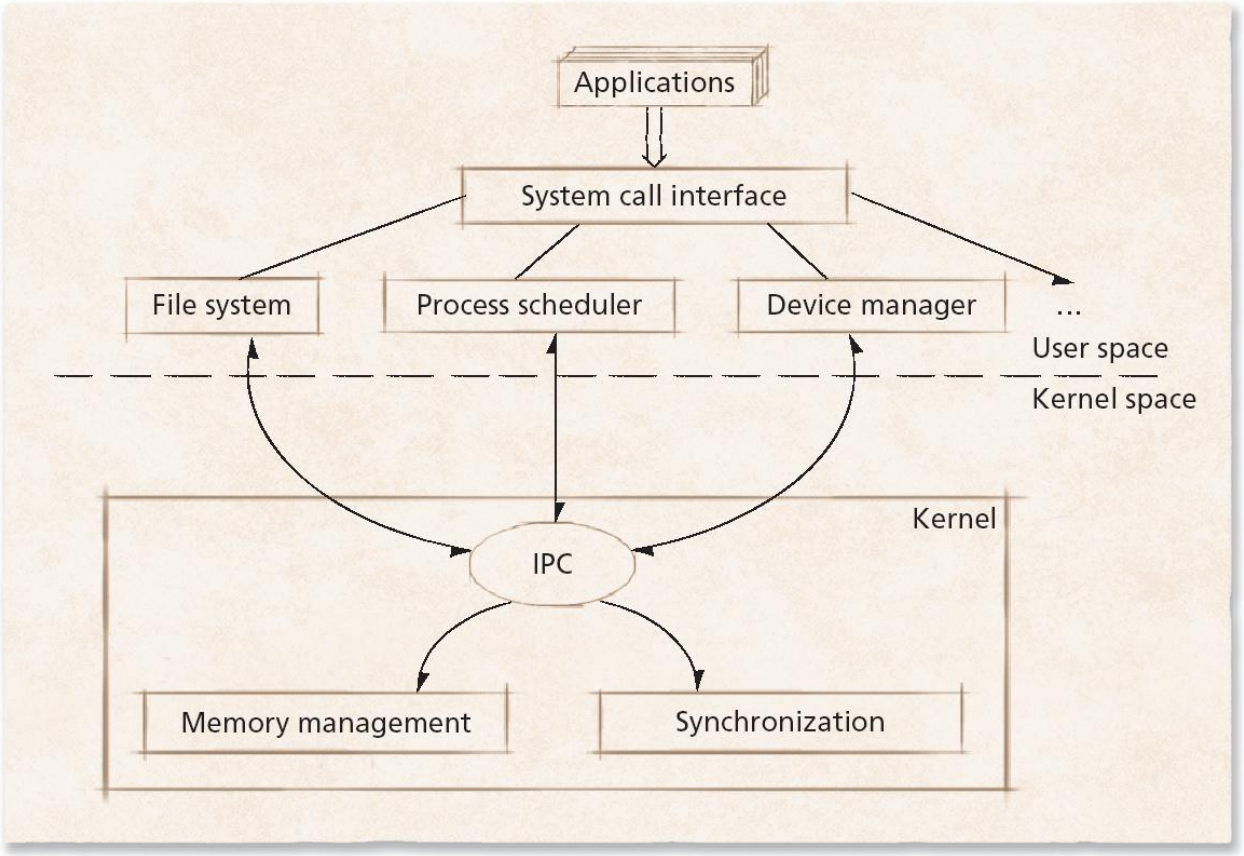
Layered Architecture

- Layered approach to operating systems
  - Tries to improve on monolithic kernel designs
    - Groups components that perform similar functions into layers
  - Each layer communicates only with layers immediately above and below it
  - Processes' requests might pass through many layers before completion
  - System throughput can be less than monolithic kernels
    - Additional methods must be invoked to pass data and control



Microkernel Architecture

- Microkernel operating system architecture
  - Provides only small number of services
    - Attempt to keep kernel small and scalable
  - High degree of modularity
    - Extensible, portable, and scalable
  - Increased level of intermodule communication
    - Can degrade system performance



Main Functions of Operating Systems:

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.

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

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

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

Error detecting aids –
The operating system constantly monitors the system to detect errors and avoid the malfunctioning of a computer system.

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

Evolution of Operating System

- **The First Generation (1940's to early 1950's)**
  - No Operating System
  - All programming was done in absolute machine language.

- **The Second Generation (1955-1965)**
  - First operating system was introduced in the early 1950's.It was called GMOS
  - Created by General Motors for IBM's machine the 701.
  - Single-stream batch processing systems
- **The Third Generation (1965-1980)**
  - Introduction of multiprogramming
  - Development of Minicomputer
- **The Fourth Generation (1980-Present Day)**
  - Development of PCs
  - Birth of Windows/MaC OS

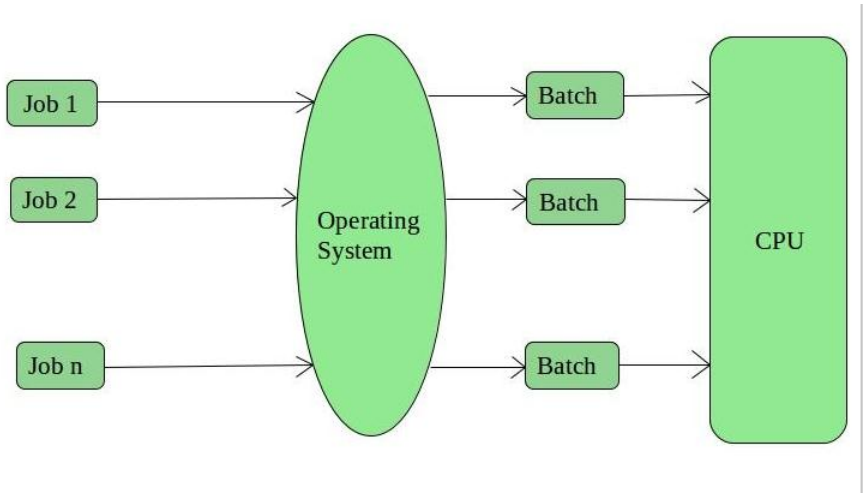
Unit-1

Module-2

Types of Operating Systems

1. Batch Operating System –

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



Advantages of Batch Operating System:

- It is very difficult to guess or know the time required for any job to complete. Processors of the batch systems know how long the job would be when it is in queue
- Multiple users can share the batch systems
- The idle time for the batch system is very less
- It is easy to manage large work repeatedly in batch systems

Disadvantages of Batch Operating System:

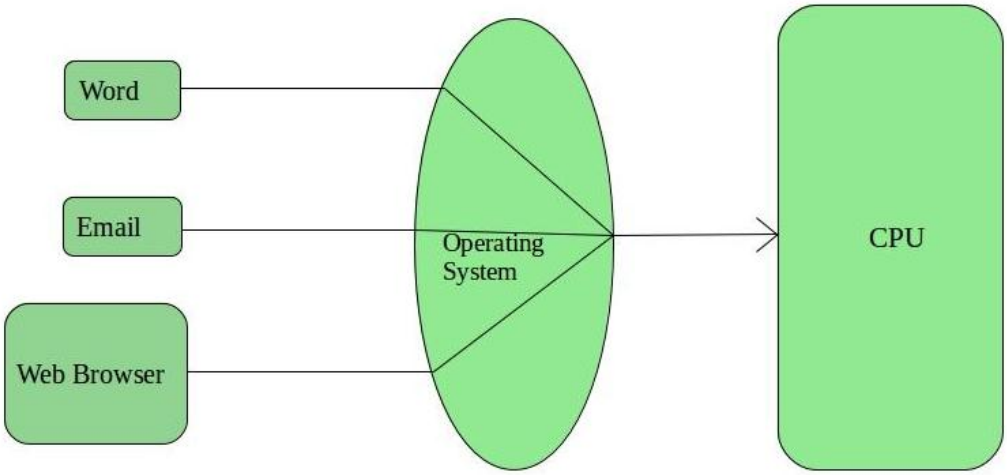
- The computer operators should be well known with batch systems
- Batch systems are hard to debug
- It is sometimes costly
- The other jobs will have to wait for an unknown time if any job fails

Examples of Batch based Operating System: Payroll System, Bank Statements, etc.

2. Time-Sharing Operating Systems –

Each task is given some time to execute so that all the tasks work smoothly. Each user gets the

time of CPU as they use a single system. These systems are also known as Multitasking Systems. The task can be from a single user or different users also. The time that each task gets to execute is called quantum. After this time interval is over OS switches over to the next task.



Advantages of Time-Sharing OS:

- Each task gets an equal opportunity
- Fewer chances of duplication of software
- CPU idle time can be reduced

Disadvantages of Time-Sharing OS:

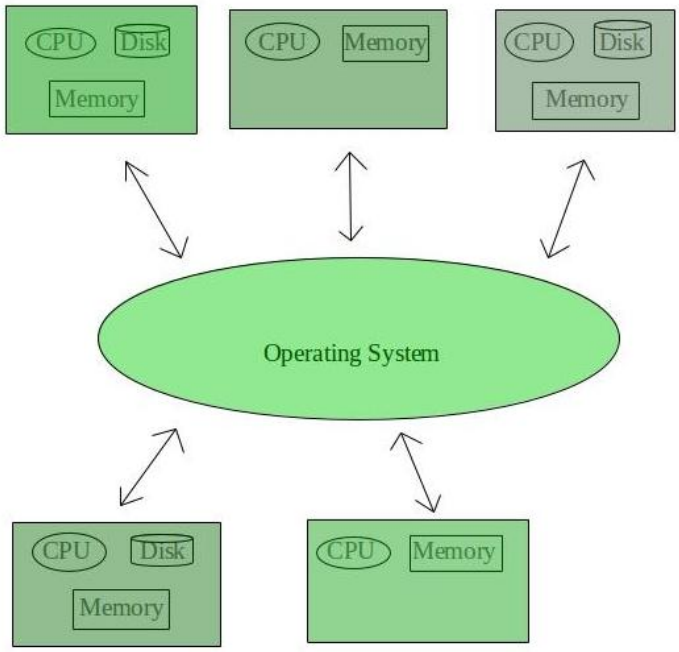
- Reliability problem
- One must have to take care of the security and integrity of user programs and data
- Data communication problem

Examples of Time-Sharing OSs are: Multics, Unix, etc. \

3. Distributed Operating System –

These types of the operating system is a recent advancement in the world of computer technology and are being widely accepted all over the world and, that too, with a great pace. Various autonomous interconnected computers communicate with each other using a shared communication network. Independent systems possess their own memory unit and CPU. These are referred to as **loosely coupled systems** or distributed systems. These system's processors differ in size and function. The major benefit of working with these types of the operating system is that it is always possible that one user can access the files or software which are not actually present on his system but some other system connected within this network i.e.,

remote access is enabled within the devices connected in that network.



Advantages of Distributed Operating System:

- Failure of one will not affect the other network communication, as all systems are independent from each other
- Electronic mail increases the data exchange speed
- Since resources are being shared, computation is highly fast and durable
- Load on host computer reduces
- These systems are easily scalable as many systems can be easily added to the network
- Delay in data processing reduces

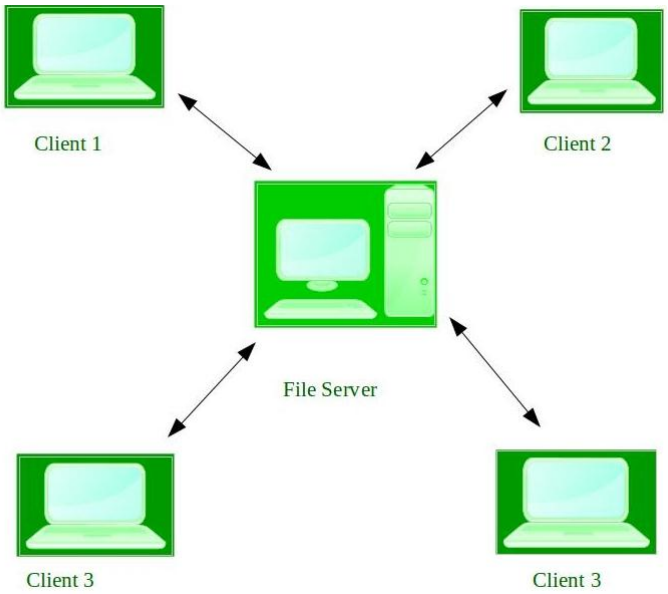
Disadvantages of Distributed Operating System:

- Failure of the main network will stop the entire communication
- To establish distributed systems the language which is used are not well defined yet
- These types of systems are not readily available as they are very expensive. Not only that the underlying software is highly complex and not understood well yet

Examples of Distributed Operating System are- LOCUS, etc.

4. Network Operating System –

These systems run on a server and provide the capability to manage data, users, groups, security, applications, and other networking functions. These types of operating systems allow shared access of files, printers, security, applications, and other networking functions over a small private network. One more important aspect of Network Operating Systems is that all the users are well aware of the underlying configuration, of all other users within the network, their individual connections, etc. and that's why these computers are popularly known as **tightly coupled systems**.



Advantages of Network Operating System:

- Highly stable centralized servers
- Security concerns are handled through servers
- New technologies and hardware up-gradation are easily integrated into the system
- Server access is possible remotely from different locations and types of systems

Disadvantages of Network Operating System:

- Servers are costly
- User has to depend on a central location for most operations
- Maintenance and updates are required regularly

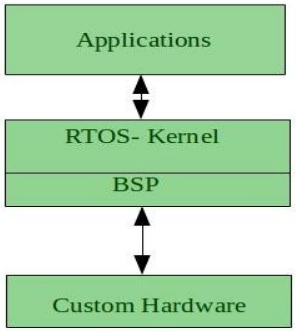
Examples of Network Operating System are: Microsoft Windows Server 2003, Microsoft Windows Server 2008, UNIX, Linux, Mac OS X, Novell NetWare, and BSD, etc.

5. Real-Time Operating System –
These types of OSs serve real-time systems. The time interval required to process and respond to inputs is very small. This time interval is called **response time**.

Real-time systems are used when there are time requirements that are very strict like missile systems, air traffic control systems, robots, etc.

Two types of Real-Time Operating System which are as follows:

- **Hard Real-Time Systems:**
  These OSs are meant for applications where time constraints are very strict and even the shortest possible delay is not acceptable. These systems are built for saving life like automatic parachutes or airbags which are required to be readily available in case of any accident. Virtual memory is rarely found in these systems.
- **Soft Real-Time Systems:**
  These OSs are for applications where for time-constraint is less strict.



Advantages of RTOS:

- **Maximum Consumption:** Maximum utilization of devices and system, thus more output from all the resources
- **Task Shifting:** The time assigned for shifting tasks in these systems are very less. For example, in older systems, it takes about 10 microseconds in shifting one task to another, and in the latest systems, it takes 3 microseconds.
- **Focus on Application:** Focus on running applications and less importance to applications which are in the queue.
- Real-time **operating system in** the **embedded system:** Since the size of programs are small, RTOS can also be used in embedded systems like in transport and others.
- **Error Free:** These types of systems are error-free.
- **Memory Allocation:** Memory allocation is best managed in these types of systems.

Disadvantages of RTOS:

- **Limited Tasks:** Very few tasks run at the same time and their concentration is very less on few applications to avoid errors.
- **Use heavy system resources:** Sometimes the system resources are not so good and they are expensive as well.
- **Complex Algorithms:** The algorithms are very complex and difficult for the designer to write on.
- **Device driver and interrupt signals:** It needs specific device drivers and interrupts signals to respond earliest to interrupts.
- **Thread Priority:** It is not good to set thread priority as these systems are very less prone to switching tasks.

Examples of Real-Time Operating Systems are: Scientific experiments, medical imaging systems, industrial control systems, weapon systems, robots, air traffic control systems, etc.

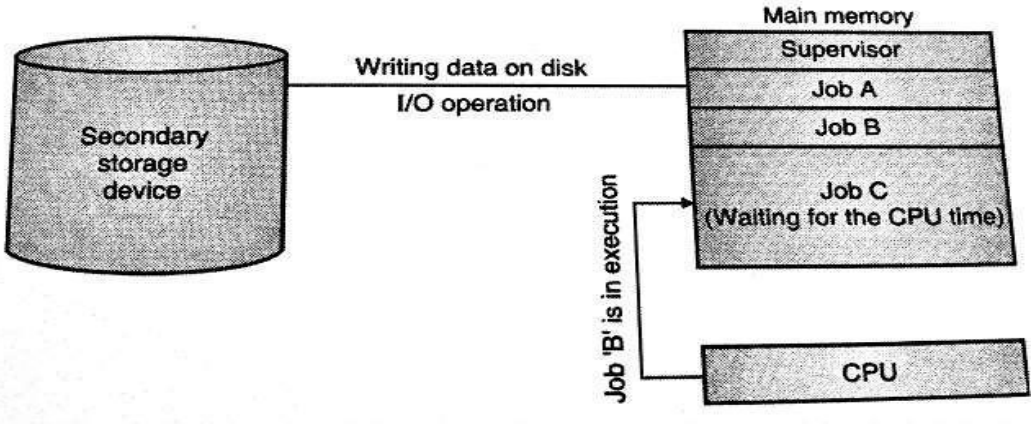
6. Embaded Operating System-

- An embedded operating system is one that is built into the circuitry of an electronic device.
- Embedded operating systems are now found in automobiles, bar-code scanners, cell phones, medical equipment, and personal digital assistants.
- The most popular embedded operating systems for consumer products, such as PDAs, include the following:
  - Windows XP Embedded
  - Windows CE .NET:- it supports wireless communications, multimedia and Web browsing. It also allows for the use of smaller versions of Microsoft Word, Excel, and Outlook.
  - Palm OS:- It is the standard operating system for Palm-brand PDAs as well as other proprietary handheld devices.
  - Symbian:- OS found in " smart" cell phones from Nokia and Sony Ericsson

7. Multiprogramming Operating System-

- This type of OS is used to execute more than one jobs simultaneously by a single processor.
- It increases CPU utilization by organizing jobs so that the CPU always has one job to execute.

- Multiprogramming operating systems use the mechanism of job scheduling and CPU scheduling.



Virtual Machine (VM)

A **virtual machine (VM)** is a virtual environment which functions as a virtual computer system with its own CPU, memory, network interface, and storage, created on a physical hardware system.

VMs are isolated from the rest of the system, and multiple VMs can exist on a single piece of hardware, like a server. That means, it as a simulated image of application software and operating system which is executed on a host computer or a server.

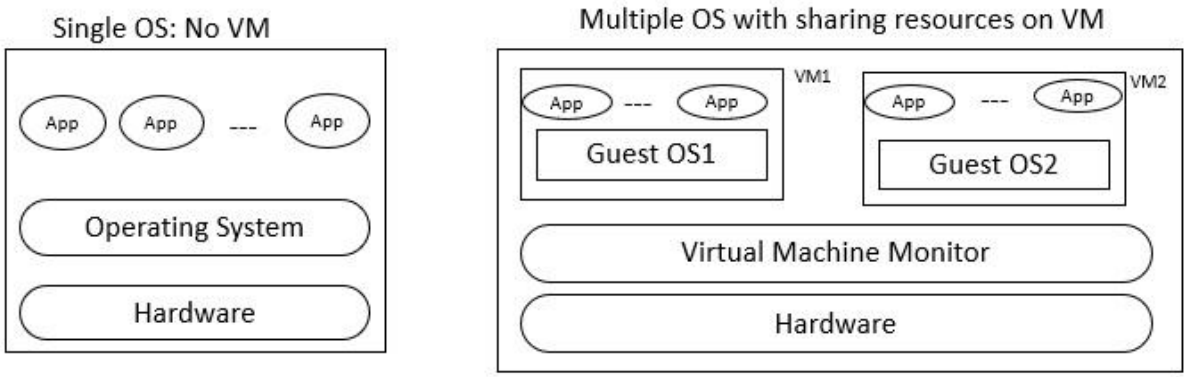
It has its own operating system and software that will facilitate the resources to virtual computers.

Characteristics of virtual machines

The characteristics of the virtual machines are as follows −

- Multiple OS systems use the same hardware and partition resources between virtual computers.
- Separate Security and configuration identity.
- Ability to move the virtual computers between the physical host computers as holistically integrated files.

The below diagram shows you the difference between the single OS with no VM and Multiple OS with VM −



Benefits

Let us see the major benefits of virtual machines for operating-system designers and users which are as follows −

- The multiple Operating system environments exist simultaneously on the same machine, which is isolated from each other.
- Virtual machine offers an instruction set architecture which differs from real computer.
- Using virtual machines, there is easy maintenance, application provisioning, availability and convenient recovery.

Virtual Machine encourages the users to go beyond the limitations of hardware to achieve their goals.

The operating system achieves virtualization with the help of a specialized software called a hypervisor, which emulates the PC client or server CPU, memory, hard disk, network and other hardware resources completely, enabling virtual machines to share resources.

The hypervisor can emulate multiple virtual hardware platforms that are isolated from each other allowing virtual machines to run Linux and window server operating machines on the same underlying physical host.

What is Shells?

A Shell provides you with an interface to the OS. It gathers input from you and executes programs based on that input. When a program finishes executing, it displays that program's output.

Shell is an environment in which we can run our commands, programs, and shell scripts. There are different flavors of a shell, just as there are different flavors of operating systems. Each flavor of shell has its own set of recognized commands and functions.

Shell Types

In Linux, there are two major types of shells −

- **Bourne shell** − If you are using a Bourne-type shell, the **$** character is the default prompt.
- **C shell** − If you are using a C-type shell, the % character is the default prompt.

Linux Shell Commands

The shell is the command interpreter on the Linux systems. Shell commands are instructions that instruct the system to do some action.

Some of the commonly used shell commands are −

basename

This command strips the directory and suffix from filenames. It prints the name of the file with all the leading directory components removed. It also removes a trailing suffix if it is specified.

Example of basename is as follows −

```
$ basename country/city.txt
```

This gets the name of the file i.e. city which is present in folder country.

```
city.txt
```

cat

This command concatenates and prints the contents of the file. If there is no file, then it reads the standard input.

Examples of cat are as follows −

Let us see how to print the contents of a file −

```
$ cat example.txt
```

The above example displays the contents of the file example.txt.

```
This is the content of the example text file
```

Let us see how to concatenate two files −

```
$ cat example1.txt example2.txt > example3.txt
$ cat example3.txt
```

In the above example, the contents of the text files example1 and example2 are concatenated into the text file example3. Then the contents of example3 file are displayed.

```
This is the example1 text file
This is the example2 text file
```

cal

This command is used to display a calendar. If a single parameter is specified, then the four-digit year is displayed. If there are two parameters, that denotes the month and the year. No parameter means that the current month is displayed.

Example of cal is as follows −

```
$ cal
```

Since there is no parameter specified with cal, it returns the calendar of the current month i.e. September.

```
September 2018
Mo Tu We Th Fr Sa Su
1 2
3 4 5 6 7 8 9
10 11 12 13 14 15 16
17 18 19 20 21 22 23
24 25 26 27 28 29 30
```

cd

cd is used to change the current working directory to the required folder. If the directory name is given, then the shell's name is changed to that directory. Otherwise, it changes to home.

Example of cd is as follows:

```
$ cd /user/local/example
$ pwd
```

The current directory is changed to example using cd and that is displayed using pwd command −

```
/user/local/example
```

cp

This command copies the contents of a file into another file.

Example of cp is as follows −

```
$ cat example.txt
This file is used as an example
$ cp example.txt example.bak
```

```
$cat example.bak
This file is used as an example
```

cp copies the contents of the text file example into the backup file example.bak

clear

The clear command clears the terminal screen. It ignores any command line parameters that may be present.

Example of clear is as follows −

```
$ clear
```

This clears the terminal screen.

cmp

This function compares two different files and reports the differences between them character by character. If the files differ, it tells the first byte and line number where they differ.

Example of cmp is as follows −

```
$ cat example1.txt
This is an example text file
$ cat example2.txt
This is also an example text file
$ cmp example1.txt example2.txt
example1.txt example2.txt differ: byte 10, line 1
```

This example demonstrates that the first difference in example1.txt and exampl2.txt is in line 1 and at byte 10.

mkdir

This command is used to create a directory in the Linux operating system.

Example of mkdir is as follows:

```
mkdir /fruit/apple
```

The above command creates a directory apple in the directory fruit.

rmdir

This command is used to remove a directory. All the files and subdirectories in a directory should be deleted first before deleting a directory.

Example of rmdir is as follows −

```
rmdir example
```

This deletes the directory example.

mv

The mv i.e. move command can be used for renaming directories.

Example of mv is as follows −

```
mv name1 name2
```

The initial name of the directory is name1 which is changed to name2.

What is shell scripting?

Shells are interactive, they accept commands as input from users and execute them. But if we want to execute a bunch of commands again and again (like a loop), we have to type in all commands each time in the terminal.

To avoid this, the shell can also take in the commands as the input from a file, where we can compile all of these commands and execute them. These files are called **Shell programs** or **shell scripts** and have the extension of **".sh"**.

Shell programs, just like any other programming language in the world, have a specific syntax and a set of rules. A shell script contains the following elements:

- Keywords: if, else, break continue, etc.
- Commands: cd, ls, echo, touch, cat, pwd, mkdir, etc.
- Loops: if…then…else, while…do, do…while, case, etc.
- Functions

Why use shell programming?

There are a gazillion reasons why you should use shell programming, here are a few:

1. Avoids repetitive work and automation
2. Helps in system monitoring
3. Used for routine backups
4. Adding new functionality to shell
5. Creating custom filters
7. Helps in automating various tasks
8. Changing time and date
9. Printing current working directory
10. Scheduling jobs

Advantages of shell programming

Again, if we start writing the advantages of the list advantages of shell programming, it may never end, nonetheless here are 8 advantages of it:

- Easy to use
- Portable
- Qiuck start
- Interactive debugging
- Same syntax as that of command line
- Avoids repetitive work and automation
- Shell scripting is much quicker than writing commands in the terminal
- Simple up learn

Disadvantages of shell programming

They say that every good thing has its negatives, which also hold true for shell programming, here are some of its disadvantages:

- Slow execution speed
- Not suited for large and complex tasks
- Design flaws within the language syntax
- Provides minimal data structure, unlike other scripting languages
- Prone to costly errors

Unit-2

Module-1

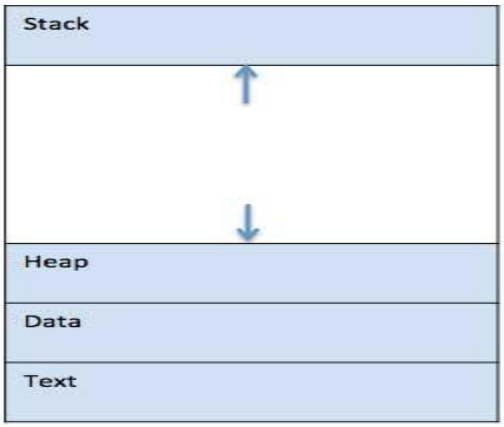
Process:

A process is basically a program in execution. The execution of a process must progress in a sequential fashion.

A process is defined as an entity which represents the basic unit of work to be implemented in the system.

To put it in simple terms, we write our computer programs in a text file and when we execute this program, it becomes a process which performs all the tasks mentioned in the program.

When a program is loaded into the memory and it becomes a process, it can be divided into four sections ─ stack, heap, text and data. The following image shows a simplified layout of a process inside main memory −



S.N.	Component & Description
1	**Stack** The process Stack contains the temporary data such as method/function parameters, return address and local variables.
2	**Heap** This is dynamically allocated memory to a process during its run time.

3	**Text** This includes the current activity represented by the value of Program Counter and the contents of the processor's registers.
4	**Data** This section contains the global and static variables.

Process Control Block:

Process Control Block is a data structure that contains information of the process related to it. The process control block is also known as a task control block, entry of the process table, etc.

It is very important for process management as the data structuring for processes is done in terms of the PCB. It also defines the current state of the operating system.

Structure of the Process Control Block

The process control stores many data items that are needed for efficient process management. Some of these data items are explained with the help of the given diagram −

Process Control Block (PCB)

The following are the data items −

Process State

This specifies the process state i.e. new, ready, running, waiting or terminated.

Process Number

This shows the number of the particular process.

Program Counter

This contains the address of the next instruction that needs to be executed in the process.

Registers

This specifies the registers that are used by the process. They may include accumulators, index registers, stack pointers, general purpose registers etc.

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.

There are three types of process scheduler

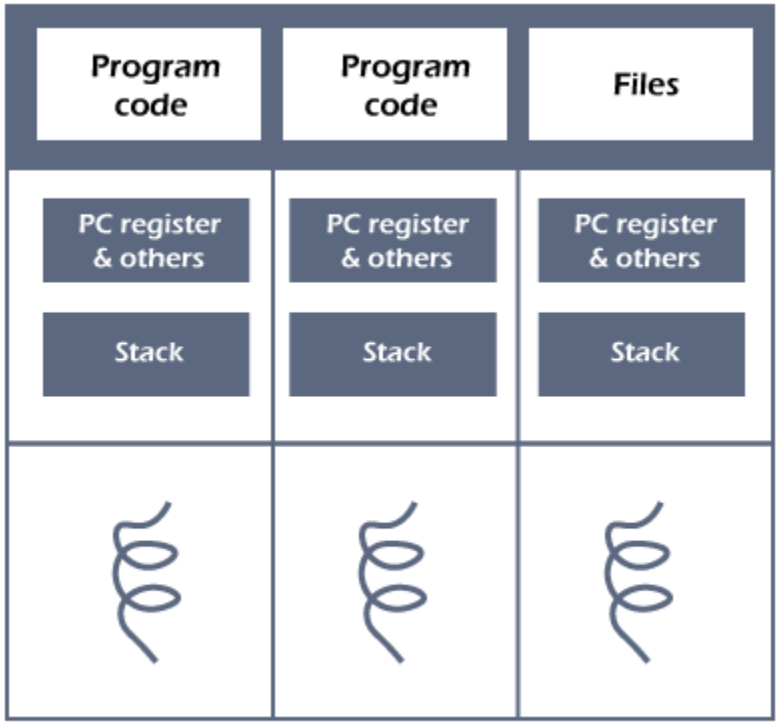
1. **Long Term or job scheduler:**

It brings the new process to the 'Ready State'. It controls **Degree of Multi-programming**, i.e., number of process present in ready state at any point of time. It is important that the long-term scheduler make a careful selection of both IO and CPU bound process. IO bound tasks are which use much of their time in input and output operations while CPU bound processes are which spend their time on CPU. The job scheduler increases efficiency by maintaining a balance between the two.

2. **Short term or CPU scheduler:**
   It is responsible for selecting one process from ready state for scheduling it on the running state. Note: Short-term scheduler only selects the process to schedule it doesn't load the process on running. Here is when all the scheduling algorithms are used. The CPU scheduler is responsible for ensuring there is no starvation owing to high burst time processes.
   Dispatcher is responsible for loading the process selected by Short-term scheduler on the CPU (Ready to Running State) Context switching is done by dispatcher only. A dispatcher does the following:
   1. Switching context.
   2. Switching to user mode.
   3. Jumping to the proper location in the newly loaded program.
3. **Medium-term scheduler:**
   It is responsible for suspending and resuming the process. It mainly does swapping (moving processes from main memory to disk and vice versa). Swapping may be necessary to improve the process mix or because a change in memory requirements has overcommitted available memory, requiring memory to be freed up. It is helpful in maintaining a perfect balance between the I/O bound and the CPU bound. It reduces the degree of multiprogramming.

Threads in Operating System

A thread is a single sequential flow of execution of tasks of a process so it is also known as thread of execution or thread of control. There is a way of thread execution inside the process of any operating system. Apart from this, there can be more than one thread inside a process. Each thread of the same process makes use of a separate program counter and a stack of activation records and control blocks. Thread is often referred to as a lightweight process.



Three threads of same process

The process can be split down into so many threads. **For example**, in a browser, many tabs can be viewed as threads. MS Word uses many threads - formatting text from one thread, processing input from another thread, etc.

Need of Thread:

- It takes far less time to create a new thread in an existing process than to create a new process.
- Threads can share the common data, they do not need to use Inter- Process communication.
- Context switching is faster when working with threads.
- It takes less time to terminate a thread than a process.

Types of Threads

In the operating system, there are two types of threads.

1. Kernel level thread.
2. User-level thread.

User level thread

The kernel is unaware of the existence of user-level threads because they are implemented by users. It treats them as if they were single-threaded applications. Threads at the user level are substantially smaller and faster than threads at the kernel level. A program counter (PC), stack, registers, and a short process control block are used to represent them. In addition, there is no kernel involvement in user-level thread synchronization.

Kernel level thread

Kernel-level threads are handled directly by the operating system, with the kernel handling thread management. The kernel is in charge of the process's context information as well as the process threads. Kernel-level threads are hence slower than user-level threads.

Uniprocessor

A uniprocessor system is defined as a computer system that has a single central processing unit that is used to execute computer tasks. A uniprocessor system consists of a single processor which executes the program instructions as a single instruction per cycle.

Unit-2

Module-2

CPU Scheduling:

Preemptive Scheduling is a CPU scheduling technique that works by dividing time slots of CPU to a given process. The time slot given might be able to complete the whole process or might not be able to it. When the burst time of the process is greater than CPU cycle, it is placed back into the ready queue and will execute in the next chance. This scheduling is used when the process switch to ready state.

Algorithms that are backed by preemptive Scheduling are round-robin (RR), priority, SRTF (shortest remaining time first).

Non-preemptive Scheduling is a CPU scheduling technique the process takes the resource (CPU time) and holds it till the process gets terminated or is pushed to the waiting state. No process is interrupted until it is completed, and after that processor switches to another process.

Algorithms that are based on non-preemptive Scheduling are non-preemptive priority, and shortest Job first.

CPU Scheduling Criteria

The criteria include the following:

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

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

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

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

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

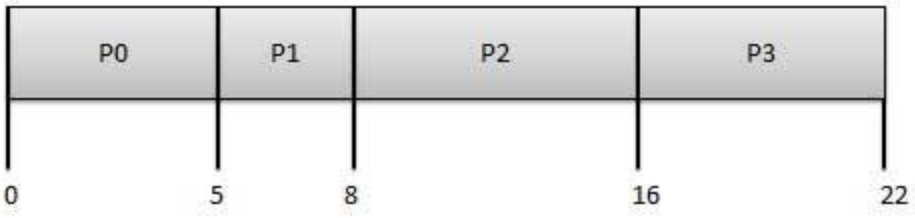
Scheduling Algorithms:

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) or Shortest Job First (SJF)

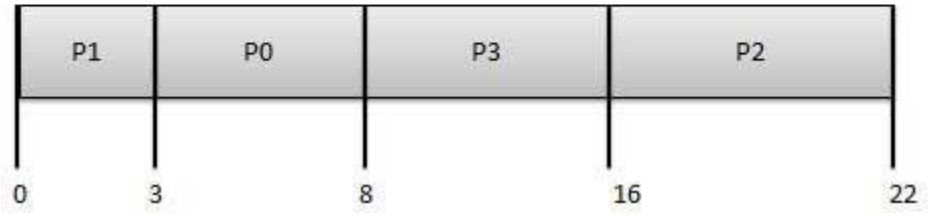
- This is also known as **shortest job first**, or SJF
- This is a non-preemptive, pre-emptive scheduling algorithm.
- Best approach to minimize waiting time.
- Easy to implement in Batch systems where required CPU time is known in advance.

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

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

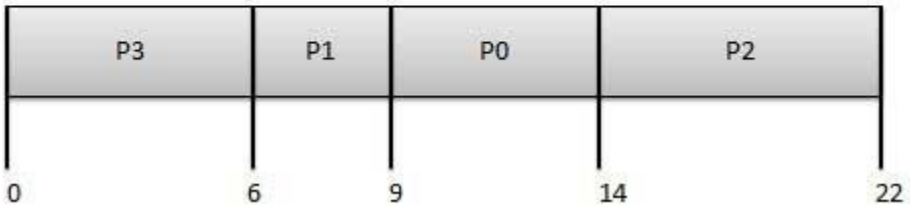
Priority Based Scheduling

- 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

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



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

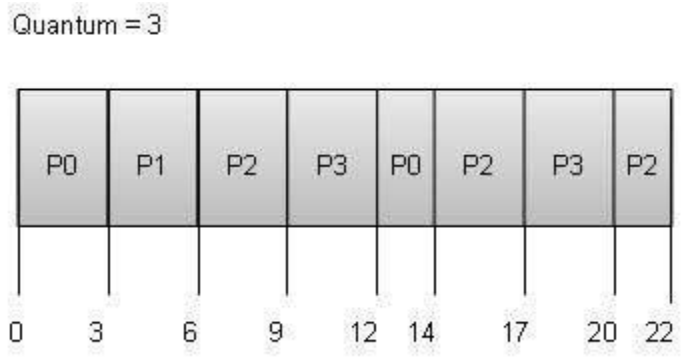
Average Wait Time: (0 + 10 + 12 + 2)/4 = 24 / 4 = 6

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

High Response Ratio Next Scheduling:

In this scheduling algorithm, scheduling is done on the basis of response ratio. The process with the highest response ratio is scheduled next which reduces starvation in the system

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.

Multilevel Feedback Queue Scheduling:

This scheduling algorithm is similar to multilevel queue scheduling except that the processes here can change their queue too i.e., if a process is in queue1, then after partial execution, it can switch to queue2.

Thread Scheduling:

Threads are scheduled for execution based on their priority. Even though threads are executing within the runtime, all threads are assigned processor time slices by the operating system. The details of the scheduling algorithm used to determine the order in which threads are executed varies with each operating system.

Multiple Processor Scheduling:

Multiple processor scheduling or multiprocessor scheduling focuses on designing the system's scheduling function, which consists of more than one processor. Multiple CPUs share the load (load sharing) in multiprocessor scheduling so that various processes run simultaneously.

In general, multiprocessor scheduling is complex as compared to single processor scheduling. In the multiprocessor scheduling, there are many processors, and they are identical, and we can run any process at any time.

Approaches to Multiple Processor Scheduling

There are two approaches to multiple processor scheduling in the operating system: Symmetric Multiprocessing and Asymmetric Multiprocessing.

1. **Symmetric Multiprocessing:** It is used where each processor is ***self-scheduling***. All processes may be in a common ready queue, or each processor may have its private queue for ready processes. The scheduling proceeds further by having the scheduler for each processor examine the ready queue and select a process to execute.
2. **Asymmetric Multiprocessing:** It is used when all the scheduling decisions and I/O processing are handled by a single processor called the ***Master Server***. The other processors execute only the ***user code***. This is simple and reduces the need for data sharing, and this entire scenario is called Asymmetric Multiprocessing.

Real-time scheduling algorithms:

In a simple real-time system, there might be no need for a scheduler, one task can simply call the next.

But for more complex real-time systems, that have a large but fixed number of tasks that do not function in pipeline function, one may need **static scheduling.**
And for real-time systems where the workload keeps changing, one may need **dynamic scheduling.**

In dynamic scheduling two important question arises:

1. How to handle the overload?

To handle the overload these processes can be used:

- Best effort
- Work shedding

1. How to choose which process to execute first?

To decide which process to execute first from the ready queue these methods may be used:

- SJF
- Static priority
- Slack time.

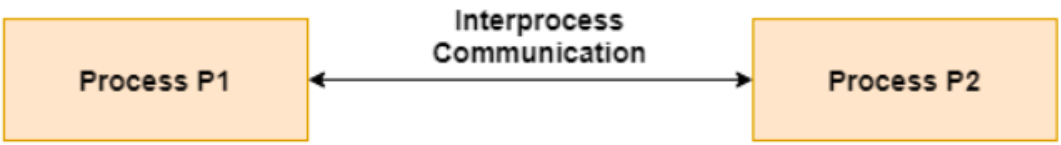
Unit-3

Module-1

Interprocess Communication:

Inter process communication is the mechanism provided by the operating system that allows processes to communicate with each other. This communication could involve a process letting another process know that some event has occurred or the transferring of data from one process to another.

A diagram that illustrates interprocess communication is as follows −



Process Synchronization:

When two or more process cooperates with each other, their order of execution must be preserved otherwise there can be conflicts in their execution and inappropriate outputs can be produced.

A **cooperative process** is the one which can affect the execution of other process or can be affected by the execution of other process. Such processes need to be synchronized so that their order of execution can be guaranteed.

The procedure involved in preserving the appropriate order of execution of cooperative processes is known as **Process Synchronization**. There are various synchronization mechanisms that are used to synchronize the processes.

Race Condition

A Race Condition typically occurs when two or more threads try to read, write and possibly make the decisions based on the memory that they are accessing concurrently.

Critical Section

The regions of a program that try to access shared resources and may cause race conditions are called critical section. To avoid race condition among the processes, we need to assure that only one process at a time can execute within the critical section.

What is Concurrency?

It refers to the execution of multiple instruction sequences at the same time. It occurs in an operating system when multiple process threads are executing concurrently. These threads can interact with one another via shared memory or message passing. Concurrency results in resource sharing, which causes issues like deadlocks and resource scarcity. It aids with techniques such as process coordination, memory allocation, and execution schedule to maximize throughput.

Principles of Concurrency

Today's technology, like multi-core processors and parallel processing, allows multiple processes and threads to be executed simultaneously. Multiple processes and threads can access the same memory space, the same declared variable in code, or even read or write to the same file.

The amount of time it takes a process to execute cannot be simply estimated, and you cannot predict which process will complete first, enabling you to build techniques to deal with the problems that concurrency creates.

Interleaved and overlapping processes are two types of concurrent processes with the same problems. It is impossible to predict the relative speed of execution, and the following factors determine it:

1. The way operating system handles interrupts
2. Other processes' activities
3. The operating system's scheduling policies

Problems in Concurrency

There are various problems in concurrency. Some of them are as follows:

1. Locating the programming errors

It's difficult to spot a programming error because reports are usually repeatable due to the varying states of shared components each time the code is executed.

2. Sharing Global Resources

Sharing global resources is difficult. If two processes utilize a global variable and both alter the variable's value, the order in which the many changes are executed is critical.

3. Locking the channel

It could be inefficient for the OS to lock the resource and prevent other processes from using it.

4. Optimal Allocation of Resources

It is challenging for the OS to handle resource allocation properly.

Issues of Concurrency

Various issues of concurrency are as follows:

1. Non-atomic

A non-atomic operation depends on other processes, and an atomic operation runs independently of other processes.

2. Deadlock

In concurrent computing, it occurs when one group member waits for another member, including itself, to send a message and release a lock. Software and hardware locks are commonly used to arbitrate shared resources and implement process synchronization in parallel computing, distributed systems, and multiprocessing.

3. Blocking

A blocked process is waiting for some event, like the availability of a resource or completing an I/O operation. Processes may block waiting for resources, and a process may be blocked for a long time waiting for terminal input. If the process is needed to update some data periodically, it will be very undesirable.

4. Race Conditions

A race problem occurs when the output of a software application is determined by the timing or sequencing of other uncontrollable events. Race situations can also happen in multithreaded software, runs in a distributed environment, or is interdependent on shared resources.

5. Starvation

A problem in concurrent computing is where a process is continuously denied the resources it needs to complete its work. It could be caused by errors in scheduling or mutual exclusion algorithm, but resource leaks may also cause it.

Concurrent system design frequently requires developing dependable strategies for coordinating their execution, data interchange, memory allocation, and execution schedule to decrease response time and maximize throughput.

Advantages and Disadvantages of Concurrency in Operating System

Various advantages and disadvantages of Concurrency in Operating systems are as follows:

Advantages

1. Better Performance

It improves the operating system's performance. When one application only utilizes the processor, and another only uses the disk drive, the time it takes to perform both apps simultaneously is less than the time it takes to run them sequentially.

2. Better Resource Utilization

It enables resources that are not being used by one application to be used by another.

3. Running Multiple Applications

It enables you to execute multiple applications simultaneously.

Disadvantages

1. It is necessary to protect multiple applications from each other.
2. It is necessary to use extra techniques to coordinate several applications.
3. Additional performance overheads and complexities in OS are needed for switching between applications.

Mutual Exclusion:

Mutual exclusion is a property of process synchronization that states that **"no two processes can exist in the critical section at any given point of time"**. The term was first coined by Dijkstra. Any process synchronization technique being used must satisfy the property of mutual exclusion, without which it would not be possible to get rid of a race condition.

The need for mutual exclusion comes with concurrency.

Mutex vs Semaphore

Mutex and semaphore both provide synchronization services, but they are not the same.

Mutex

Mutex is a mutual exclusion object that synchronizes access to a resource. It is created with a unique name at the start of a program. The mutex locking mechanism ensures only one thread can acquire the mutex and enter the critical section. This thread only releases the mutex when it exits in the critical section.

Advantages of Mutex

Here are the following advantages of the mutex, such as:

- Mutex is just simple locks obtained before entering its critical section and then releasing it.
- Since only one thread is in its critical section at any given time, there are no race conditions, and data always remain consistent.

Disadvantages of Mutex

Mutex also has some disadvantages, such as:

- If a thread obtains a lock and goes to sleep or is preempted, then the other thread may not move forward. This may lead to starvation.
- It can't be locked or unlocked from a different context than the one that acquired it.
- Only one thread should be allowed in the critical section at a time.
- The normal implementation may lead to a busy waiting state, which wastes CPU time.

Semaphores:

Semaphores are integer variables that are used to solve the critical section problem by using two atomic operations, wait and signal that are used for process synchronization.

The definitions of wait and signal are as follows −

- **Wait**

  The wait operation decrements the value of its argument S, if it is positive. If S is negative or zero, then no operation is performed.

```
wait(S)
{
   while (S<=0);

   S--;
}
```

- **Signal**

  The signal operation increments the value of its argument S.

```
signal(S)
{
   S++;
}
```

Types of Semaphores

There are two main types of semaphores i.e. counting semaphores and binary semaphores. Details about these are given as follows −

- **Counting Semaphores**

  These are integer value semaphores and have an unrestricted value domain. These semaphores are used to coordinate the resource access, where the semaphore count is the number of available resources. If the resources are added, semaphore count automatically incremented and if the resources are removed, the count is decremented.

- **Binary Semaphores**

  The binary semaphores are like counting semaphores but their value is restricted to 0 and 1. The wait operation only works when the semaphore is 1 and the signal operation succeeds when semaphore is 0. It is sometimes easier to implement binary semaphores than counting semaphores.

Advantages of Semaphores

Some of the advantages of semaphores are as follows −

- Semaphores allow only one process into the critical section. They follow the mutual exclusion principle strictly and are much more efficient than some other methods of synchronization.

- There is no resource wastage because of busy waiting in semaphores as processor time is not wasted unnecessarily to check if a condition is fulfilled to allow a process to access the critical section.
- Semaphores are implemented in the machine independent code of the microkernel. So they are machine independent.

Disadvantages of Semaphores

Some of the disadvantages of semaphores are as follows −

- Semaphores are complicated so the wait and signal operations must be implemented in the correct order to prevent deadlocks.
- Semaphores are impractical for last scale use as their use leads to loss of modularity. This happens because the wait and signal operations prevent the creation of a structured layout for the system.
- Semaphores may lead to a priority inversion where low priority processes may access the critical section first and high priority processes later.

What is Shared Memory?

The fundamental model of inter-process communication is the shared memory system. In a shared memory system, the collaborating communicates with each other by establishing the shared memory region in the address space region.

If the process wishes to initiate communication and has data to share, create a shared memory region in its address space. After that, if another process wishes to communicate and tries to read the shared data, it must attach to the starting process's shared address space.

What is Message Passing?

In this message passing process model, the processes communicate with others by exchanging messages. A communication link between the processes is required for this purpose, and it must provide at least two operations: ***transmit (message)*** and ***receive (message)***. Message sizes might be flexible or fixed.

Key differences between the Shared Memory and Message Passing

The main differences between the Shared Memory and the Message Passing are as follows:

Shared Memory	Message Passing
It is mainly used for data communication.	It is mainly used for communication.
It offers a maximum speed of computation because communication is completed via the shared memory, so the system calls are only required to establish the shared memory.	It takes a huge time because it is performed via the kernel (system calls).
The code for reading and writing the data from the shared memory should be written explicitly by the developer.	No such code is required in this case because the message passing feature offers a method for communication and synchronization of activities executed by the communicating processes.
It is used to communicate between the single processor and multiprocessor systems in which the processes to be communicated are on the same machine and share the same address space.	It is most commonly utilized in a distributed setting when communicating processes are spread over multiple devices linked by a network.
It is a faster communication strategy than the message passing.	It is a relatively slower communication strategy than the shared memory.
Make sure that processes in shared memory aren't writing to the same address simultaneously.	It is useful for sharing little quantities of data without causing disputes.

Classical Problems of Synchronization

Below are some of the classical problems depicting flaws of process synchronaization in systems where cooperating processes are present.

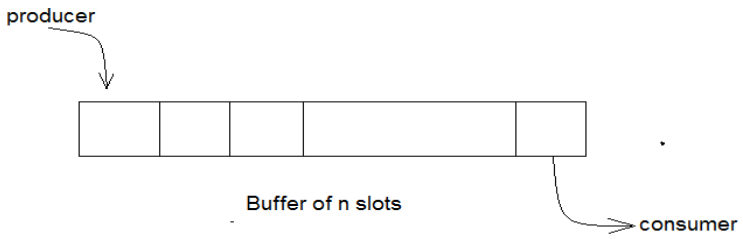
We will discuss the following three problems:

1. Bounded Buffer (Producer-Consumer) Problem
2. Dining Philosophers Problem
3. The Readers Writers Problem

Bounded Buffer Problem

Bounded buffer problem, which is also called **producer consumer problem**, is one of the classic problems of synchronization.

There is a buffer of n slots and each slot is capable of storing one unit of data. There are two processes running, namely, producer and consumer, which are operating on the buffer.



Bounded Buffer Problem

A producer tries to insert data into an empty slot of the buffer. A consumer tries to remove data from a filled slot in the buffer. As you might have guessed by now, those two processes will not produce the expected output if they are being executed concurrently.

There needs to be a way to make the producer and consumer work in an independent manner.

Solution

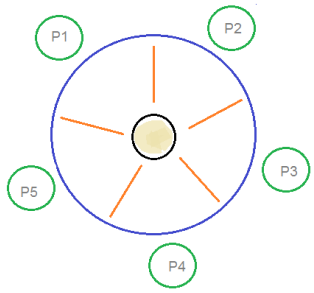
One solution of this problem is to use semaphores. The semaphores which will be used here are:

- a **binary semaphore** which is used to acquire and release the lock.
- a **counting semaphore** whose initial value is the number of slots in the buffer, initially all slots are empty.

At any instant, the current value of empty represents the number of empty slots in the buffer and full represents the number of occupied slots in the buffer.

Dining Philosophers Problem

- The dining philosopher's problem involves the allocation of limited resources to a group of processes in a deadlock-free and starvation-free manner.
- There are five philosophers sitting around a table, in which there are five chopsticks/forks kept beside them and a bowl of rice in the centre, when a philosopher wants to eat, he uses two chopsticks - one from their left and one from their right. When a philosopher wants to think, he keeps down both chopsticks at their original place.



Dining Philosophers Problem

Solution

From the problem statement, it is clear that a philosopher can think for an indefinite amount of time. But when a philosopher starts eating, he has to stop at some point of time. The philosopher is in an endless cycle of thinking and eating.

When a philosopher wants to eat the rice, he will wait for the chopstick at his left and picks up that chopstick. Then he waits for the right chopstick to be available, and then picks it too. After eating, he puts both the chopsticks down.

But if all five philosophers are hungry simultaneously, and each of them pickup one chopstick, then a deadlock situation occurs because they will be waiting for another chopstick forever. The possible solutions for this are:

- A philosopher must be allowed to pick up the chopsticks only if both the left and right chopsticks are available.
- Allow only four philosophers to sit at the table. That way, if all the four philosophers pick up four chopsticks, there will be one chopstick left on the table. So, one philosopher can start eating and eventually, two chopsticks will be available. In this way, deadlocks can be avoided.

The Readers Writers Problem

There are two types of processes in this context. They are **reader** and **writer**. Any number of **readers** can read from the shared resource simultaneously, but only one **writer** can write to the shared resource. When a **writer** is writing data to the resource, no other process can access the resource. A **writer** cannot write to the resource if there are non zero number of readers accessing the resource at that time.

Solution

From the above problem statement, it is evident that readers have higher priority than writer.

If a writer wants to write to the resource, it must wait until there are no readers currently accessing that resource.

Unit-3

Module-2

Deadlock

- A deadlock consists of a set of blocked processes, each holding a resource and waiting to acquire a resource held by another process in the set
- Example
  - A system has 2 disk drives
  - P_1 and P_2 each hold one disk drive and each needs the other one

Deadlock Characterization

Deadlock can arise if four conditions hold simultaneously.

- **Mutual exclusion:** only one process at a time can use a resource
- **Hold and wait:** a process holding at least one resource is waiting to acquire additional resources held by other processes
- **No pre emption:** a resource can be released only voluntarily by the process holding it after that process has completed its task
- **Circular wait:** there exists a set $\{P_0, P_1, \ldots, P_0\}$ of waiting processes such that P_0 is waiting for a resource that is held by P_1, P_1 is waiting for a resource that is held by P_2, …, P_{n-1} is waiting for a resource that is held by P_n, and P_n is waiting for a resource that is held by P_0

Deadlock Prevention

- Deadlocks can be prevented by preventing at least one of the four required conditions:

Mutual Exclusion

- Shared resources such as read-only files do not lead to deadlocks.
- Unfortunately, some resources, such as printers and tape drives, require exclusive access by a single process.

Hold and Wait

- To prevent this condition processes must be prevented from holding one or more resources while simultaneously waiting for one or more others. There are several possibilities for this:
    - Require that all processes request all resources at one time. This can be wasteful of system resources if a process needs one resource early in its execution and doesn't need some other resource until much later.
    - Require that processes holding resources must release them before requesting new resources, and then re-acquire the released resources along with the new ones in a single new request. This can be a problem if a process has partially completed an operation using a resource and then fails to get it re-allocated after releasing it.
    - Either of the methods described above can lead to starvation if a process requires one or more popular resources.

No Preemption

- Preemption of process resource allocations can prevent this condition of deadlocks, when it is possible.
    - One approach is that if a process is forced to wait when requesting a new resource, then all other resources previously held by this process are implicitly released, (preempted), forcing this process to re-acquire the old resources along with the new resources in a single request, similar to the previous discussion.
    - Another approach is that when a resource is requested and not available, then the system looks to see what other processes currently have those resources *and* are themselves blocked waiting for some other resource. If such a process is found, then some of their resources may get preempted and added to the list of resources for which the process is waiting.
    - Either of these approaches may be applicable for resources whose states are easily saved and restored, such as registers and memory, but are generally not applicable to other devices such as printers and tape drives.

Circular Wait

- One way to avoid circular wait is to number all resources, and to require that processes request resources only in strictly increasing (or decreasing) order.
- In other words, in order to request resource Rj, a process must first release all Ri such that i >= j.
- One big challenge in this scheme is determining the relative ordering of the different resources

Deadlock Avoidance

- The general idea behind deadlock avoidance is to prevent deadlocks from ever happening, by preventing at least one of the mentioned conditions.
- This requires more information about each process, AND tends to lead to low device utilization. (i.e. it is a conservative approach.)
- When a scheduler sees that starting a process or granting resource requests may lead to future deadlocks, then that process is just not started or the request is not granted.
- A resource allocation **state** is defined by the number of available and allocated resources, and the maximum requirements of all processes in the system.

Deadlock Detection

- If a system does not employ either a deadlock-prevention or a deadlock avoidance algorithm, then a deadlock situation may occur. In this environment, the system must provide:
  - An algorithm that examines the state of the system to determine whether a deadlock has occurred
  - An algorithm to recover from the deadlock

- A detection-and-recovery scheme requires various kinds of overhead
  - Run-time costs of maintaining necessary information and executing the detection algorithm
  - Potential losses inherent in recovering from a deadlock

Recovery from Deadlock:

- **Two Approaches**
  - Process Termination
  - Resource Pre emption

Process Termination

Abort all deadlocked processes

- This approach will break the deadlock, but at great expense

  Abort one process at a time until the deadlock cycle is eliminated
- This approach incurs considerable overhead, since, after each process is aborted, a deadlock-detection algorithm must be re-invoked to determine whether any

processes are still deadlocked

Many factors may affect which process is chosen for termination

- What is the priority of the process?
- How long has the process run so far and how much longer will the process need to run before completing its task?
- How many more resources does the process need in order to finish its task?
- How many processes will need to be terminated?

Resource Pre emption

- With this approach, we successively pre empt some resources from processes and give these resources to other processes until the deadlock cycle is broken
- When pre emption is required to deal with deadlocks, then three issues need to be addressed:
  - **Selecting a victim** – Which resources and which processes are to be pre-empted?
  - **Rollback** – If we pre empt a resource from a process, what should be done with that process?
  - **Starvation** – How do we ensure that starvation will not occur? That is, how can we guarantee that resources will not always be pre-empted from the same 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.

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 or in unsafe state at every step which the operating system performs. The process 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.

Unit-4

Module-1

What do you mean by memory management?

Memory is the important part of the computer that is used to store the data. Its management is critical to the computer system because the amount of main memory available in a computer system is very limited. At any time, many processes are competing for it. Moreover, to increase performance, several processes are executed simultaneously. For this, we must keep several processes in the main memory, so it is even more important to manage them effectively.

Role of Memory management

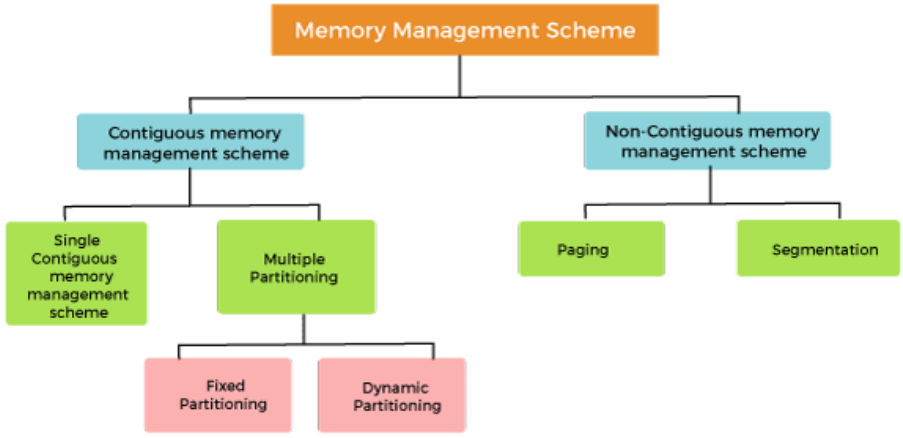
Following are the important roles of memory management in a computer system:

- Memory manager is used to keep track of the status of memory locations, whether it is free or allocated. It addresses primary memory by providing abstractions so that software perceives a large memory is allocated to it.
- Memory manager permits computers with a small amount of main memory to execute programs larger than the size or amount of available memory. It does this by moving information back and forth between primary memory and secondary memory by using the concept of swapping.
- The memory manager is responsible for protecting the memory allocated to each process from being corrupted by another process. If this is not ensured, then the system may exhibit unpredictable behavior.
- Memory managers should enable sharing of memory space between processes. Thus, two programs can reside at the same memory location although at different times.

Memory Management Techniques:

The memory management techniques can be classified into following main categories:

- Contiguous memory management schemes
- Non-Contiguous memory management schemes



Classification of memory management schemes

Contiguous memory management schemes:

In a Contiguous memory management scheme, each program occupies a single contiguous block of storage locations, i.e., a set of memory locations with consecutive addresses.

Single contiguous memory management schemes:

The Single contiguous memory management scheme is the simplest memory management scheme used in the earliest generation of computer systems. In this scheme, the main memory is divided into two contiguous areas or partitions. The operating systems reside permanently in one partition, generally at the lower memory, and the user process is loaded into the other partition.

Advantages of Single contiguous memory management schemes:

- Simple to implement.
- Easy to manage and design.
- In a Single contiguous memory management scheme, once a process is loaded, it is given full processor's time, and no other processor will interrupt it.

Disadvantages of Single contiguous memory management schemes:

- Wastage of memory space due to unused memory as the process is unlikely to use all the available memory space.
- The CPU remains idle, waiting for the disk to load the binary image into the main memory.
- It can not be executed if the program is too large to fit the entire available main memory space.

- It does not support multiprogramming, i.e., it cannot handle multiple programs simultaneously.

Multiple Partitioning:

The single Contiguous memory management scheme is inefficient as it limits computers to execute only one program at a time resulting in wastage in memory space and CPU time. The problem of inefficient CPU use can be overcome using multiprogramming that allows more than one program to run concurrently. To switch between two processes, the operating systems need to load both processes into the main memory. The operating system needs to divide the available main memory into multiple parts to load multiple processes into the main memory. Thus multiple processes can reside in the main memory simultaneously.

The multiple partitioning schemes can be of two types:

- Fixed Partitioning
- Dynamic Partitioning

Fixed Partitioning

The main memory is divided into several fixed-sized partitions in a fixed partition memory management scheme or static partitioning. These partitions can be of the same size or different sizes. Each partition can hold a single process. The number of partitions determines the degree of multiprogramming, i.e., the maximum number of processes in memory. These partitions are made at the time of system generation and remain fixed after that.

Advantages of Fixed Partitioning memory management schemes:

- Simple to implement.
- Easy to manage and design.

Disadvantages of Fixed Partitioning memory management schemes:

- This scheme suffers from internal fragmentation.
- The number of partitions is specified at the time of system generation.

Dynamic Partitioning

The dynamic partitioning was designed to overcome the problems of a fixed partitioning scheme. In a dynamic partitioning scheme, each process occupies only as much memory as they require when loaded for processing. Requested processes are allocated memory until the entire physical memory is exhausted or the remaining space is insufficient to hold the requesting process. In this

scheme the partitions used are of variable size, and the number of partitions is not defined at the system generation time.

Advantages of Dynamic Partitioning memory management schemes:

- Simple to implement.
- Easy to manage and design.

Disadvantages of Dynamic Partitioning memory management schemes:

- This scheme also suffers from internal fragmentation.

Non-Contiguous memory management schemes:

In a Non-Contiguous memory management scheme, the program is divided into different blocks and loaded at different portions of the memory that need not necessarily be adjacent to one another. This scheme can be classified depending upon the size of blocks and whether the blocks reside in the main memory or not.

What is paging?

Paging is a technique that eliminates the requirements of contiguous allocation of main memory. In this, the main memory is divided into **fixed-size blocks** of physical memory called frames. The size of a frame should be kept the same as that of a page to maximize the main memory and avoid external fragmentation.

Advantages of paging:

- Pages reduce external fragmentation.
- Simple to implement.
- Memory efficient.
- Due to the equal size of frames, swapping becomes very easy.
- It is used for faster access of data.

What is Segmentation?

Segmentation is a technique that eliminates the requirements of contiguous allocation of main memory. In this, the main memory is divided into **variable-size blocks** of physical memory called segments. It is based on the way the programmer follows to structure their programs. With segmented memory allocation, each job is divided into several segments of different sizes, one for each module. Functions, subroutines, stack, array, etc., are examples of such modules.

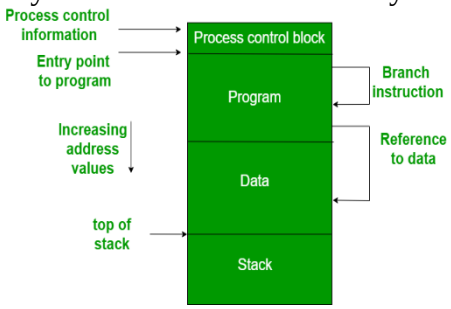
Requirements of Memory Management System

Memory management keeps track of the status of each memory location, whether it is allocated or free. It allocates the memory dynamically to the programs at their request and frees it for reuse when it is no longer needed. Memory management meant to satisfy some requirements that we should keep in mind.

These Requirements of memory management are:

1. **Relocation** – The available memory is generally shared among a number of processes in a multiprogramming system, so it is not possible to know in advance which other programs will be resident in main memory at the time of execution of this program. Swapping the active processes in and out of the main memory enables the operating system to have a larger pool of ready-to-execute process.

   When a program gets swapped out to a disk memory, then it is not always possible that when it is swapped back into main memory then it occupies the previous memory location, since the location may still be occupied by another process. We may need to **relocate** the process to a different area of memory. Thus there is a possibility that program may be moved in main memory due to swapping.



   The figure depicts a process image. The process image is occupying a continuous region of main memory. The operating system will need to know many things including the location of process control information, the execution stack, and the code entry. Within a program, there are memory references in various instructions and these are called logical addressesAfter loading of the program into main memory, the processor and the operating system must be able to translate logical addresses into physical addresses. Branch instructions contain the address of the next instruction to be executed. Data reference instructions contain the address of byte or word of data referenced.

2. **Protection –** There is always a danger when we have multiple programs at the same time as one program may write to the address space of another program. So every process must be protected against unwanted interference when other process tries to write in a process whether accidental or incidental. Between relocation and protection requirement a trade-off occurs as the satisfaction of relocation requirement increases the difficulty of satisfying the protection requirement.

   Prediction of the location of a program in main memory is not possible, that's why it is impossible to check the absolute address at compile time to assure protection. Most of the programming language allows the dynamic calculation of address at run

time. The memory protection requirement must be satisfied by the processor rather than the operating system because the operating system can hardly control a process when it occupies the processor. Thus it is possible to check the validity of memory references.

3. **Sharing –** A protection mechanism must have to allow several processes to access the same portion of main memory. Allowing each processes access to the same copy of the program rather than have their own separate copy has an advantage.

   For example, multiple processes may use the same system file and it is natural to load one copy of the file in main memory and let it shared by those processes. It is the task of Memory management to allow controlled access to the shared areas of memory without compromising the protection. Mechanisms are used to support relocation supported sharing capabilities.

4. **Logical organization –** Main memory is organized as linear or it can be a one-dimensional address space which consists of a sequence of bytes or words. Most of the programs can be organized into modules, some of those are unmodifiable (read-only, execute only) and some of those contain data that can be modified. To effectively deal with a user program, the operating system and computer hardware must support a basic module to provide the required protection and sharing. It has the following advantages:
   - Modules are written and compiled independently and all the references from one module to another module are resolved by `the system at run time.
   - Different modules are provided with different degrees of protection.
   - There are mechanisms by which modules can be shared among processes. Sharing can be provided on a module level that lets the user specify the sharing that is desired.

5. **Physical organization –** The structure of computer memory has two levels referred to as main memory and secondary memory. Main memory is relatively very fast and costly as compared to the secondary memory. Main memory is **volatile**. Thus secondary memory is provided for storage of data on a long-term basis while the main memory holds currently used programs. The major system concern between main memory and secondary memory is the flow of information and it is impractical for programmers to understand this for two reasons:
   - The programmer may engage in a practice known as overlaying when the main memory available for a program and its data may be insufficient. It allows different modules to be assigned to the same region of memory. One disadvantage is that it is time-consuming for the programmer.
   - In a multiprogramming environment, the programmer does not know how much space will be available at the time of coding and where that space will be located inside the memory.

Partitioning Algorithms

There are various algorithms which are implemented by the Operating System in order to find out the holes in the linked list and allocate them to the processes.

The explanation about each of the algorithm is given below.

1. First Fit Algorithm

First Fit algorithm scans the linked list and whenever it finds the first big enough hole to store a process, it stops scanning and load the process into that hole. This procedure produces two partitions. Out of them, one partition will be a hole while the other partition will store the process.

First Fit algorithm maintains the linked list according to the increasing order of starting index. This is the simplest to implement among all the algorithms and produces bigger holes as compare to the other algorithms.

2. Next Fit Algorithm

Next Fit algorithm is similar to First Fit algorithm except the fact that, Next fit scans the linked list from the node where it previously allocated a hole.

Next fit doesn't scan the whole list, it starts scanning the list from the next node. The idea behind the next fit is the fact that the list has been scanned once therefore the probability of finding the hole is larger in the remaining part of the list.

Experiments over the algorithm have shown that the next fit is not better then the first fit. So it is not being used these days in most of the cases.

3. Best Fit Algorithm

The Best Fit algorithm tries to find out the **smallest hole** possible in the list that can accommodate the size requirement of the process.

Using Best Fit has some disadvantages.

1. It is slower because it scans the entire list every time and tries to find out the smallest hole which can satisfy the requirement the process.
2. Due to the fact that the difference between the whole size and the process size is very small, the holes produced will be as small as it cannot be used to load any process and therefore it remains useless. Despite of the fact that the name of the algorithm is best fit, It is not the best algorithm among all.

4. Worst Fit Algorithm

The worst fit algorithm scans the entire list every time and tries to find out the **biggest hole** in the list which can fulfill the requirement of the process.

Despite of the fact that this algorithm produces the larger holes to load the other processes, this is not the better approach due to the fact that it is slower because it searches the entire list every time again and again.

5. Quick Fit Algorithm

The quick fit algorithm suggests maintaining the different lists of frequently used sizes. Although, it is not practically suggestible because the procedure takes so much time to create the different lists and then expanding the holes to load a process.

The **first fit algorithm** is **the best algorithm** among all because

1. It takes lesser time compare to the other algorithms.
2. It produces bigger holes that can be used to load other processes later on.
3. It is easiest to implement.

Buddy System – Memory allocation technique

The Buddy System is a memory allocation technique used in computer operating systems to efficiently allocate and manage memory. The technique is based on dividing the memory into fixed-size blocks, and whenever a process requests memory, the system finds the smallest available block that can accommodate the requested memory size.

Basic steps in the Buddy System memory allocation technique:

1. The memory is divided into fixed-size blocks that are a power of 2 in size (such as 2, 4, 8, 16, 32, etc. bytes).
2. Each block is labeled with its size and a unique identifier, such as a binary number.
3. Initially, all the memory blocks are free and are linked together in a binary tree structure, with each node representing a block and the tree's leaves representing the smallest available blocks.
4. When a process requests memory, the system finds the smallest available block that can accommodate the requested size. If the block is larger than the requested size, the system splits the block into two equal-sized **"buddy"** blocks.

5. The system marks one of the buddy blocks as allocated and adds it to the process's memory allocation table, while the other buddy block is returned to the free memory pool and linked back into the binary tree structure.
6. When a process releases memory, the system marks the corresponding block as free and looks for its buddy block. If the buddy block is also free, the system merges the two blocks into a larger block and links it back into the binary tree structure.

The Buddy System technique has several **advantages**, including efficient use of memory, reduced fragmentation, and fast allocation and deallocation of memory blocks. However, it also has some **drawbacks**, such as internal fragmentation, where a block may be larger than what the process requires, leading to wastage of memory. **Overall,** the Buddy System is a useful memory allocation technique in operating systems, particularly for embedded systems with limited memory.

- **If $2^{U-1}<S<=2^{U}$:** Allocate the whole block
- **Else:** Recursively divide the block equally and test the condition at each time, when it satisfies, allocate the block and get out the loop.

System also keep the record of all the unallocated blocks each and can merge these different size blocks to make one big chunk. **Advantage –**

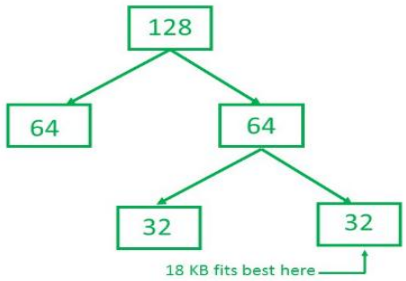
- Easy to implement a buddy system
- Allocates block of correct size
- It is easy to merge adjacent holes
- Fast to allocate memory and de-allocating memory
- Provides good memory utilization as it allocates memory blocks of the correct size and prevents unnecessary memory wastage, unlike other allocation techniques that allocate memory blocks larger than required.
- Offers a relatively simple and efficient way to manage memory allocation in systems that require dynamic memory allocation, such as embedded systems and operating systems.
- Can handle a large number of small memory allocations efficiently due to its block division mechanism, which helps prevent fragmentation and maintains system performance.
- Can prevent memory leaks by ensuring that all allocated memory is deallocated when it's no longer in use, which can improve system stability and reliability. Provides a high level of flexibility in managing memory allocation and usage, which is particularly useful in systems that require frequent memory allocation and deallocation, such as real-time systems.

Disadvantage –

- It requires all allocation unit to be powers of two
- It leads to internal fragmentation
- It requires a significant amount of overhead to manage the free memory blocks and keep track of the allocated blocks. This overhead can become a performance bottleneck in systems with limited resources.
- It is designed for fixed-sized memory allocations, and it is not suitable for variable-sized allocations. This can limit its applicability in some systems, such as databases and file systems.

- It is a general-purpose memory allocation technique and may not be optimal for all applications. Some applications may require more specialized memory allocation techniques to achieve the best performance.

Example – Consider a system having buddy system with physical address space 128 KB.



Calculate the size of partition for 18 KB process. **Solution –**
So, size of partition for 18 KB process = 32 KB. It divides by 2, till possible to get minimum block to fit 18 KB.

Unit-4

Module-2

What is Fragmentation?

Fragmentation is an unwanted problem in the operating system in which the processes are loaded and unloaded from memory, and free memory space is fragmented. Processes can't be assigned to memory blocks due to their small size, and the memory blocks stay unused. It is also necessary to understand that as programs are loaded and deleted from memory, they generate free space or a hole in the memory. These small blocks cannot be allotted to new arriving processes, resulting in inefficient memory use.

The conditions of fragmentation depend on the memory allocation system. As the process is loaded and unloaded from memory, these areas are fragmented into small pieces of memory that cannot be allocated to incoming processes. It is called **fragmentation**.

Causes of Fragmentation

User processes are loaded and unloaded from the main memory, and processes are kept in memory blocks in the main memory. Many spaces remain after process loading and swapping that another process cannot load due to their size. Main memory is available, but its space is

insufficient to load another process because of the dynamical allocation of main memory processes.

Types of Fragmentation

There are mainly two types of fragmentation in the operating system. These are as follows:

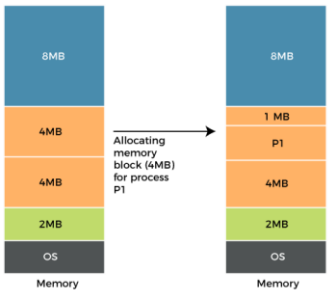
1. **Internal Fragmentation**
2. **External Fragmentation**

Internal Fragmentation

When a process is allocated to a memory block, and if the process is smaller than the amount of memory requested, a free space is created in the given memory block. Due to this, the free space of the memory block is unused, which causes **internal fragmentation**.

For Example:

Assume that memory allocation in RAM is done using fixed partitioning (i.e., memory blocks of fixed sizes). **2MB, 4MB, 4MB**, and **8MB** are the available sizes. The Operating System uses a part of this RAM.



Let's suppose a process **P1** with a size of **3MB** arrives and is given a memory block of **4MB**. As a result, the **1MB** of free space in this block is unused and cannot be used to allocate memory to another process. It is known as **internal fragmentation**.

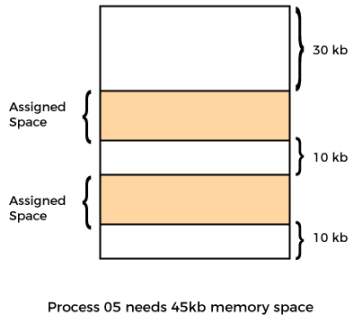
How to avoid internal fragmentation?

The problem of internal fragmentation may arise due to the fixed sizes of the memory blocks. It may be solved by assigning space to the process via **dynamic partitioning**. Dynamic partitioning allocates only the amount of space requested by the process. As a result, there is no internal fragmentation.

External Fragmentation

External fragmentation happens when a dynamic memory allocation method allocates some memory but leaves a small amount of memory unusable. The quantity of available memory is substantially reduced if there is too much external fragmentation. There is enough memory space to complete a request, but it is not contiguous. It's known as **external** fragmentation.

For Example:



Process 05 needs 45kb memory space

Let's take the example of external fragmentation. In the above diagram, you can see that there is sufficient space **(50 KB)** to run a process **(05) (need 45KB)**, but the memory is not contiguous. You can use compaction, paging, and segmentation to use the free space to execute a process.

How to remove external fragmentation?

This problem occurs when you allocate RAM to processes continuously. It is done in **paging** and **segmentation**, where memory is allocated to processes non-contiguously. As a result, if you remove this condition, external fragmentation may be decreased.

Compaction is another method for removing external fragmentation. External fragmentation may be decreased when dynamic partitioning is used for memory allocation by combining all free memory into a single large block. The larger memory block is used to allocate space based on the requirements of the new processes. This method is also known as defragmentation.

Advantages and disadvantages of fragmentation

There are various advantages and disadvantages of fragmentation. Some of them are as follows:

Advantages

There are various advantages of fragmentation. Some of them are as follows:

Fast Data Writes

Data write in a system that supports data fragmentation may be faster than reorganizing data storage to enable contiguous data writes.

Fewer Failures

If there is insufficient sequential space in a system that does not support fragmentation, the write will fail.

Storage Optimization

A fragmented system might potentially make better use of a storage device by utilizing every available storage block.

Disadvantages

There are various disadvantages of fragmentation. Some of them are as follows:

Need for regular defragmentation

A more fragmented storage device's performance will degrade with time, necessitating the requirement for time-consuming defragmentation operations.

Slower Read Times

The time it takes to read a non-sequential file might increase as a storage device becomes more fragmented.

Swapping in Operating System

Swapping is a memory management scheme in which any process can be temporarily swapped from main memory to secondary memory so that the main memory can be made available for other processes. It is used to improve main memory utilization. In secondary memory, the place where the swapped-out process is stored is called **swap space**.

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.

1. User process size is 2048Kb
2. Data transfer rate is 1Mbps = 1024 kbps
3. Time = process size / transfer rate
4. = 2048 / 1024
5. = 2 seconds
6. = 2000 milliseconds
7. 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.

Note:

- In a single tasking operating system, only one process occupies the user program area of memory and stays in memory until the process is complete.
- In a multitasking operating system, a situation arises when all the active processes cannot coordinate in the main memory, then a process is swap out from the main memory so that other processes can enter it.

Segmentation in OS

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

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

Segment table contains mainly two information about segment:

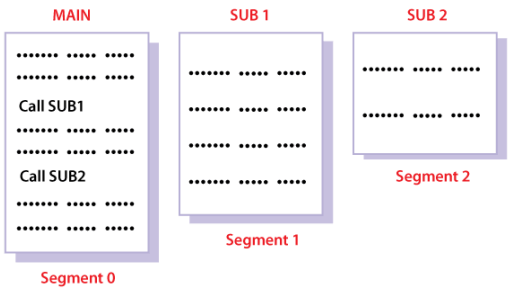
1. **Base**: It is the base address of the segment
2. **Limit**: It is the length of the segment.

Why Segmentation is required?

Till now, we were using Paging as our main memory management technique. Paging is more close to the Operating system rather than the User. It divides all the processes into the form of pages regardless of the fact that a process can have some relative parts of functions which need to be loaded in the same page.

Operating system doesn't care about the User's view of the process. It may divide the same function into different pages and those pages may or may not be loaded at the same time into the memory. It decreases the efficiency of the system.

It is better to have segmentation which divides the process into the segments. Each segment contains the same type of functions such as the 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:

1. Segment Number

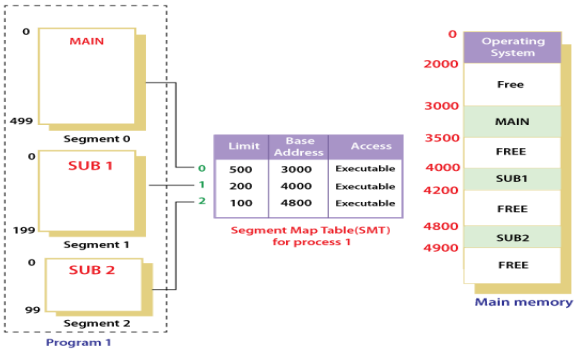
2. Offset

For Example:

Suppose a 16 bit address is used with 4 bits for the segment number and 12 bits for the segment offset so the maximum segment size is 4096 and the maximum number of segments that can be refereed is 16.

When a program is loaded into memory, the segmentation system tries to locate space that is large enough to hold the first segment of the process, space information is obtained from the free list maintained by memory manager. Then it tries to locate space for other segments. Once adequate space is located for all the segments, it loads them into their respective areas.

The operating system also generates a segment map table for each program.



With the help of segment map tables and hardware assistance, the operating system can easily translate a logical address into physical address on execution of a program.

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

The above figure shows how address translation is done in case of segmentation.

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 compared 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.

Virtual Memory in OS

Virtual Memory is a storage scheme that provides user an illusion of having a very big main memory. This is done by treating a part of secondary memory as the main memory.

In this scheme, User can load the bigger size processes than the available main memory by having the illusion that the memory is available to load the process.

Instead of loading one big process in the main memory, the Operating System loads the different parts of more than one process in the main memory.

By doing this, the degree of multiprogramming will be increased and therefore, the CPU utilization will also be increased.

How Virtual Memory Works?

In modern word, virtual memory has become quite common these days. In this scheme, whenever some pages needs to be loaded in the main memory for the execution and the memory is not available for those many pages, then in that case, instead of stopping the pages from entering in the main memory, the OS search for the RAM area that are least used in the recent times or that are not referenced and copy that into the secondary memory to make the space for the new pages in the main memory.

Since all this procedure happens automatically, therefore it makes the computer feel like it is having the unlimited RAM.

Demand Paging

Demand Paging is a popular method of virtual memory management. In demand paging, the pages of a process which are least used, get stored in the secondary memory. A page is copied to the main memory when its demand is made or page fault occurs.

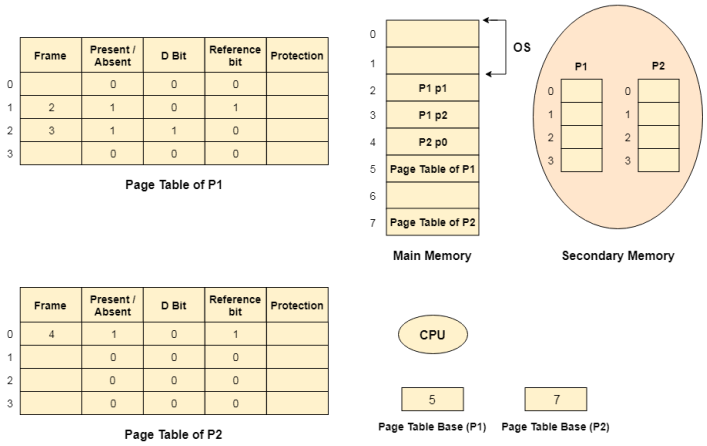
There are various page replacement algorithms which are used to determine the pages which will be replaced.

Snapshot of a virtual memory management system

Let us assume 2 processes, P1 and P2, contains 4 pages each. Each page size is 1 KB. The main memory contains 8 frame of 1 KB each. The OS resides in the first two partitions. In the third partition, 1st page of P1 is stored and the other frames are also shown as filled with the different pages of processes in the main memory.

The page tables of both the pages are 1 KB size each and therefore they can be fit in one frame each. The page tables of both the processes contain various information that is also shown in the image.

The CPU contains a register which contains the base address of page table that is 5 in the case of P1 and 7 in the case of P2. This page table base address will be added to the page number of the Logical address when it comes to accessing the actual corresponding entry.



Advantages of Virtual Memory

1. The degree of Multiprogramming will be increased.
2. User can run large application with less real RAM.
3. There is no need to buy more memory RAMs.

Disadvantages of Virtual Memory

1. The system becomes slower since swapping takes time.
2. It takes more time in switching between applications.
3. The user will have the lesser hard disk space for its use.

Page Replacement Algorithms

There are three types of Page Replacement Algorithms. They are:

- First In First Out Page Replacement Algorithm
- Optimal Page Replacement Algorithm
- Least Recently Used (LRU) Page Replacement Algorithm

First in First out Page Replacement Algorithm

This is the first basic algorithm of Page Replacement Algorithms. This algorithm is basically dependent on the number of frames used. Then each frame takes up the certain page and tries to access it. When the frames are filled then the actual problem starts. The fixed number of frames is filled up with the help of first frames present. This concept is fulfilled with the help of Demand Paging

After filling up of the frames, the next page in the waiting queue tries to enter the frame. If the frame is present then, no problem is occurred. Because of the page which is to be searched is already present in the allocated frames.

If the page to be searched is found among the frames then, this process is known as **Page Hit.**

If the page to be searched is not found among the frames then, this process is known as **Page Fault**.

When Page Fault occurs this problem arises, then the First In First Out Page Replacement Algorithm comes into picture.

The First In First Out (FIFO) Page Replacement Algorithm removes the Page in the frame which is allotted long back. This means the useless page which is in the frame for a longer time is removed and the new page which is in the ready queue and is ready to occupy the frame is allowed by the First In First Out Page Replacement.

Let us understand this First In First Out Page Replacement Algorithm working with the help of an example.

Example:

Consider the reference string 6, 1, 1, 2, 0, 3, 4, 6, 0, 2, 1, 2, 1, 2, 0, 3, 2, 1, 2, 0 for a memory with three frames and calculate number of page faults by using FIFO (First In First Out) Page replacement algorithms.

Points to Remember

Page Not Found - - - > Page Fault

Page Found - - - > Page Hit

Reference String:

6, 1, 1, 2, 0, 3, 4, 6, 0, 2, 1, 2, 1, 2, 0, 3, 2, 1, 2, 0

S. no	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
F3				2	2	2	4	4	4	2	2	2	2	2	2	2	2	2	2	0
F2		1	1	1	1	3	3	3	0	0	0	0	0	0	0	3	3	3	3	3
F1	6	6	6	6	0	0	0	6	6	6	1	1	1	1	1	1	1	1	1	1
Hit (H)/ Fault (F)	F	F	H	F	F	F	F	F	F	F	F	H	H	H	H	F	H	H	H	F

Number of Page Hits = 8

Number of Page Faults = 12

The Ratio of Page Hit to the Page Fault = 8 : 12 - - - > 2 : 3 - - - > 0.66

The Page Hit Percentage = 8 *100 / 20 = 40%

The Page Fault Percentage = 100 - Page Hit Percentage = 100 - 40 = 60%

Explanation

First, fill the frames with the initial pages. Then, after the frames are filled we need to create a space in the frames for the new page to occupy. So, with the help of First in First Out Page Replacement Algorithm we remove the frame which contains the page is older among the pages. By removing the older page we give access for the new frame to occupy the empty space created by the First in First out Page Replacement Algorithm.

OPTIMAL Page Replacement Algorithm

This is the second basic algorithm of Page Replacement Algorithms. This algorithm is basically dependent on the number of frames used. Then each frame takes up the certain page and tries to access it. When the frames are filled then the actual problem starts. The fixed number of frames is filled up with the help of first frames present. This concept is fulfilled with the help of Demand Paging

After filling up of the frames, the next page in the waiting queue tries to enter the frame. If the frame is present then, no problem is occurred. Because of the page which is to be searched is already present in the allocated frames.

If the page to be searched is found among the frames then, this process is known as **Page Hit.**

If the page to be searched is not found among the frames then, this process is known as **Page Fault**.

When Page Fault occurs this problem arises, then the OPTIMAL Page Replacement Algorithm comes into picture.

The OPTIMAL Page Replacement Algorithms works on a certain principle.

The principle is: Replace the Page which is not used in the Longest Dimension of time in future

This principle means that after all the frames are filled then, see the future pages which are to occupy the frames. Go on checking for the pages which are already available in the frames. Choose the page which is at last.

Example:

Suppose the Reference String is:

0, 3, 4, 6, 0, 2, 1, 2, 1, 2, 0, 3, 2, 1, 2, 0

6, 1, 2 are in the frames occupying the frames.

Now we need to enter 0 into the frame by removing one page from the page

So, let us check which page number occurs last

From the sub sequence 0, 3, 4, 6, 0, 2, 1 we can say that 1 is the last occurring page number. So we can say that 0 can be placed in the frame body by removing 1 from the frame.

Let us understand this OPTIMAL Page Replacement Algorithm working with the help of an example.

Example:

Consider the reference string 6, 1, 1, 2, 0, 3, 4, 6, 0, 2, 1, 2, 1, 2, 0, 3, 2, 1, 4, 0 for a memory with three frames and calculate number of page faults by using OPTIMAL Page replacement algorithms.

Points to Remember

Page Not Found - - - > Page Fault

Page Found - - - > Page Hit

Reference String:

6, 1, 1, 2, 0, 3, 4, 6, 0, 2, 1, 2, 1, 2, 0, 3, 2, 1, 4, 0

S. no	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
F3				2	2	2	4	4	4	4	4	4	4	4	4	3	3	3	4	4
F2		1	1	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1
F1	6	6	6	6	0	0	0	6	6	2	2	2	2	2	2	2	2	2	2	0
Hit (H)/ Faul t (F)	F	F	H	F	F	F	F	H	H	F	F	H	H	H	H	F	H	F	F	F

Number of Page Hits = 8

Number of Page Faults = 12

The Ratio of Page Hit to the Page Fault = 8 : 12 - - - > 2 : 3 - - - > 0.66

The Page Hit Percentage = 8 *100 / 20 = 40%

The Page Fault Percentage = 100 - Page Hit Percentage = 100 - 40 = 60%

Explanation

First, fill the frames with the initial pages. Then, after the frames are filled we need to create a space in the frames for the new page to occupy.

Here, we would fill the empty spaces with the pages we and the empty frames we have. The problem occurs when there is no space for occupying of pages. We have already known that we would replace the Page which is not used in the Longest Dimension of time in future.

There comes a question what if there is absence of page which is in the frame.

Suppose the Reference String is:

0, 2, 4, 6, 0, 2, 1, 2, 1, 2, 0, 3, 2, 1, 2, 0

6, 1, 5 are in the frames occupying the frames.

Here, we can see that page number 5 is not present in the Reference String. But the number 5 is present in the Frame. So, as the page number 5 is absent we remove it when required and other page can occupy that position.

Least Recently Used (LRU) Replacement Algorithm

This is the last basic algorithm of Page Replacement Algorithms. This algorithm is basically dependent on the number of frames used. Then each frame takes up the certain page and tries to access it. When the frames are filled then the actual problem starts. The fixed number of frames is filled up with the help of first frames present. This concept is fulfilled with the help of Demand Paging

After filling up of the frames, the next page in the waiting queue tries to enter the frame. If the frame is present then, no problem is occurred. Because of the page which is to be searched is already present in the allocated frames.

If the page to be searched is found among the frames then, this process is known as Page Hit.

If the page to be searched is not found among the frames then, this process is known as Page Fault.

When Page Fault occurs this problem arises, then the Least Recently Used (LRU) Page Replacement Algorithm comes into picture.

The Least Recently Used (LRU) Page Replacement Algorithms works on a certain principle. The principle is:

Replace the page with the page which is less dimension of time recently used page in the past.

Example:

Suppose the Reference String is:

6, 1, 1, 2, 0, 3, 4, 6, 0

The pages with page numbers 6, 1, 2 are in the frames occupying the frames.

Now, we need to allot a space for the page numbered 0.

Now, we need to travel back into the past to check which page can be replaced.

6 is the oldest page which is available in the Frame.

So, replace 6 with the page numbered 0.

Let us understand this Least Recently Used (LRU) Page Replacement Algorithm working with the help of an example.

Example:

Consider the reference string 6, 1, 1, 2, 0, 3, 4, 6, 0, 2, 1, 2, 1, 2, 0, 3, 2, 1, 2, 0 for a memory with three frames and calculate number of page faults by using Least Recently Used (LRU) Page replacement algorithms.

Points to Remember

Page Not Found - - - > Page Fault

Page Found - - - > Page Hit

Reference String:

6, 1, 1, 2, 0, 3, 4, 6, 0, 2, 1, 2, 1, 2, 0, 3, 2, 1, 2, 0

S. no	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
F3				2	2	2	4	4	4	2	2	2	2	2	2	2	2	2	2	2
F2		1	1	1	1	3	3	3	0	0	0	0	0	0	0	0	0	1	1	1
F1	6	6	6	6	0	0	0	6	6	6	1	1	1	1	1	3	3	3	3	0
Hit (H)/ Fault (F)	F	F	H	F	F	F	F	F	F	F	F	H	H	H	H	F	H	F	H	F

Number of Page Hits = 7

Number of Page Faults = 13

The Ratio of Page Hit to the Page Fault = 7 : 12 - - - > 0.5833 : 1

The Page Hit Percentage = 7 * 100 / 20 = 35%

The Page Fault Percentage = 100 - Page Hit Percentage = 100 - 35 = 65%

Explanation

First, fill the frames with the initial pages. Then, after the frames are filled we need to create a space in the frames for the new page to occupy.

Here, we would fill the empty spaces with the pages we and the empty frames we have. The problem occurs when there is no space for occupying of pages. We have already known that we would replace the Page which is not used in the Longest Dimension of time in past or can be said as the Page which is very far away in the past.

Second Chance (or Clock) Page Replacement Policy

Apart from LRU, OPT and FIFO page replacement policies, we also have the second chance/clock page replacement policy. In the Second Chance page replacement policy, the candidate pages for removal are considered in a round robin matter, and a page that has been accessed between consecutive considerations will not be replaced. The page replaced is the one

that, when considered in a round robin matter, has not been accessed since its last consideration.
It can be implemented by adding a "second chance" bit to each memory frame-every time the frame is considered (due to a reference made to the page inside it), this bit is set to 1, which gives the page a second chance, as when we consider the candidate page for replacement, we replace the first one with this bit set to 0 (while zeroing out bits of the other pages we see in the process). Thus, a page with the "second chance" bit set to 1 is never replaced during the first consideration and will only be replaced if all the other pages deserve a second chance too!
Traditionally Second Chance and Clock are believed to be less efficient than LRU (having a higher miss ratio). Recent research from Carnegie Mellon University finds that Second Chance and Clock are more efficient than LRU with a lower miss ratio. Because Second Chance and Clock is faster and more scalable than LRU, this means LRU has no advantage over Second Chance and Clock anymore.
Example –
Let's say the reference string is **0 4 1 4 2 4 3 4 2 4 0 4 1 4 2 4 3 4** and we have **3** frames. Let's see how the algorithm proceeds by tracking the second chance bit and the pointer.

- Initially, all frames are empty so after first 3 passes they will be filled with {0, 4, 1} and the second chance array will be {0, 0, 0} as none has been referenced yet. Also, the pointer will cycle back to 0.

- **Pass-4:** Frame={0, 4, 1}, second_chance = {0, 1, 0} [4 will get a second chance], pointer = 0 (No page needed to be updated so the candidate is still page in frame 0), pf = 3 (No increase in page fault number).

- **Pass-5:** Frame={2, 4, 1}, second_chance= {0, 1, 0} [0 replaced; it's second chance bit was 0, so it didn't get a second chance], pointer=1 (updated), pf=4

- **Pass-6:** Frame={2, 4, 1}, second_chance={0, 1, 0}, pointer=1, pf=4 (No change)

- **Pass-7:** Frame={2, 4, 3}, second_chance= {0, 0, 0} [4 survived but it's second chance bit became 0], pointer=0 (as element at index 2 was finally replaced), pf=5

- **Pass-8:** Frame={2, 4, 3}, second_chance= {0, 1, 0} [4 referenced again], pointer=0, pf=5

- **Pass-9:** Frame={2, 4, 3}, second_chance= {1, 1, 0} [2 referenced again], pointer=0, pf=5

- **Pass-10:** Frame={2, 4, 3}, second_chance= {1, 1, 0}, pointer=0, pf=5 (no change)

- **Pass-11:** Frame={2, 4, 0}, second_chance= {0, 0, 0}, pointer=0, pf=6 (2 and 4 got second chances)

- **Pass-12:** Frame={2, 4, 0}, second_chance= {0, 1, 0}, pointer=0, pf=6 (4 will again get a second chance)
- **Pass-13:** Frame={1, 4, 0}, second_chance= {0, 1, 0}, pointer=1, pf=7 (pointer updated, pf updated)
- **Page-14:** Frame={1, 4, 0}, second_chance= {0, 1, 0}, pointer=1, pf=7 (No change)
- **Page-15:** Frame={1, 4, 2}, second_chance= {0, 0, 0}, pointer=0, pf=8 (4 survived again due to 2nd chance!)
- **Page-16:** Frame={1, 4, 2}, second_chance= {0, 1, 0}, pointer=0, pf=8 (2nd chance updated)
- **Page-17:** Frame={3, 4, 2}, second_chance= {0, 1, 0}, pointer=1, pf=9 (pointer, pf updated)
- **Page-18:** Frame={3, 4, 2}, second_chance= {0, 1, 0}, pointer=1, pf=9 (No change)

In this example, second chance algorithm does as well as the LRU method, which is much more expensive to implement in hardware.

What is Thrash?

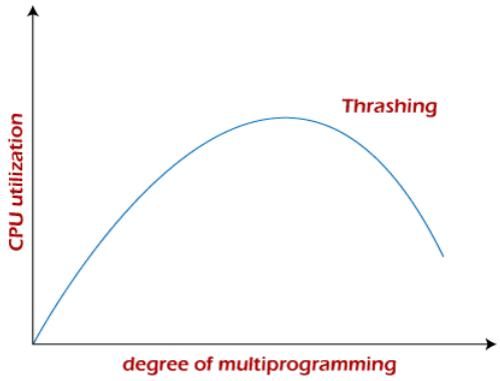
In computer science, ***thrash*** is the poor performance of a virtual memory (or paging) system when the same pages are being loaded repeatedly due to a lack of main memory to keep them in memory. Depending on the configuration and algorithm, the actual throughput of a system can degrade by multiple orders of magnitude.

In computer science, ***thrashing*** occurs when a computer's virtual memory resources are overused, leading to a constant state of paging and page faults, inhibiting most application-level processing. It causes the performance of the computer to degrade or collapse. The situation can continue indefinitely until the user closes some running applications or the active processes free up additional virtual memory resources.

To know more clearly about thrashing, first, we need to know about page fault and swapping.

- **Page fault:** We know every program is divided into some pages. A page fault occurs when a program attempts to access data or code in its address space but is not currently located in the system RAM.
- **Swapping:** Whenever a page fault happens, the operating system will try to fetch that page from secondary memory and try to swap it with one of the pages in RAM. This process is called swapping.

Thrashing is when the page fault and swapping happens very frequently at a higher rate, and then the operating system has to spend more time swapping these pages. This state in the operating system is known as thrashing. Because of thrashing, the CPU utilization is going to be reduced or negligible.



The basic concept involved is that if a process is allocated too few frames, then there will be too many and too frequent page faults. As a result, no valuable work would be done by the CPU, and the CPU utilization would fall drastically.

The long-term scheduler would then try to improve the CPU utilization by loading some more processes into the memory, thereby increasing the degree of multiprogramming. Unfortunately, this would result in a further decrease in the CPU utilization, triggering a chained reaction of higher page faults followed by an increase in the degree of multiprogramming, called thrashing.

Algorithms during Thrashing

Whenever thrashing starts, the operating system tries to apply either the Global page replacement Algorithm or the Local page replacement algorithm.

1. Global Page Replacement

Since global page replacement can bring any page, it tries to bring more pages whenever thrashing is found. But what actually will happen is that no process gets enough frames, and as a result, the thrashing will increase more and more. Therefore, the global page replacement algorithm is not suitable when thrashing happens.

2. Local Page Replacement

Unlike the global page replacement algorithm, local page replacement will select pages which only belong to that process. So there is a chance to reduce the thrashing. But it is proven that there are many disadvantages if we use local page replacement. Therefore, local page replacement is just an alternative to global page replacement in a thrashing scenario.

Causes of Thrashing

Programs or workloads may cause thrashing, and it results in severe performance problems, such as:

- If CPU utilization is too low, we increase the degree of multiprogramming by introducing a new system. A global page replacement algorithm is used. The CPU scheduler sees the decreasing CPU utilization and increases the degree of multiprogramming.
- CPU utilization is plotted against the degree of multiprogramming.
- As the degree of multiprogramming increases, CPU utilization also increases.
- If the degree of multiprogramming is increased further, thrashing sets in, and CPU utilization drops sharply.
- So, at this point, to increase CPU utilization and to stop thrashing, we must decrease the degree of multiprogramming.

How to Eliminate Thrashing

Thrashing has some negative impacts on hard drive health and system performance. Therefore, it is necessary to take some actions to avoid it. To resolve the problem of thrashing, here are the following methods, such as:

- **Adjust the swap file size:**If the system swap file is not configured correctly, disk thrashing can also happen to you.
- **Increase the amount of RAM:** As insufficient memory can cause disk thrashing, one solution is to add more RAM to the laptop. With more memory, your computer can handle tasks easily and don't have to work excessively. Generally, it is the best long-term solution.
- **Decrease the number of applications running on the computer:** If there are too many applications running in the background, your system resource will consume a lot. And the remaining system resource is slow that can result in thrashing. So while closing, some applications will release some resources so that you can avoid thrashing to some extent.
- **Replace programs:** Replace those programs that are heavy memory occupied with equivalents that use less memory.

Techniques to Prevent Thrashing

The Local Page replacement is better than the Global Page replacement, but local page replacement has many disadvantages, so it is sometimes not helpful. Therefore below are some other techniques that are used to handle thrashing:

1. Locality Model

A locality is a set of pages that are actively used together. The locality model states that as a process executes, it moves from one locality to another. Thus, a program is generally composed of several different localities which may overlap.

For example, when a function is called, it defines a new locality where memory references are made to the function call instructions, local and global variables, etc. Similarly, when the function is exited, the process leaves this locality.

2. Working-Set Model

This model is based on the above-stated concept of the Locality Model.

The basic principle states that if we allocate enough frames to a process to accommodate its current locality, it will only fault whenever it moves to some new locality. But if the allocated frames are lesser than the size of the current locality, the process is bound to thrash.

According to this model, based on parameter A, the working set is defined as the set of pages in the most recent 'A' page references. Hence, all the actively used pages would always end up being a part of the working set.

The accuracy of the working set is dependent on the value of parameter A. If A is too large, then working sets may overlap. On the other hand, for smaller values of A, the locality might not be covered entirely.

If D is the total demand for frames and WSS_i is the working set size for process i,

$D = \sum WSS_i$

Now, if 'm' is the number of frames available in the memory, there are two possibilities:

- D>m, i.e., total demand exceeds the number of frames, then thrashing will occur as some processes would not get enough frames.

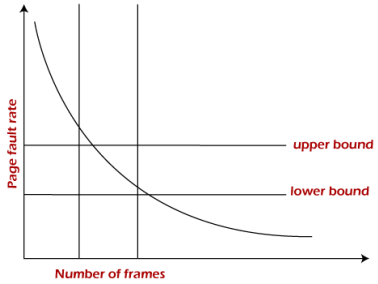
- D<=m, then there would be no thrashing.

If there are enough extra frames, then some more processes can be loaded into the memory. On the other hand, if the summation of working set sizes exceeds the frames' availability, some of the processes have to be suspended (swapped out of memory).

This technique prevents thrashing along with ensuring the highest degree of multiprogramming possible. Thus, it optimizes CPU utilization.

3. Page Fault Frequency

A more direct approach to handle thrashing is the one that uses the Page-Fault Frequency concept.



The problem associated with thrashing is the high page fault rate, and thus, the concept here is to control the page fault rate.

If the page fault rate is too high, it indicates that the process has too few frames allocated to it. On the contrary, a low page fault rate indicates that the process has too many frames.

Upper and lower limits can be established on the desired page fault rate, as shown in the diagram.

If the page fault rate falls below the lower limit, frames can be removed from the process. Similarly, if the page faults rate exceeds the upper limit, more frames can be allocated to the process.

In other words, the graphical state of the system should be kept limited to the rectangular region formed in the given diagram.

If the page fault rate is high with no free frames, some of the processes can be suspended and allocated to them can be reallocated to other processes. The suspended processes can restart later.

Unit-5

Module-1

I/O Management

Categories of I/O Devices

• Human readable
– Used to communicate with the user
– Printers
– Video display terminals
–Keyboard
–Mouse

• Machine readable
– Used to communicate with electronic equipment
– Disk and tape drives
– Sensors
– Controllers
– Actuators

• Communication
– Used to communicate with remote devices
– Digital line drivers
– Modems

Evolution of the I/O Function

• Processor directly controls a peripheral device

• Controller of I/O module is added
– Processor uses programmed I/O without interrupts
– Processor does not need to handle details of external devices

• Controller or I/O module with interrupts
– Processor does not spend time waiting for an I/O operation to be performed

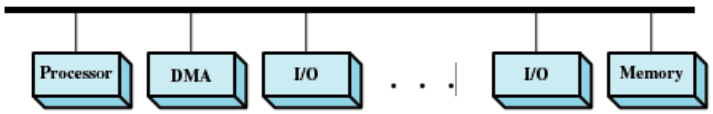
• Direct Memory Access
– Blocks of data are moved into memory without involving the processor
– Processor involved at beginning and end only

• I/O module is a separate processor
– CPU directs I/O processor to execute program in main memory

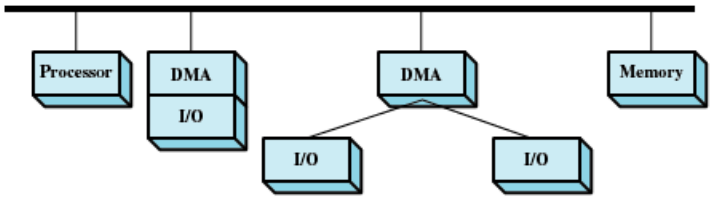
• I/O processor
– I/O module now has its own local memory
– It's a computer in its own right

Direct Memory Access
• Processor delegates I/O operation to the DMA module
• DMA module transfers data directly to or form memory
• When complete DMA module sends an interrupt signal to the processor

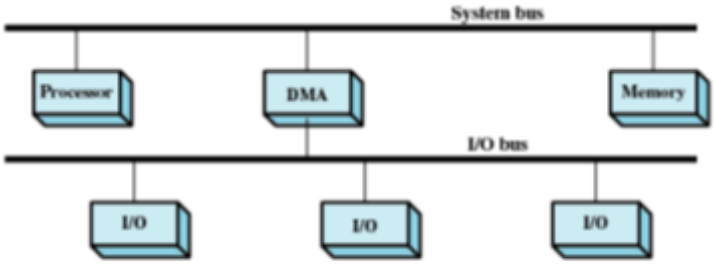
DMA Configurations



(a) Single-bus, detached DMA



(b) Single-bus, Integrated DMA-I/O



(c) I/O bus

Alternative DMA Configuration

Operating System Design Issues

• **Efficiency**
– Most I/O devices extremely slow compared to main memory
– Use of multiprogramming allows for some processes to be waiting on I/O while another process executes
– I/O cannot keep up with processor speed
– Swapping is used to bring in additional *Ready Processes*, which is an I/O operation

• **Generality**
– Desirable to handle all I/O devices in a uniform manner
– Hide most of the details of device I/O in lower-level routines so that processes and upper levels see devices in general terms such as read, write, open, close, lock, unlock

I/O Buffering

• **Reasons for buffering**
– Processes must wait for I/O to complete before proceeding
– Certain pages must remain in main memory during I/O

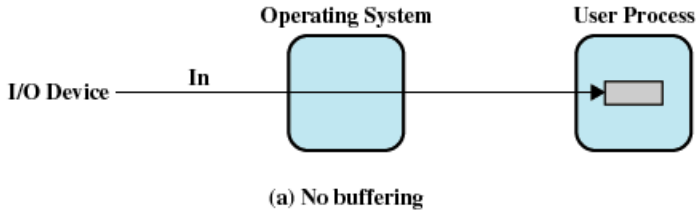
• **Block-oriented**
– Information is stored in fixed sized blocks
– Transfers are made a block at a time
– Used for disks and tapes

• **Stream-oriented**
– Transfer information as a stream of bytes
– Used for terminals, printers, communication ports, mouse and other pointing devices, and most other devices that are not secondary storage

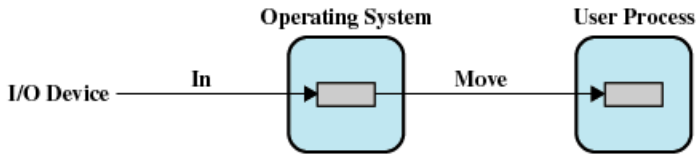
Single Buffer

• Operating system assigns a buffer in main memory for an I/O request

• Block-oriented
– Input transfers made to buffer
– Block moved to user space when needed
– Another block is moved into the buffer • "Read ahead"
– User process can process one block of data while next block is read in
– Swapping can occur since input is taking place in system memory, not user memory
– Operating system keeps track of assignment of system buffers to user processes
• Stream-oriented
– e.g. terminal
• Used a line at time
• User input from a terminal is one line at a time with carriage return signaling the end of the line
• Output to the terminal is one line at a time
– e.g. network I/O
• NIC (**n**etwork **i**nterface **c**ard)
• protocol stack



(a) No buffering

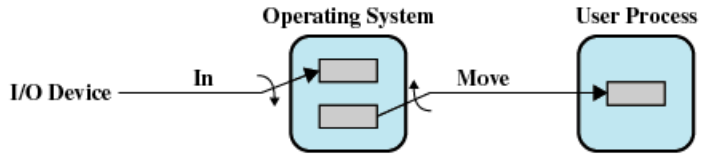


(b) Single buffering

Double Buffer

• Use two system buffers instead of one

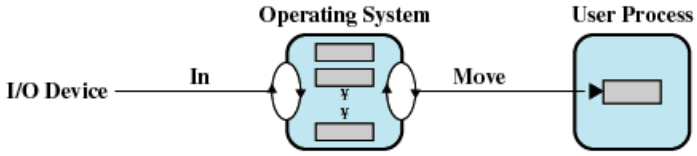
• A process can transfer data to or from one buffer while the operating system empties or fills the other buffer



(c) Double buffering

Circular Buffer

• More than two buffers are used
• Each individual buffer is one unit in a circular buffer
• Used when I/O operation must keep up with Process



(d) Circular buffering

Disk Scheduling Algorithms in OS

As we know, a process needs two type of time, CPU time and IO time. For I/O, it requests the Operating system to access the disk.

However, the operating system must be fare enough to satisfy each request and at the same time, operating system must maintain the efficiency and speed of process execution.

The technique that operating system uses to determine the request which is to be satisfied next is called disk scheduling.

Let's discuss some important terms related to disk scheduling.

Seek Time

Seek time is the time taken in locating the disk arm to a specified track where the read/write request will be satisfied.

Rotational Latency

It is the time taken by the desired sector to rotate itself to the position from where it can access the R/W heads.

Transfer Time

It is the time taken to transfer the data.

Disk Access Time

Disk access time is given as,

Disk Access Time = Rotational Latency + Seek Time + Transfer Time

Disk Response Time

It is the average of time spent by each request waiting for the IO operation.

Purpose of Disk Scheduling

The main purpose of disk scheduling algorithm is to select a disk request from the queue of I/O requests and decide the schedule when this request will be processed.

Goal of Disk Scheduling Algorithm

- Fairness
- High throughout
- Minimal traveling head time

Disk Scheduling Algorithms

The list of various disks scheduling algorithm is given below. Each algorithm is carrying some advantages and disadvantages. The limitation of each algorithm leads to the evolution of a new algorithm.

- FCFS scheduling algorithm
- SSTF (shortest seek time first) algorithm
- SCAN scheduling
- C-SCAN scheduling
- LOOK Scheduling
- C-LOOK scheduling

FCFS Scheduling Algorithm

It is the simplest Disk Scheduling algorithm. It services the IO requests in the order in which they arrive. There is no starvation in this algorithm, every request is serviced.

Disadvantages

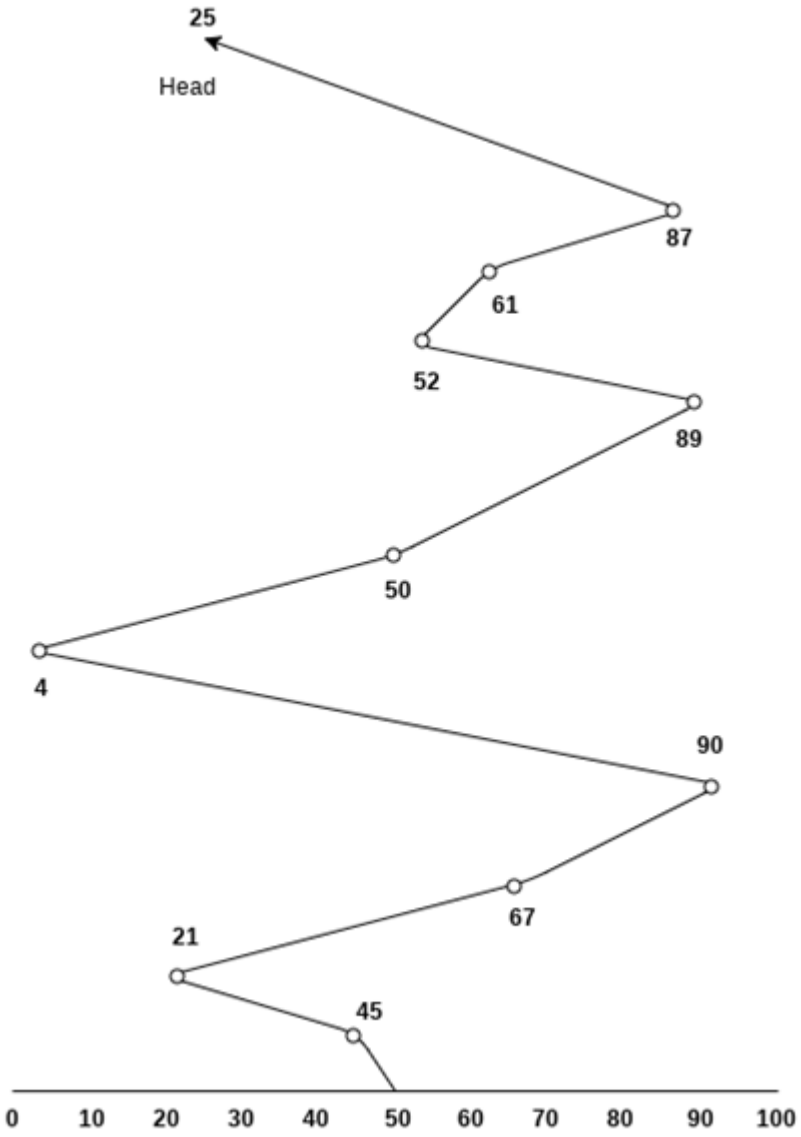
- The scheme does not optimize the seek time.
- The request may come from different processes therefore there is the possibility of inappropriate movement of the head.

Example

Consider the following disk request sequence for a disk with 100 tracks 45, 21, 67, 90, 4, 50, 89, 52, 61, 87, 25

Head pointer starting at 50 and moving in left direction. Find the number of head movements in cylinders using FCFS scheduling.

Solution



Number of cylinders moved by the head

= (50-45)+(45-21)+(67-21)+(90-67)+(90-4)+(50-4)+(89-50)+(61-52)+(87-61)+(87-25)

= 5 + 24 + 46 + 23 + 86 + 46 + 49 + 9 + 26 + 62

= 376

SSTF Scheduling Algorithm

Shortest seek time first (SSTF) algorithm selects the disk I/O request which requires the least disk arm movement from its current position regardless of the direction. It reduces the total seek time as compared to FCFS.

It allows the head to move to the closest track in the service queue.

Disadvantages

- It may cause starvation for some requests.
- Switching direction on the frequent basis slows the working of algorithm.
- It is not the most optimal algorithm.

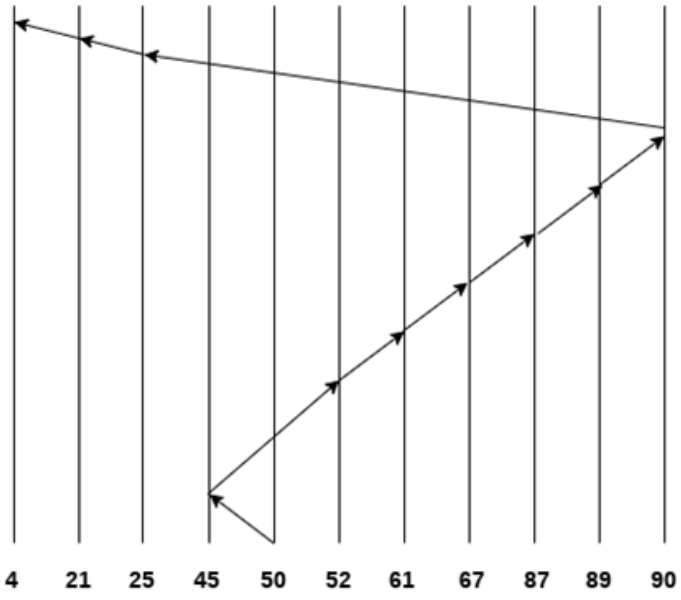
Example

Consider the following disk request sequence for a disk with 100 tracks

45, 21, 67, 90, 4, 89, 52, 61, 87, 25

Head pointer starting at 50. Find the number of head movements in cylinders using SSTF scheduling.

Solution:



Number of cylinders = 5 + 7 + 9 + 6 + 20 + 2 + 1 + 65 + 4 + 17 = 136

SCAN and C-SCAN algorithm

Scan Algorithm

It is also called as Elevator Algorithm. In this algorithm, the disk arm moves into a particular direction till the end, satisfying all the requests coming in its path,and then it turns backand moves in the reverse direction satisfying requests coming in its path.

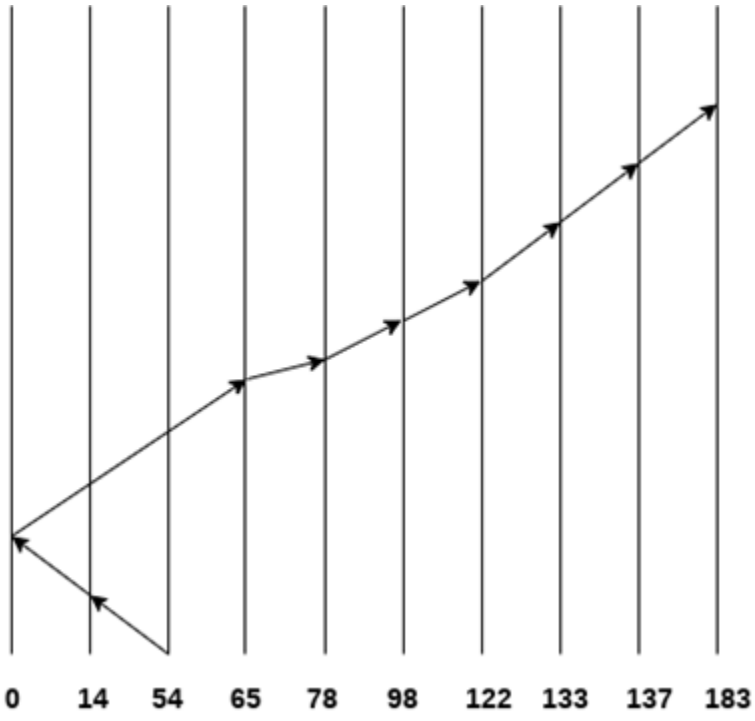
It works in the way an elevator works, elevator moves in a direction completely till the last floor of that direction and then turns back.

Example

Consider the following disk request sequence for a disk with 100 tracks

98, 137, 122, 183, 14, 133, 65, 78

Head pointer starting at 54 and moving in left direction. Find the number of head movements in cylinders using SCAN scheduling.



Number of Cylinders = 40 + 14 + 65 + 13 + 20 + 24 + 11 + 4 + 46 = 237

C-SCAN algorithm

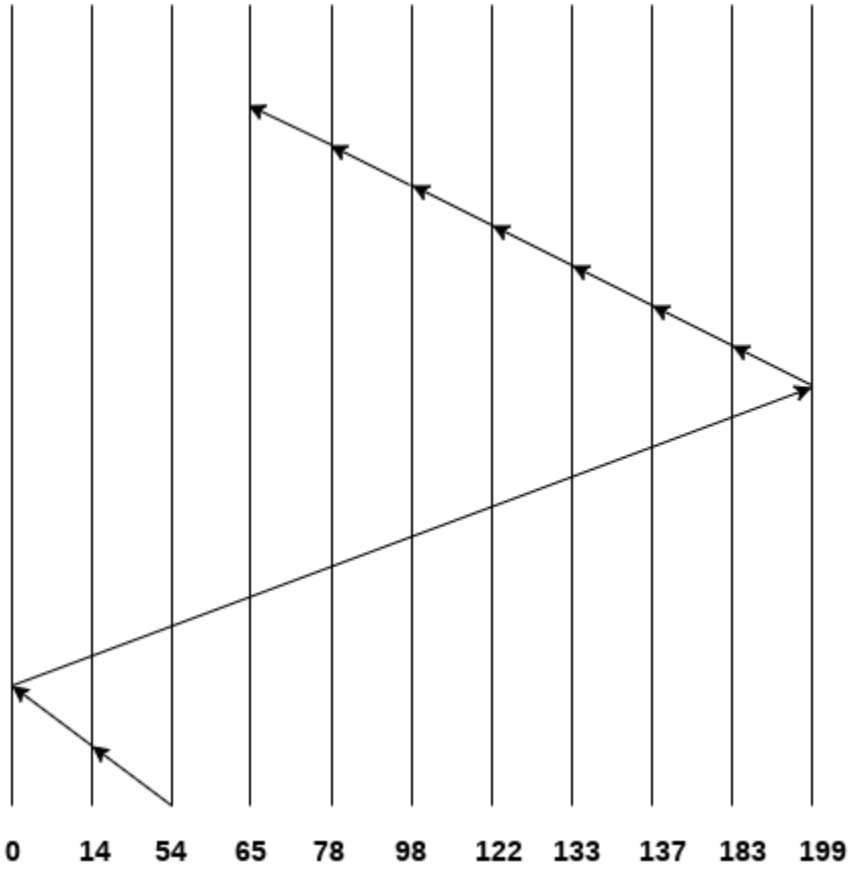
In C-SCAN algorithm, the arm of the disk moves in a particular direction servicing requests until it reaches the last cylinder, then it jumps to the last cylinder of the opposite direction without servicing any request then it turns back and start moving in that direction servicing the remaining requests.

Example

Consider the following disk request sequence for a disk with 100 tracks

98, 137, 122, 183, 14, 133, 65, 78

Head pointer starting at 54 and moving in left direction. Find the number of head movements in cylinders using C-SCAN scheduling.



No. of cylinders crossed = 40 + 14 + 199 + 16 + 46 + 4 + 11 + 24 + 20 + 13 = 387

Look Scheduling

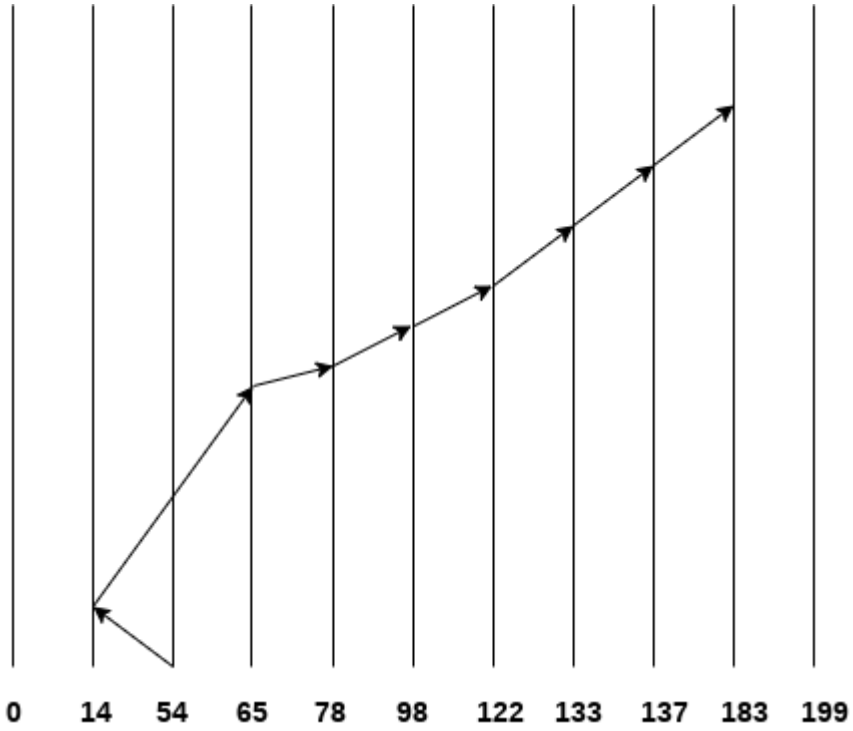
It is like SCAN scheduling Algorithm to some extant except the difference that, in this scheduling algorithm, the arm of the disk stops moving inwards (or outwards) when no more request in that direction exists. This algorithm tries to overcome the overhead of SCAN algorithm which forces disk arm to move in one direction till the end regardless of knowing if any request exists in the direction or not.

Example

Consider the following disk request sequence for a disk with 100 tracks

98, 137, 122, 183, 14, 133, 65, 78

Head pointer starting at 54 and moving in left direction. Find the number of head movements in cylinders using LOOK scheduling.



Number of cylinders crossed = 40 + 51 + 13 + +20 + 24 + 11 + 4 + 46 = 209

C Look Scheduling

C Look Algorithm is similar to C-SCAN algorithm to some extent. In this algorithm, the arm of the disk moves outwards servicing requests until it reaches the highest request cylinder, then it jumps to the lowest request cylinder without servicing any request then it again start moving outwards servicing the remaining requests.

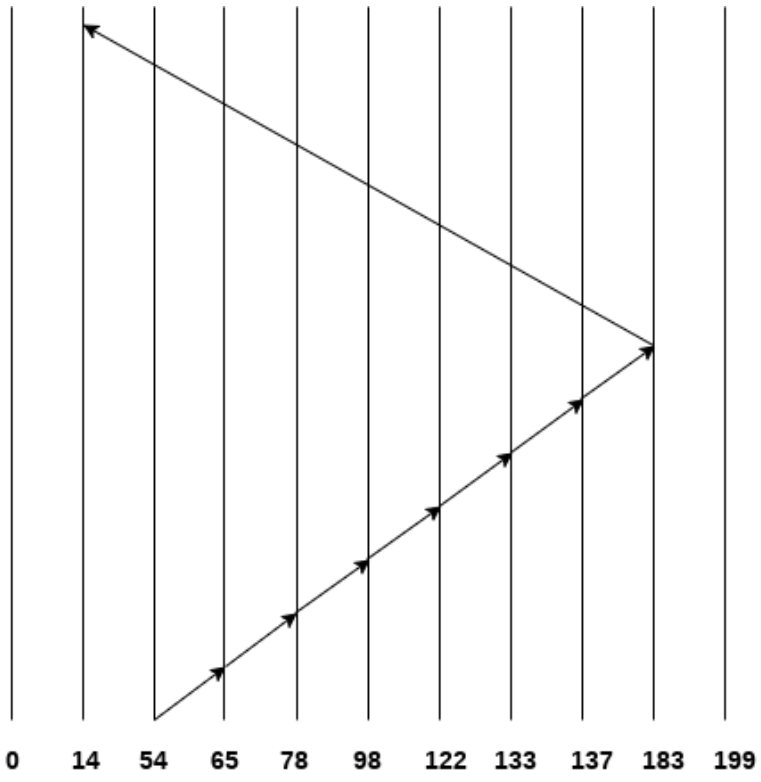
It is different from C SCAN algorithm in the sense that, C SCAN force the disk arm to move till the last cylinder regardless of knowing whether any request is to be serviced on that cylinder or not.

Example

Consider the following disk request sequence for a disk with 100 tracks

98, 137, 122, 183, 14, 133, 65, 78

Head pointer starting at 54 and moving in left direction. Find the number of head movements in cylinders using C LOOK scheduling.



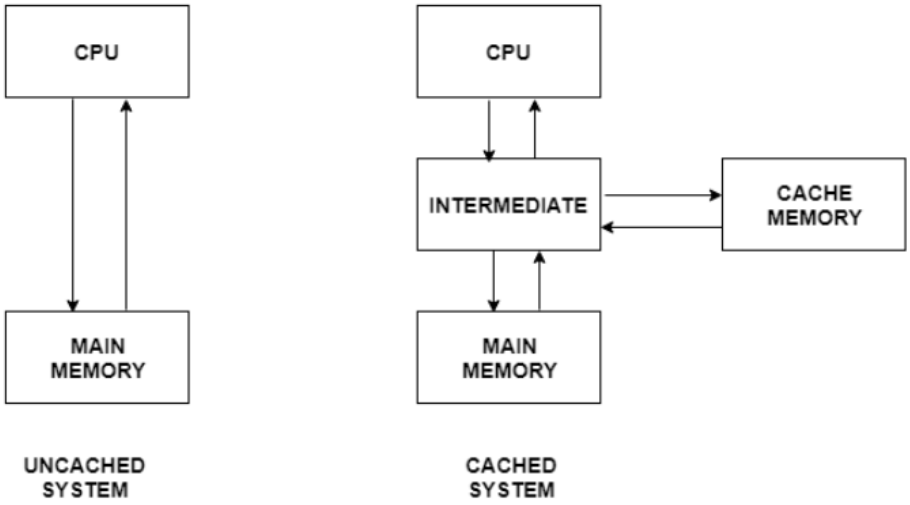
Number of cylinders crossed = 11 + 13 + 20 + 24 + 11 + 4 + 46 + 169 = 298

Disk Caches

Cache is a type of memory that is used to increase the speed of data access. Normally, the data required for any process resides in the main memory. However, it is transferred to the cache memory temporarily if it is used frequently enough. The process of storing and accessing data from a cache is known as **caching.**

Uncached System vs Cached System

A figure to better understand the difference between cached and uncached system is as follows −



Some important points to explain the above figure are −

- In an uncached system, there is no cache memory. So, all the data required by the processor during execution is obtained from main memory. This is a comparatively time consuming process.
- In contrast to this, a cached system contains a cache memory. Any data required by the processor is searched in the cache memory first. If it is not available there then main memory is searched. The cache system yields faster results than the uncached system because cache is much faster than main memory.

Advantages of Cache Memory

Some of the advantages of cache memory are as follows −

- Cache memory is faster than main memory as it is located on the processor chip itself. Its speed is comparable to the processor registers and so frequently required data is stored in the cache memory.

- The memory access time is considerably less for cache memory as it is quite fast. This leads to faster execution of any process.
- The cache memory can store data temporarily as long as it is frequently required. After the use of any data has ended, it can be removed from the cache and replaced by new data from the main memory.

Disadvantages of Cache Memory

Some of the disadvantages of cache memory are as follows −

- Since the cache memory is quite fast, it is extremely useful in any computer system. However, it is also quite expensive and so is used judiciously.
- The cache is memory expensive as observed from the previous point. Also, it is located directly on the processor chip. Because of these reasons, it has a limited capacity and is much smaller than main memory.

Unit-5

Module-2

Files

A ***file*** is a named collection of related information that is recorded on secondary storage such as magnetic disks, magnetic tapes and optical disks.

In general, a file is a sequence of bits, bytes, lines or records whose meaning is defined by the files creator and user.

Attributes of a File

Following are some of the attributes of a file:

Name: It is the only information which is in human-readable form.

Identifie: The file is identified by a unique tag(number) within file system.

Type: It is needed for systems that support different types of files.

Location: Pointer to file location on device.

Size: The current size of the file.

Protection: This controls and assigns the power of reading, writing, executing.

Time, date, and user identification: This is the data for protection, security, and usage monitoring.

File Operations

The operating system must do to perform basic file operations given below.

• **Creating a file:** Two steps are necessary to create a file. First, space in the file system must be found for the file. Second, an entry for the new file must be made in the directory.

• **Writing a file:** To write a file, we make a system call specifying both the name of the file and the information to be written to the file. Given the name of the file, the system searches the directory to find the file's location. The system must keep a write pointer to the location in the file where the next write is to take place. The write pointer must be updated whenever a write occurs.

• **Reading a file:** To read from a file, we use a system call that specifies the name of the file and where (in memory) the next block of the file should be put. Again, the directory is searched for the associated entry, and the system needs to keep a read pointer to the location in the file where the next read is to take place. Once the read has taken place, the read pointer is updated.

• **Repositioning within a file**: The directory is searched for the appropriate entry, and the current-file-position pointer is repositioned to a given value. Repositioning within a file need not involve any actual I/O. This file operation is also known as a **file seek**.

• **Deleting a file.** To delete a file, we search the directory for the named file. Having found the associated directory entry, we release all file space, so that it can be reused bv other files, and erase the directory entry.

• **Protection:** Access-control information determines who can do reading, writing, executing, and so on.

• **Truncating a file:** The user may want to erase the contents of a file but keep its attributes. Rather than forcing the user to delete the file and then recreate it, this function allows all attributes to remain unchanged—except for file length—but lets the file be reset to length zero and its file space released.

File System Structure

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

• A file has a certain defined structure according to its type.

• A text file is a sequence of characters organized into lines.

• A source file is a sequence of procedures and functions.

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

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

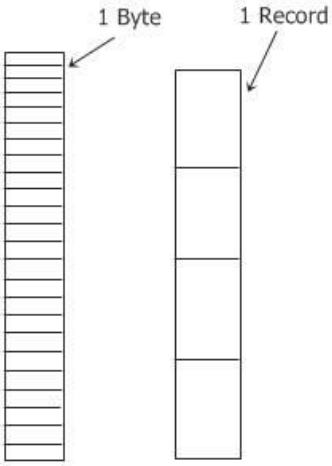


Fig 1 Fig 2

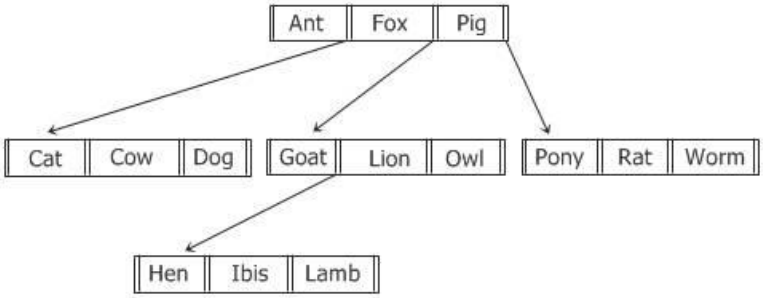


Fig 3

File Structure 1

• Here, as you can see from the figure 1, the file is an unstructured sequence of bytes.

• Therefore, the OS doesn't care about what is in the file, as all it sees are bytes.

File Structure 2

• Now, as you can see from the figure 2 that shows the second structure of a file, where a file is a sequence of fixed-length records where each with some internal structure.

• Central to the idea about a file being a sequence of records is the idea that read operation returns a record and write operation just appends a record.

File Structure 3

• Now in the last structure of a file that you can see in the figure 3, a file basically consists of a tree of records, not necessarily all the same length, each containing a key field in a fixed position in the record. The tree is stored on the field, just to allow the rapid searching for a specific key.

File Access method

File access mechanism refers to the manner in which the records of a file may be accessed. There are several ways to access files −

• Sequential access

• Direct/Random access

• Indexed sequential access

1. Sequential Access

• A sequential access is that in which the records are accessed in some sequence, i.e., the information in the file is processed in order, one record after the other. This access method is the most primitive one.

• The idea of Sequential access is based on the **tape model** which is a sequential access device.

• The Sequential access method is best because most of the records in a file are to be processed. For example, transaction files.

Example: Compilers usually access files in this fashion.

Advantages of sequential access
• It is simple to program and easy to design.
• Sequential file is best use if storage space.

Disadvantages of sequential access
• Sequential file is time consuming process.
• It has high data redundancy.
• Random searching is not possible.

2.Direct Access
• Sometimes it is not necessary to process every record in a file.
• It is not necessary to process all the records in the order in which they are present in the memory. In all such cases, direct access is used.
• The **disk** is a direct access device which gives us the reliability to random access of any file block.
• In the file, there is a collection of physical blocks and the records of that blocks.

Example: Databases are often of this type since they allow query processing that involves immediate access to large amounts of information. All reservation systems fall into this category.

Advantages:
• Direct access file helps in online transaction processing system (OLTP) like online railway reservation system.
• In direct access file, sorting of the records are not required.
• It accesses the desired records immediately.
• It updates several files quickly.
• It has better control over record allocation.

Disadvantages:
• Direct access file does not provide backup facility.
• It is expensive.
• It has less storage space as compared to sequential file.

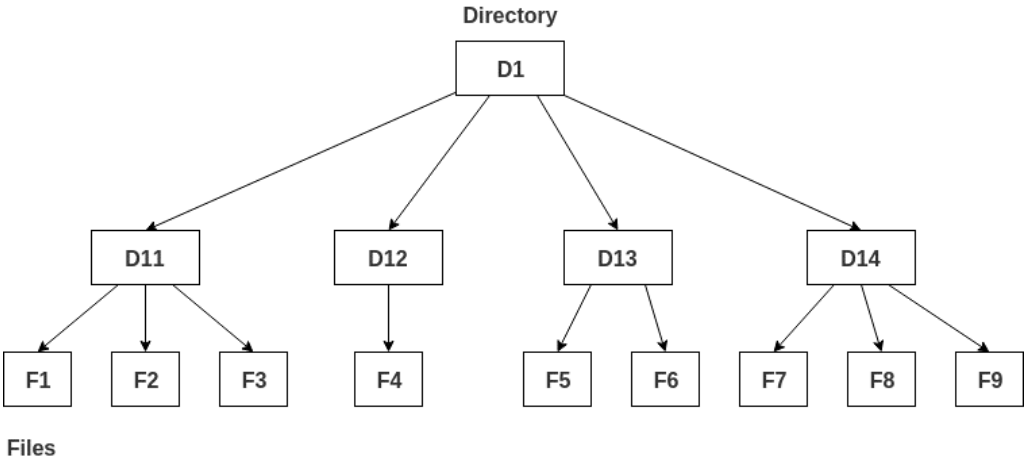
3.Indexed Sequential Access
• The index sequential access method is a modification of the direct access method.
• Basically, it is kind of combination of both the sequential access as well as direct access.
• The main idea of this method is to first access the file directly and then it accesses sequentially.
• In this access method, it is necessary for maintaining an index.
• The index is nothing but a pointer to a block.
• The direct access of the index is made to access a record in a file.
• The information which is obtained from this access is used to access the file. Sometimes the indexes are very big.

• It is built on top of Sequential access.
• It uses an Index to control the pointer while accessing files.

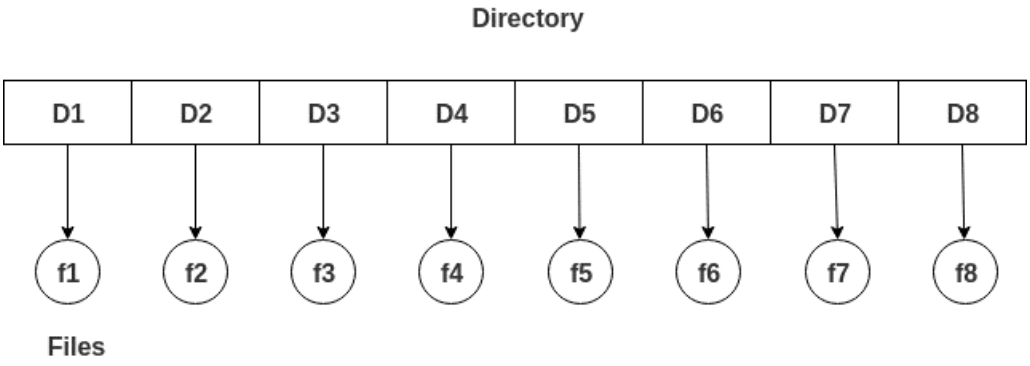
Advantages:
• In indexed sequential access file, sequential file and random file access is possible.
• It accesses the records very fast if the index table is properly organized.
• The records can be inserted in the middle of the file.
• It provides quick access for sequential and direct processing.
• It reduces the degree of the sequential search.

Disadvantages:
• Indexed sequential access file requires unique keys and periodic reorganization.
• Indexed sequential access file takes longer time to search the index for the data access or retrieval.
• It requires more storage space.
• It is expensive because it requires special software.
• It is less efficient in the use of storage space as compared to other file organizations.

Directory Structure
• A directory is a container that is used to contain folders and file.
• It organizes files and folders into a hierarchical manner.



1. Single-level directory –
• Single level directory is simplest directory structure.
• In it all files are contained in same directory which make it easy to support and understand.
• A single level directory has a significant limitation, however, when the number of files increases or when the system has more than one user.
• Since all the files are in the same directory, they must have the unique name. if two users call their dataset test, then the unique name rule violated.



Advantages:

• Since it is a single directory, so its implementation is very easy.

• If files are smaller in size, searching will faster.

• The operations like file creation, searching, deletion, updating are very easy in such a directory structure.

Disadvantages:

• There may chance of name collision because two files cannot have the same name.

• Searching will become time taking if directory will large.

• In this cannot group the same type of files together.

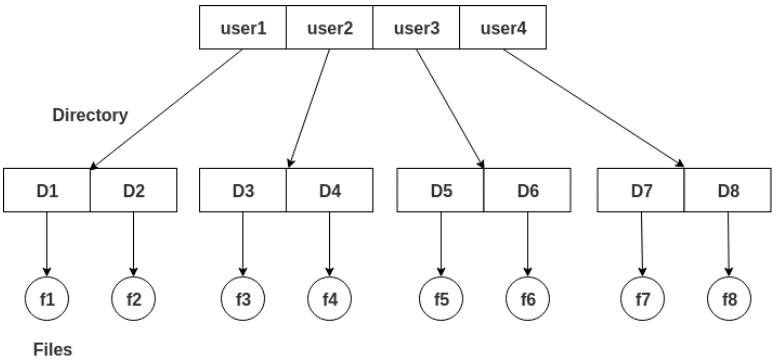
2. Two-level directory –

• As, a single level directory often leads to confusion of files names among different users hence the solution to this problem is to create a separate directory for each user.

• In the two-level directory structure, each user has their own *user files directory (UFD)*.

• The UFDs has similar structures, but each lists only the files of a single user. system's *master file directory (MFD)* is searches whenever a new user id=s logged in.

• The MFD is indexed by username or account number, and each entry points to the UFD for that user.



Advantages:

• We can give full path like /User-name/directory-name/.

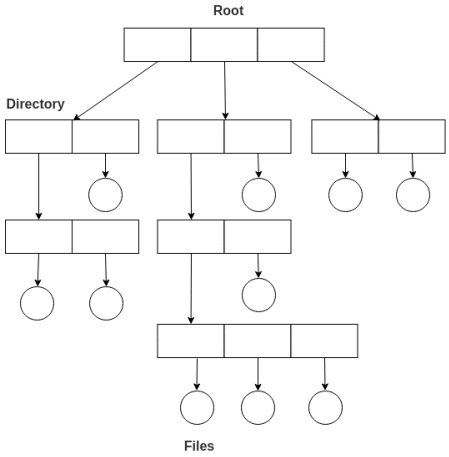
• Different users can have same directory as well as file name.

• Searching of files become more easy due to path name and user-grouping.

Disadvantages:

• A user is not allowed to share files with other users.
• Still it not very scalable, two files of the same type cannot be grouped together in the same user.

3. Tree-structured directory –
• Once we have seen a two-level directory as a tree of height 2, the natural generalization is to extend the directory structure to a tree of arbitrary height.
• This generalization allows the user to create their own subdirectories and to organize on their files accordingly.
• A tree structure is the most common directory structure. The tree has a root directory, and every file in the system have a unique path.



Advantages:
• Very generalize, since full path name can be given.
• Very scalable, the probability of name collision is less.
• Searching becomes very easy, we can use both absolute path as well as relative.

Disadvantages:
• Every file does not fit into the hierarchical model; files may be saved into multiple directories.
• We cannot share files.
• It is inefficient, because accessing a file may go under multiple directories.

File Sharing
File sharing refers to the process of sharing or distributing electronic files such as documents, music, videos, images, and software between two or more users or computers.

Importance of file sharing
File sharing plays a vital role in facilitating collaboration and communication among individuals and organizations. It allows people to share files quickly and easily across different locations, reducing the need for physical meetings and enabling remote work. File sharing also helps individuals and organizations save time and money, as it eliminates the need for physical transportation of files.

Types of File Sharing

File sharing refers to the practice of distributing or providing access to digital files, such as documents, images, audio, and video files, between two or more users or devices. There are several types of file sharing methods available, and each method has its own unique advantages and disadvantages.

- **Peer-to-Peer (P2P) File Sharing** − Peer-to-peer file sharing allows users to share files with each other without the need for a centralized server. Instead, users connect to each other directly and exchange files through a network of peers. P2P file sharing is commonly used for sharing large files such as movies, music, and software.
- **Cloud-Based File Sharing** − Cloud-based file sharing involves the storage of files in a remote server, which can be accessed from any device with an internet connection. Users can upload and download files from cloud-based file sharing services such as Google Drive, Dropbox, and OneDrive. Cloud-based file sharing allows users to easily share files with others, collaborate on documents, and access files from anywhere.
- **Direct File Transfer** − Direct file transfer involves the transfer of files between two devices through a direct connection such as Bluetooth or Wi-Fi Direct. Direct file transfer is commonly used for sharing files between mobile devices or laptops.
- **Removable Media File Sharing** − Removable media file sharing involves the use of physical storage devices such as USB drives or external hard drives. Users can copy files onto the device and share them with others by physically passing the device to them.

Risks of File Sharing

File sharing is a convenient and efficient way to share information and collaborate on projects. However, it comes with several risks and challenges that can compromise the confidentiality, integrity, and availability of files. In this section, we will explore some of the most significant risks of file sharing.

- **Malware and Viruses** − One of the most significant risks of file sharing is the spread of malware and viruses. Files obtained from untrusted sources, such as peer-to-peer (P2P) networks, can contain malware that can infect the user's device and compromise the security of their files. Malware and viruses can cause damage to the user's device, steal personal information, or even use their device for illegal activities without their knowledge.
- **Data Breaches and Leaks** − Another significant risk of file sharing is the possibility of data breaches and leaks. Cloud-based file sharing services and P2P networks are particularly vulnerable to data breaches if security measures are not implemented properly. Data breaches can result in the loss of sensitive information, such as personal data or intellectual property, which can have severe consequences for both individuals and organizations.
- **Legal Consequences** − File sharing copyrighted material without permission can lead to legal consequences. Sharing copyrighted music, movies, or software can result in copyright infringement lawsuits and hefty fines.

- **Identity Theft** − File sharing can also expose users to identity theft. Personal information, such as login credentials or social security numbers, can be inadvertently shared through file sharing if security measures are not implemented properly. Cybercriminals can use this information to commit identity theft, which can have severe consequences for the victim.

File Sharing Protection Measures

- **Encryption** − Encryption is the process of converting data into a coded language that can only be accessed by authorized users with a decryption key. This can help protect files from unauthorized access and ensure that data remains confidential even if it is intercepted during file sharing.
- **Password protection** − Password protection involves securing files with a password that must be entered before the file can be accessed. This can help prevent unauthorized access to files and ensure that only authorized users can view or modify the files.
- **Secure file transfer protocols** − Secure file transfer protocols, such as SFTP (Secure File Transfer Protocol) and HTTPS (Hypertext Transfer Protocol Secure), provide a secure way to transfer files over the internet. These protocols use encryption and other security measures to protect files from interception and unauthorized access during transfer.
- **Firewall protection** − Firewall protection involves using a firewall to monitor and control network traffic to prevent unauthorized access to the user's device or network. Firewalls can also be configured to block specific file sharing protocols or limit access to certain users or devices, providing an additional layer of protection for shared files.

Best Practices for Secure File Sharing

- **Use trusted sources for file sharing** − To reduce the risk of downloading malware or viruses, it is essential to use trusted sources for file sharing. Users should only download files from reputable sources and avoid downloading files from unknown or suspicious websites.
- **Limit access to files** − To minimize the risk of data breaches or leaks, users should limit access to files only to authorized individuals or groups. This can be done by using password protection, encryption, and other access control measures.
- **Educate users on safe file sharing practices** − Educating users on safe file sharing practices can help reduce the risk of security incidents. Users should be trained on how to identify and avoid phishing scams, how to recognize suspicious files or emails, and how to securely share files.
- **Regularly update antivirus and anti-malware software** − To ensure maximum protection against malware and viruses, it is essential to regularly update antivirus and anti-malware software. This will help to identify and remove any potential threats to the user's device or network.

Record Blocking in OS:

Records are the logical unit of a file whereas blocks are units of I/O with secondary storage. It performs for I/O and the records must organize as blocks. **Blocking** is a process of grouping similar records into blocks that the operating system component explores exhaustively. Block is the unit of I/O for secondary storage. Generally, the larger blocks reduce the I/O transfer time. Larger blocks require larger I/O buffers. In record blocking, the records are grouped into blocks by shared properties that are indicators of duplication.

Generally, there are three types of record-blocking methods.

1. Fixed blocking
2. Variable-length spanned blocking
3. Variable-length unspanned blocking

Fixed blocking:

In this method, record lengths are fixed. The prescribed number of records stored in a block. Internal fragmentation is stored in a block. Fixed blocking is common for sequential files with fixed-length records

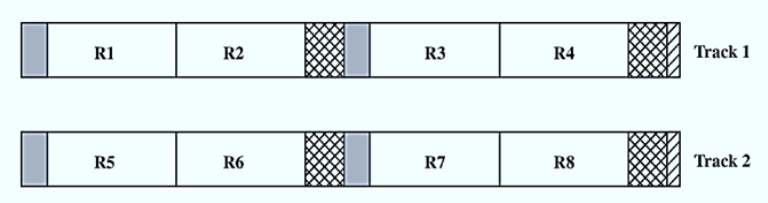


Fig: Fixed Blocking

Variable-length spanned blocking:

In this method, record sizes aren't same, Variable-length records packing into blocks with no unused space. So, some records may divide into two blocks. In this type of situation, a pointer passes from one block to another block. So, the Pointers used to span block unused space. It is efficient in length and efficiency of storage. It doesn't limit record size, but it is more complicated to implement. So, the files are more difficult to update

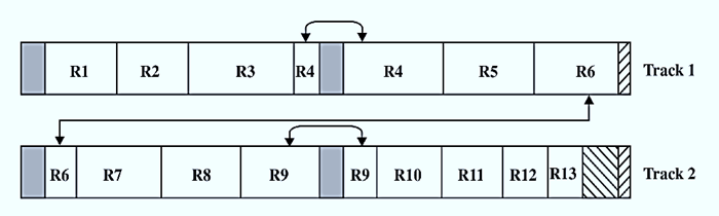


Fig: Variable-length spanned blocking

Variable-length unspanned blocking:

Here, records are variable in length, but the records span between blocks. In this method, the wasted area is a serious problem, because of the inability to use the remainder of a block, if the next record is larger than the remaining unused space. These blocking methods result in blocking results in wasted space and limit record size to the size of the block.

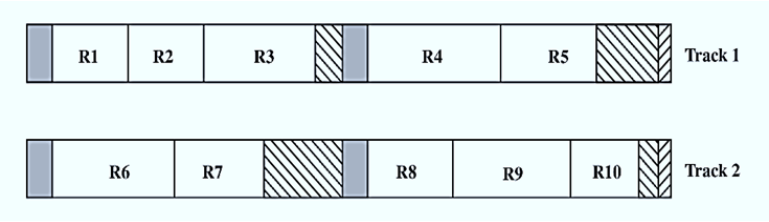


Fig: Variable-length unspanned blocking

Secondary storage management

Secondary storage management in operating system is the process of managing the computer's secondary storage. Secondary storage is the computer's long-term memory, where data is stored for future use. This is in contrast to primary storage, which is the computer's short-term memory, where data is stored for immediate use. Secondary storage management is responsible for keeping track of what data is stored on the secondary storage devices, and for making sure that the data is organized in a way that is efficient and easy to access. It is also responsible for ensuring that the data on the secondary storage devices is backed up and safe from corruption or loss.

What is Windows File System?

Windows mainly support FAT (File Allocation Table) and NTFS (New Technology File system). Windows NT 4.0, Windows 200, Windows XP, Windows .NET server and Windows workstation use NTFS as their preferred file system. Still, FAT can be used with floppy disks and older Windows versions (for multi-boot systems). FAT is the initial file system used in Windows. FAT was used with DOS, and its three versions are FAT12, FAT16 and FAT32. The number of bits used to identify a cluster is the number that is used as the suffix in the name. FAT12, FAT16 and FAT32 have 32MB, 4GB and 32GB as the maximum partition sizes.

NTFS has completely different data organization architecture. Basically, Microsoft developed NTFS to compete with UNIX, by replacing the much more simple FAT. However, the newest FAT version called exFAT is claimed to have certain advantages over NTFS. A FAT partition can be easily converted to a NTFS partition without loosing data. NTFS supports features like indexing, quota tracking, encryption, compression and repair points. Windows uses drive letter to distinguish partitions. Traditionally, The C drive is the primary partition. Primary partition is used to install and boot Windows. Drive letter can be used for mapping network drives as well.

What is Linux File System?

A variety of files systems can be used with Linux. Commonly used file systems are ext* family (ext, ext2, ext3 and ext4) and XFS. Silicon Graphics developed XFS, which is a journaling system with high performance. The ext (extended file system) was developed in early 1990's. It was the first file system used in Linux operating system. Remy Card developed it by getting inspiration from the UFS (UNIX File System).

On Linux, everything is a file. If something is not a file, then it is a process. Programs, audio, video, I/O devices and other devices are considered as files. In Linux, there is no difference between a file and a directory. A directory is simply a file containing names of a set of other files. Special files are a mechanism used for I/O (found in /dev). Sockets (another special file type) provide inter-process communication. Named pipes (much like sockets) are used for inter-process communication without network semantics.

What is the difference between Linux File System and Windows File System?

Windows uses FAT and NTFS as file systems, while Linux uses a variety of file systems. Unlike Windows, Linux is bootable from a network drive. In contrast to Windows, everything is either a file or a process in Linux. Linux has two kinds of major partitions called data partitions and swap partitions. Because of the existence of swap partitions, you never run out of memory in Linux (like in windows). In terms of recovery tools, only a limited number of tools can be used on Windows, while there is a large number of UNIX based recovery tools available for Linux file systems.

Computer security

Computer security refers to protecting and securing computers and their related data, networks, software, hardware from unauthorized access, misuse, theft, information loss, and other security issues. The Internet has made our lives easier and has provided us with lots of advantages but it has also put our system's security at risk of being infected by a virus, of being hacked, information theft, damage to the system, and much more.

Technology is growing day by day and the entire world is in its grasp. We cannot imagine even a day without electronic devices around us. With the use of this growing technology, invaders, hackers and thieves are trying to harm our computer's security for monetary gains, recognition

purposes, ransom demands, bullying others, invading into other businesses, organizations, etc. In order to protect our system from all these risks, computer security is important.

Types of computer security

Computer security can be classified into four types:

1. Cyber Security: Cyber security means securing our computers, electronic devices, networks , programs, systems from cyber attacks. Cyber attacks are those attacks that happen when our system is connected to the Internet.
2. Information Security: Information security means protecting our system's information from theft, illegal use and piracy from unauthorized use. Information security has mainly three objectives: confidentiality, integrity, and availability of information.
3. Application Security: Application security means securing our applications and data so that they don't get hacked and also the databases of the applications remain safe and private to the owner itself so that user's data remains confidential.
4. Network Security: Network security means securing a network and protecting the user's information about who is connected through that network. Over the network hackers steal, the packets of data through sniffing and spoofing attacks, man in the middle attack, war driving, etc, and misuse the data for their benefits.

Types of cyber attack

1. Denial of service attack or DOS: A denial of service attack is a kind of cyber attack in which the attackers disrupt the services of the particular network by sending infinite requests and temporary or permanently making the network or machine resources unavailable to the intended audience.
2. Backdoor: In a backdoor attack, malware, trojan horse or virus gets installed in our system and start affecting it's security along with the main file. Consider an example: suppose you are installing free software from a certain website on the Internet. Now, unknowingly, along with this software, a malicious file also gets installed, and as soon as you execute the installed software that file's malware gets affected and starts affecting your computer security. This is known as a backdoor.
3.Eavesdropping: Eavesdropping refers to secretly listening to someone's talk without their permission or knowledge. Attackers try to steal, manipulate, modify, hack information or systems by passively listening to network communication, knowing passwords etc. A physical example would be, suppose if you are talking to another person of your organization and if a third person listens to your private talks then he/ she is said to eavesdrop on your conversation. Similarly, your conversation on the internet maybe eavesdropped by attackers listening to your private conversation by connecting to your network if it is insecure.
4. Phishing: In phishing, a user is tricked by the attacker who gains the trust of the user or acts as if he is a genuine person and then steals the information by ditching. Not only attackers but some certain websites that seem to be genuine, but actually they are fraud sites. These sites trick the users and they end up giving their personal information such as login details or bank details or card number etc. Phishing is of many types: Voice phishing, text phishing etc.

5. Spoofing: Spoofing is the act of masquerading as a valid entity through falsification of data(such as an IP address or username), in order to gain access to information or resources that one is otherwise unauthorized to obtain. Spoofing is of several types- email spoofing, IP address spoofing, MAC spoofing , biometric spoofing etc. When an email is sent from a fake sender address, asking the recipient to provide sensitive data.
6. Malware: Malware is made up of two terms: Malicious + Software = Malware. Malware intrudes into the system and is designed to damage our computers. Different types of malware are adware, spyware, ransomware, Trojan horse, etc.
7. Social engineering: Social engineering attack involves manipulating users psychologically and extracting confidential or sensitive data from them by gaining their trust. The attacker generally exploits the trust of people or users by relying on their cognitive basis.
8. Polymorphic Attacks: Poly means "many" and morph means "form", polymorphic attacks are those in which attacker adopts multiple forms and changes them so that they are not recognized easily. These kinds of attacks are difficult to detect due to their changing forms.

Steps to ensure computer security

In order to protect our system from the above-mentioned attacks, users should take certain steps to ensure system security:

1. Always keep your Operating System up to date. Keeping it up to date reduces the risk of their getting attacked by malware, viruses, etc.

2. Always use a secure network connection. One should always connect to a secure network. Public wi-fi's and unsecured networks should be avoided as they are at risk of being attacked by the attacker.

3. Always install an Antivirus and keep it up to date. An antivirus is software that scans your PC against viruses and isolates the infected file from other system files so that they don't get affected. Also, we should try to go for paid anti-viruses as they are more secure.

4. Enable firewall. A firewall is a system designed to prevent unauthorized access to/from a computer or even to a private network of computers. A firewall can be either in hardware, software or a combination of both.

5. Use strong passwords. Always make strong passwords and different passwords for all social media accounts so that they cannot be key logged, brute forced or detected easily using dictionary attacks. A strong password is one that has 16 characters which are a combination of upper case and lower case alphabets, numbers and special characters. Also, keep changing your passwords regularly.

6. Don't trust someone easily. You never know someone's intention, so don't trust someone easily and end up giving your personal information to them. You don't know how they are going to use your information.

7. Keep your personal information hidden. Don't post all your personal information on social media. You never know who is spying on you. As in the real world, we try to avoid talking to strangers and sharing anything with them. Similarly, social media also have people whom you don't know and if you share all your information on it you may end up troubling yourself.

8. Don’t download attachments that come along with e-mails unless and until you know that e-mail is from a genuine source. Mostly, these attachments contain malware which, upon execution infect or harms your system.

9. Don’t purchase things online from anywhere. Make sure whenever you are shopping online you are doing so from a well-known website. There are multiple fraud websites that may steal your card information as soon as you checkout and you may get bankrupt by them.

10. Learn about computer security and ethics. You should be well aware of the safe computing and ethics of the computing world. Gaining appropriate knowledge is always helpful in reducing cyber-crime.

11. If you are attacked, immediately inform the cyber cell so that they may take appropriate action and also protect others from getting attacked by the same person. Don’t hesitate to complain just because you think people may make your fun.

12. Don’t use pirated content. Often, people try to download pirated movies, videos or web series in order to get them for free. These pirated content are at major risk of being infected with viruses, worms, or malware, and when you download them you end up compromising your system security.

What is meant by Intruder in Network Security?

An intruder is an unauthorized person or entity that tries to access a system or network without authorization with the intent of doing harm, stealing data, or interfering with regular operations.

The intrusive party might be a hacker, or someone seeking to take advantage of a system weakness. The hacker may enter the network or system via a variety of methods, including software flaws, social engineering, and password cracking.

After the hacker has access to the network or system, they could try to steal important data, put malware on the system, or seize control of it. This may result in a number of security risks, including identity theft, denial-of-service assaults, and data breaches.

To stop unauthorized access and defend against attackers, network security mechanisms like firewalls, intrusion detection systems, and access restrictions are put in place. In order to recognise and reduce possible security concerns, it's also crucial to conduct regular security audits and vulnerability assessments.

Malwares – Malicious Software

Malware is a software that gets into the system without user consent with an intention to steal private and confidential data of the user that includes bank details and password. They also generates annoying pop up ads and makes changes in system settings

They get into the system through various means:

1. Along with free downloads.
2. Clicking on suspicious link.
3. Opening mails from malicious source.

4. Visiting malicious websites.
5. Not installing an updated version of antivirus in the system.

Types:

1. Virus
2. Worm
3. Logic Bomb
4. Trojan/Backdoor
5. Rootkit
6. Advanced Persistent Threat
7. Spyware and Adware

What is computer virus:

Computer virus refers to a program which damages computer systems and/or destroys or erases data files. A computer virus is a malicious program that self-replicates by copying itself to another program. In other words, the computer virus spreads by itself into other executable code or documents. The purpose of creating a computer virus is to infect vulnerable systems, gain admin control and steal user sensitive data. Hackers design computer viruses with malicious intent and prey on online users by tricking them.

Symptoms:

- Letter looks like they are falling to the bottom of the screen.
- The computer system becomes slow.
- The size of available free memory reduces.
- The hard disk runs out of space.
- The computer does not boot.

Types of Computer Virus:

These are explained as following below.

1. **Parasitic –**
   These are the executable (.COM or .EXE execution starts at first instruction). Propagated by attaching itself to particular file or program. Generally resides at the start (prepending) or at the end (appending) of a file, e.g. Jerusalem.
2. **Boot Sector –**
   Spread with infected floppy or pen drives used to boot the computers. During system boot, boot sector virus is loaded into main memory and destroys data stored in hard disk, e.g. Polyboot, Disk killer, Stone, AntiEXE.
3. **Polymorphic –**
   Changes itself with each infection and creates multiple copies. Multipartite: use more than one propagation method. >Difficult for antivirus to detect, e.g. Involutionary, Cascade, Evil, Virus 101., Stimulate.
   Three major parts: Encrypted virus body, Decryption routine varies from infection to infection, and Mutation engine.
4. **Memory Resident –**
   Installs code in the computer memory. Gets activated for OS run and damages all files opened at that time, e.g. Randex, CMJ, Meve.
5. **Stealth –**
   Hides its path after infection. It modifies itself hence difficult to detect and masks the size of infected file, e.g. Frodo, Joshi, Whale.

6. **Macro –**
   Associated with application software like word and excel. When opening the infected document, macro virus is loaded into main memory and destroys the data stored in hard disk. As attached with documents; spreads with those infected documents only, e.g. DMV, Melissa, A, Relax, Nuclear, Word Concept.
7. **Hybrids –**
   Features of various viruses are combined, e.g. Happy99 (Email virus).

Worm:

A worm is a destructive program that fills a computer system with self-replicating information, clogging the system so that its operations are slowed down or stopped.

Types of Worm:

1. **Email worm –** Attaching to fake email messages.
2. **Instant messaging worm –** Via instant messaging applications using loopholes in network.
3. **Internet worm –** Scans systems using OS services.
4. **Internet Relay Chat (IRC) worm –** Transfers infected files to web sites.
5. **Payloads –** Delete or encrypt file, install backdoor, creating zombie etc.
6. **Worms with good intent –** Downloads application patches.

Logical Bomb:

A logical bomb is a destructive program that performs an activity when a certain action has occurred. These are hidden in programming code. Executes only when a specific condition is met, e.g. Jerusalem.

Script Virus:

Commonly found script viruses are written using the Visual Basic Scripting Edition (VBS) and the JavaScript programming language.

Trojan / Backdoor:

Trojan Horse is a destructive program. It usually pretends as computer games or application software. If executed, the computer system will be damaged. Trojan Horse usually comes with monitoring tools and key loggers. These are active only when specific events are alive. These are hidden with packers, crypters and wrappers.< Hence, difficult to detect through antivirus. These can use manual removal or firewall precaution.

RootKits:

Collection of tools that allow an attacker to take control of a system.

- Can be used to hide evidence of an attacker's presence and give them backdoor access.
- Can contain log cleaners to remove traces of attacker.
- Can be divided as:
  – Application or file rootkits: replaces binaries in Linux system
  – Kernel: targets kernel of OS and is known as a loadable kernel module (LKM)
- Gains control of infected m/c by:
  – DLL injection: by injecting malicious DLL (dynamic link library)
  – Direct kernel object manipulation: modify kernel structures and directly target trusted part of OS
  – Hooking: changing applicant's execution flow

Advanced Persistent Threat:
Created by well funded, organized groups, nation-state actors, etc. Desire to compromise government and commercial entities, e.g. Flame: used for reconnaissance and information gathering of system.

Spyware and Adware:
Normally gets installed along with free software downloads. Spies on the end-user, attempts to redirect the user to specific sites. Main tasks: Behavioral surveillance and advertising with pop up ads Slows down the system.